# PENETRATION TESTING REPORT


Name: Narayan Ram Narayanan

Student ID:119398579, Email nana2011@umd.edu

Name: Dhanush Devaladakere Arvind

Student ID:119151556, Email: dagowda@umd.edu

Name: Sumanth Vankineni

Student ID: 119351130, Email: svankine@umd.edu

Name: Tanishq Javvaji

Student ID: 119070185, Email: tanishqj@umd.edu

# Table of Contents

## Executive Summary

This penetration test revealed several critical vulnerabilities in the target system, notably around weak default credentials, command injection, and improperly secured sensitive files. Key findings include successful command injection leading to a reverse shell, extraction of sensitive database credentials, and the discovery of multiple encrypted files.

## Major vulnerabilities identified:

**Weak Default Credentials:** The use of common credentials like 'admin/password' allowed unauthorized access to the Damn Vulnerable Web Application (DVWA) admin panel.

**Command Injection Vulnerabilities:** Enabled execution of arbitrary system commands and reverse shell establishment.

**Exposure of Sensitive Information:** Access to /etc/shadow and /etc/passwd files revealed user authentication details. Further, unsecured configuration files contained MySQL database credentials.

## Recommendations:

Implement strong, unique passwords and regularly update them.
Sanitize user inputs to mitigate command injection risks.
Secure sensitive files and employ encryption for critical data storage.

## Introduction

The penetration test was conducted to assess the security posture of the target system, comprising web applications and underlying server infrastructure. The test aimed to identify vulnerabilities, analyze potential impacts, and recommend mitigations. Our approach was comprehensive, encompassing both network and application-level assessments. The scope included identifying service misconfigurations, authentication weaknesses, and testing for common vulnerabilities like SQL injection, command injection, and exposure of sensitive data. Constraints included adhering to ethical hacking guidelines and avoiding any disruption to normal system operations.

## Methodology

Our penetration testing methodology followed a structured approach, utilizing both black-box and white-box testing techniques:

**Reconnaissance:** Gathered initial information using tools like Nmap to map the network and identify open ports and services.

**Service Enumeration:** Employed advanced scanning to determine service versions and configurations.

**Vulnerability Assessment:** Utilized tools like Gobuster for directory enumeration and manual code reviews to identify potential vulnerabilities.

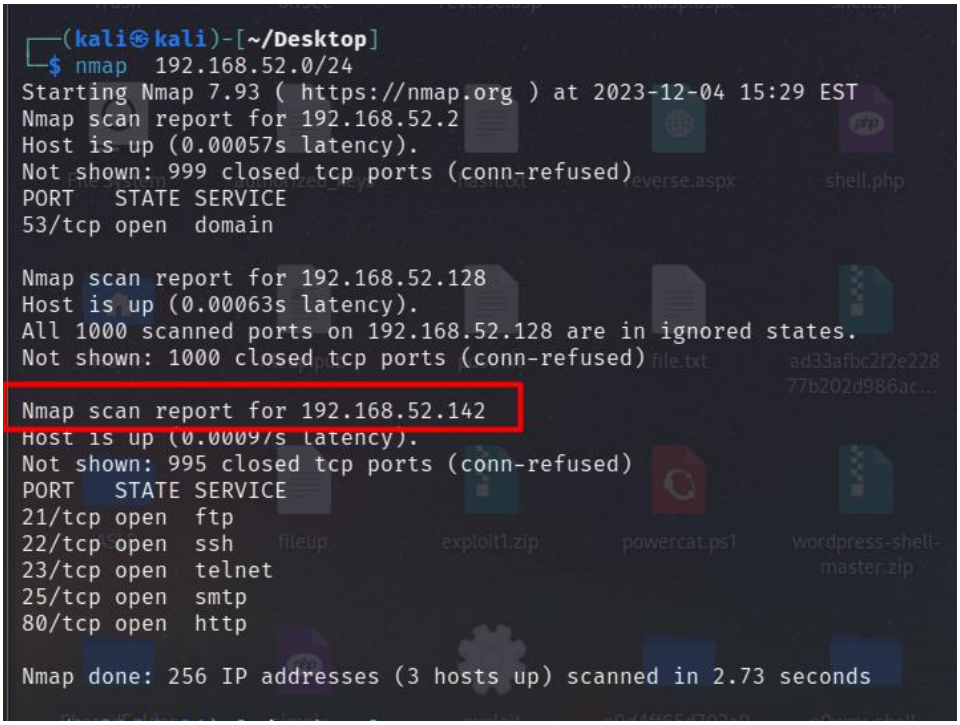
**Exploitation:** Tested vulnerabilities by attempting to exploit them, including command injection, and accessing sensitive files.

