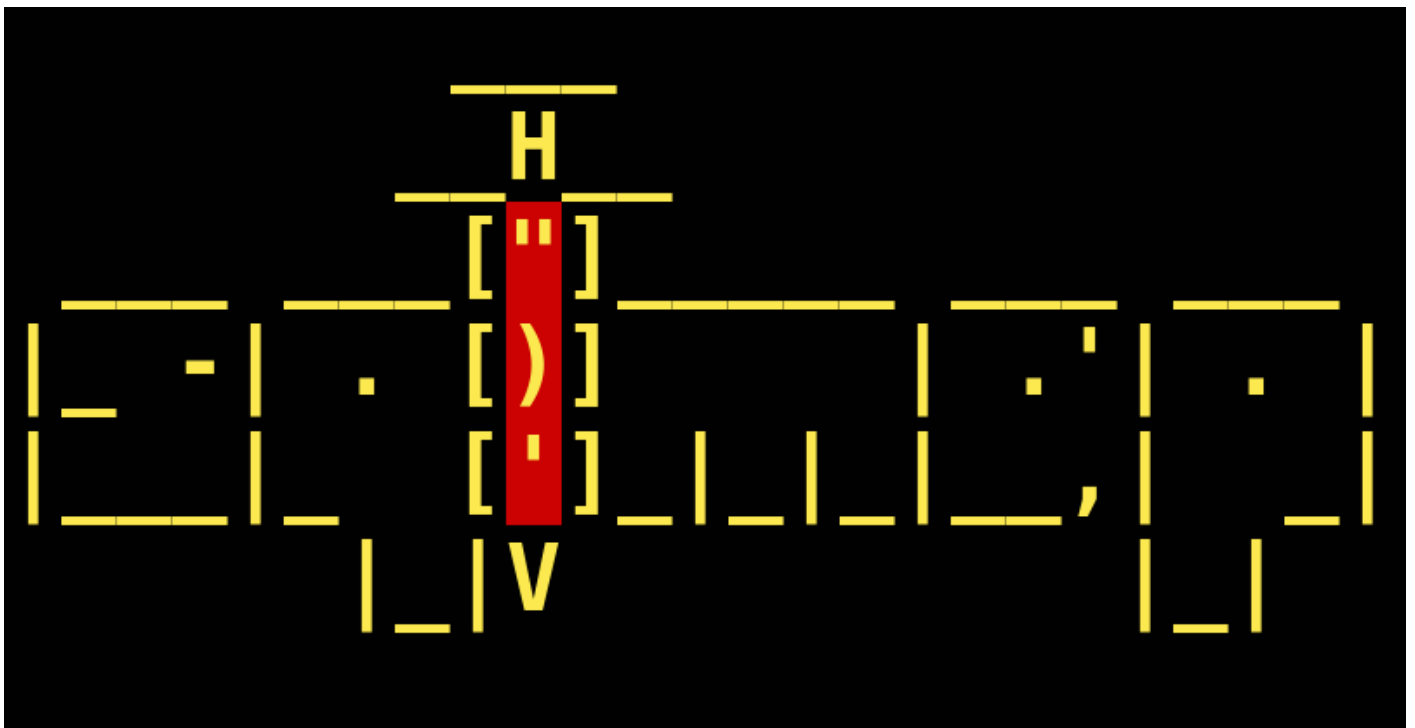


[Exploitation/Web] SQL injection

Introduction

Certainement l'une des injections web les plus connues, la **SQLi** permet de modifier une requête SQL initiale afin d'effectuer des actions non prévues par le concepteur de l'application.

Un attaquant pourrait exploiter cette vulnérabilité afin de contourner un système d'authentification ou récupérer le contenu d'une ou plusieurs tables de la base de données.



Exploitation manuelle

Bien que des outils comme **SQLmap** permettent d'exploiter automatiquement les failles SQL, il est toujours intéressant de savoir les exploiter manuellement pour plusieurs raisons :

- Comprendre le fonctionnement de ce que vous faites.

- Éviter le bombardement de requêtes sur la cible.
- Utiliser des fonctions non proposées par les outils automatisés.

Pour manipuler manuellement les requêtes, je vous recommande d'utiliser **BurpSuite**.

Tester l'injection

Les injections SQL sont possibles lorsqu'un paramètre modifiable par l'utilisateur est vulnérable car le développeur n'a pas préparé la requête.

Vous comprenez donc que l'injection SQL fonctionne donc aussi bien avec les paramètres utilisant la méthode **GET** que la méthode **POST**, seulement l'exploitation changera.

On peut tester la vulnérabilité du paramètre grâce au caractère ' qui devrait générer une erreur SQL qui s'affichera dans le cas où l'administrateur n'a pas désactivé le retour des erreurs :

```
http://monsitevulnérable.com/profil.php?id=id=1'
```

Contourner l'authentification

Si un formulaire d'authentification est vulnérable, il est possible d'usurper le compte d'un utilisateur ou de l'administrateur sans connaître son mot de passe grâce à l'injection suivante :

```
pass' OR 1=1 ;--
```

Afficher les champs d'une table

Il est aussi possible de dumper les tables.

Remarque : on sera limité au nombre de champs affiché initialement sur la page !

```
1 UNION SELECT 1,username,password FROM users ;--
```

Time based

Dans le cas où l'administrateur de l'application aurait désactivé l'affichage des erreurs SQL, nous ne pouvons pas afficher directement le résultat de nos requêtes SQL personnalisées sur la page.

Cependant, cela ne signifie en aucun cas que la vulnérabilité est inexploitable !

Puisqu'il n'y a aucun moyen d'avoir un retour visuel, on peut utiliser la fonction **SLEEP()** qui permet de mettre un délai et ainsi vérifier que la requête aboutie ou n'aboutie pas.

C'est assez rudimentaire mais efficace !

La fonction peut porter un autre nom selon le SGBD utilisé.

Par exemple, pour PostgreSQL il s'agit de **PG_SLEEP()**. Voici la syntaxe :

```
1;SELECT PG_SLEEP(2)--
```

SqlMap

Présentation

Cet outil permet d'automatiser les injections SQL et peut vous faire gagner beaucoup de temps.

Manuel

Voici la syntaxe globale :

```
sqlmap -u <URL> <OPTIONS>
```

Voici quelques options intéressantes :

Options	Descriptifs
--dump	Permet d'extraire les données sensibles.
--tables	Affiche les noms des tables dans la base de données.
--columns	Affiche les noms des colonnes d'une table.
--cookie=" "	Spécifie le cookie à utiliser pour l'authentification.
--technique=[U B]	Définit la technique d'injection à utiliser (U pour Union-Based et B pour Blind).
--delay=	Définit un temps d'attente entre les requêtes pour contourner certaines protections.
--data=" "	Spécifie les données POST.
--level=[1-5]	Définit le niveau de tests de pénétration (de 1 à 5, 5 étant le plus agressif).
--dbms	Spécifie le type de SGBD utilisé.

--os-shell	Ouvre un shell sur le système d'exploitation hôte.
--users	Lance une attaque brute force pour trouver les utilisateurs.
--passwords	Lance une attaque brute force pour trouver les mots de passe.

Revision #7
Created 20 October 2023 13:06:45 by Elieroc
Updated 3 May 2024 13:19:21 by Elieroc