Sqlmap

Description

SQLmap est un outil open source qui permet d'automatiser la détection et l'exploitation de vulnérabilités de type injection SQL. Il permet ainsi de tester la sécurité des bases de données utilisées par les applications web en identifiant les potentielles failles de sécurité sur celles-ci.

Installation

Voici le guide d'installation pour l'outil SQLMap :

Étape 1 : téléchargement du paquet git afin de cloner le dépôt GitHub de SQLMap

- Sous Linux (Debian/Ubuntu) :

```
sudo apt-get install git
```

- Sous Linux (Red Hat/Fedora) :

```
sudo yum install git
```

Étape 2 : clonage du dépôt GitHub de SQLMap vers un dossier en local (sqlmap-latest)

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

Étape 3 : accès et lancement de SQLMap

- Se déplacer dans le répertoire précédemment créé avec la commande suivante :

```
cd sqlmap-latest
```

- Vérifiez l'installation en lançant SQLmap avec la commande suivante :

```
python3 sqlmap.py --version
```

- SQLmap devrait afficher sa version, ce qui confirme que l'installation s'est correctement déroulée.

Cas d'utilisation

- **Identification de la Structure de la Base de Données** : SQLmap peut extraire des informations sur la structure de la base de données, notamment les tables, les colonnes et les utilisateurs. Cela peut être utile pour comprendre la structure de la base de données d'une application cible.
- **Exploitation de base de données** : SQLmap permet la détection d'injections SQL dans une application web, il analyse les entrées utilisateur pour rechercher des méthodes d'exploitations appropriées en fonction du type de base de données, des paramètres de celle-ci, etc.
- **Tests et contournement d'authentification** : SQLmap peut être utilisé pour tester l'authentification d'une application en tentant de contourner les mécanismes d'authentification en utilisant des techniques d'injection SQL sur des formulaires ou des accès privés par exemple.

Fonctionnalités principales

- **Détection automatique** : automatise la détection des vulnérabilités SQL en analysant les paramètres de requête HTTP, en identifiant les erreurs SQL, en effectuant des tests d'injection SQL, etc.
- **Prise en charge de multiples bases de données** : compatible avec une large gamme de bases de données, notamment MySQL, PostgreSQL, Microsoft SQL Server, Oracle, SQLite, et bien d'autres.
- **Support de l'injection basée sur les erreurs** : peut identifier les erreurs SQL dans les réponses HTTP afin de déterminer si une vulnérabilité SQL existe et si elle peut être exploitée.
- **Options de personnalisation** : offre de nombreuses options pour personnaliser les tests d'injection SQL, y compris la possibilité de spécifier des en-têtes personnalisées, des cookies, des méthodes HTTP, un User-Agent custom, etc.

- **Gestion des sessions** : permet de conserver les sessions pour les applications nécessitant une authentification.

Manuel des options principales

```
-h, --help : affiche l'aide et la liste des options disponibles.

-u URL, --url=URL : spécifie l'URL cible pour l'analyse.

-r FILE, --file=FILE : spécifie un fichier contenant une requête HTTP à analyser (plutôt qu'une URL).

--data=DATA : fournit des données POST pour une requête (utilisé avec -u ou -r).

--cookie=COOKIE : fournit des cookies pour l'authentification (utilisé avec -u ou -r).

--level=LEVEL : définit le niveau de tests d'injection SQL (1-5, 1 étant le moins intrusif, 5 le plus
intrusif).

--risk=RISK : définit le niveau de risque pour les tests d'injection SQL (1-3, 1 étant le moins risqué,
3 le plus risqué).

--dbms=DBMS : spécifie le type de base de données à cibler (MySQL, PostgreSQL, Microsoft SQL Server,
etc.).

--technique=TECHNIQUE : spécifie la technique d'injection SQL à utiliser (parmi une liste de techniques
spécifiques).

--banner : affiche la bannière de SQLmap lors de son démarrage.

--batch : active le mode batch, qui ne nécessite pas d'interaction utilisateur.

--dbs : affiche la liste des bases de données disponibles sur le serveur.

--tables : affiche la liste des tables dans la base de données cible.

--columns : affiche la liste des colonnes dans une table spécifiée.

--dump : extrait les données de la base de données cible.

--os-shell : active un shell interactif sur le système d'exploitation distant (lorsqu'une injection SQL
permet cela).

--tamper=TAMPER : spécifie des scripts de contournement personnalisés pour modifier les données avant
l'injection.

--user-agent=AGENT : spécifie un agent utilisateur personnalisé dans les en-têtes HTTP.

--random-agent : utilise un agent utilisateur aléatoire pour masquer la présence de SQLmap.

--tor : dirige les requêtes SQL via le réseau Tor pour l'anonymat.

--proxy=PROXY : utilise un proxy pour rediriger les requêtes.

--threads=THREADS : définit le nombre de threads à utiliser pour les tests (parallélisation).

--time-sec=TIME_SEC : définit le temps d'attente maximal pour chaque requête.

--flush-session : efface les données de session enregistrées précédemment.

--purge-output : ssupprime les données de sortie de fichiers existants.

--exclude=EXCLUDE : exclut spécifiquement une option de test (par exemple, "--exclude=users").

-v, --verbose : active le mode verbeux, qui affiche davantage d'informations de débogage.

--wizard : active le mode assistant pour guider l'utilisateur dans la configuration des tests.

--eval=CODE : évalue le code Python personnalisé pendant l'exécution de SQLmap.

--sqlshell : active le shell SQL interactif.
```

```
--sql-file=SQL_FILE : spécifie un fichier contenant des requêtes SQL personnalisées à exécuter.
```

Exemple d'exploitation ou d'utilisation

Supposons que vous soyez chargé de tester la sécurité d'un site web e-commerce, et que vous ayez découvert une vulnérabilité d'injection SQL potentielle dans le formulaire de recherche du site. Vous souhaitez utiliser SQLmap pour exploiter cette vulnérabilité et extraire des données sensibles de la base de données du site.

Voici les étapes du chemin d'exploitation :

- **1 - Identification de la vulnérabilité** : vous avez identifié un point d'entrée potentiel pour une injection SQL dans le champ de recherche du site.
- **2 - Lancement de l'analyse** : vous lancez SQLmap pour analyser le formulaire de recherche et le paramètre associé afin de découvrir de potentielles vulnérabilités dans celui-ci. Vous utilisez l'option “-u” pour spécifier l'URL du site :

```
sqlmap -u "https://www.example.com/search?q=test"
```

- **3 - Détection de la vulnérabilité** : SQLmap identifie une vulnérabilité grâce à une des injections SQL dans le formulaire de recherche et affiche les résultats avec le type de vulnérabilité trouvé.
- **4 - Extraction de données** : vous utilisez SQLmap pour afficher la liste des bases de données disponibles sur le serveur en question :

```
sqlmap -u "https://www.example.com/search?q=test" --dbs
```

- **5 - Sélection de la base de données** : vous sélectionnez la base de données que vous souhaitez explorer parmi celles détectées par SQLmap.
- **6 - Lister les tables** : vous sélectionnez la ou les table(s) de la base de données que vous avez précédemment sélectionnée :

```
sqlmap -u "https://www.example.com/search?q=test" -D database_name --tables
```

- **7 - Extraction des données de table** : vous procédez à l'extraction des données à partir d'une table spécifique. Vous pouvez extraire des informations clients depuis la table “clients” par exemple :

```
sqlmap -u "https://www.example.com/search?q=test" -D database_name -T clients --dump
```

- **8 - {{Analyse des données** : vous pouvez ainsi analyser les données extraites afin d'identifier des informations sensibles, telles que les informations de paiement des clients.

References

URL :

- https://github.com/sqlmapproject/sqlmap/wiki/Usage
- https://book.hacktricks.xyz/pentesting-web/sql-injection/sqlmap

Retour fiches outils

- Cyber fiches outils