**Post-Exploitation**: After gaining higher privileges, conducted thorough system analysis to uncover further vulnerabilities and sensitive data.

# I. Initial Reconnaissance Phase

## 1. Initial Nmap Scan Results

During the reconnaissance phase of the penetration test, we employed an Nmap scan to map the network and identify active services. The scan was executed on a local IP range, specifically 192.168.52.0/24. This targeted scan was part of a broader batch process encompassing 256 IP addresses.




**Key Findings:**

**Host Identification**: The scan identified three responsive hosts within the specified IP range.

Service Enumeration for Host 192.168.52.142:

**Open Ports and Services:**

**Port 21 (FTP):** File Transfer Protocol service detected.

**Port 22 (SSH):** Secure Shell service detected.

**Port 23 (Telnet)**: Telnet service detected.

**Port 25 (SMTP)**: Simple Mail Transfer Protocol service detected.

**Port 80 (HTTP):** Hypertext Transfer Protocol service detected.

These open ports suggest active network services, offering potential vectors for further exploration.

**Closed Ports:**

The scan reported 995 TCP ports as closed, indicating a possible firewall or filtering mechanism. This response was characterized by connection refusals, hinting at a robust network defense strategy.

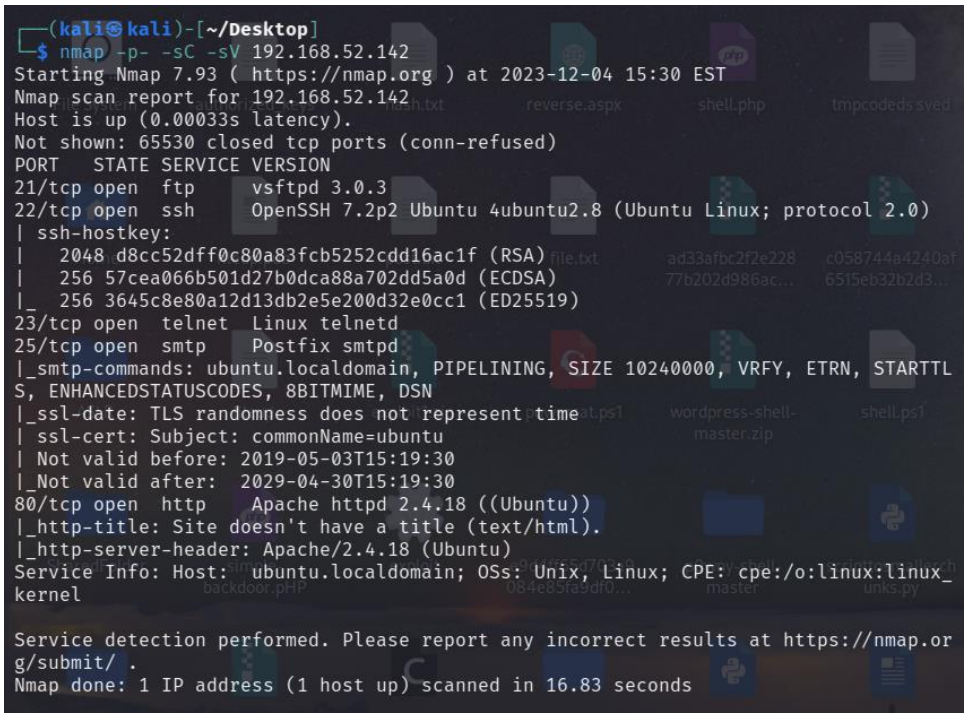
**Observations:**

The presence of multiple open ports, especially on standard service ports like FTP, SSH, Telnet, SMTP, and HTTP, points to a potentially diverse set of services and applications in use. This variety necessitates a comprehensive approach in subsequent testing phases, considering the variety of protocols and services identified.

# II. Service Enumeration and Version Identification

## 1. Advanced Nmap Scan Execution

Building upon the foundational data gathered in the initial reconnaissance phase, we advanced our penetration testing efforts by conducting a more detailed Nmap scan. This scan incorporated the use of script scanning (-sC) and service version detection (-sV) options, aimed at obtaining an in-depth understanding of the services running on the target host at IP address 192.168.52.142.




**Key Findings:**

**Enhanced Service Details:**

**FTP Service (Port 21):**

Service Version: vsftpd 3.0.3.

**SSH Service (Port 22):**

Service Version: OpenSSH 7.2p2.

Operating System: Ubuntu.

SSH Host Keys: Detailed in the scan output.

**SMTP Service (Port 25):**

Service Version: Postfix, with extended capabilities.

**HTTP Service (Port 80):**

Service Version: Apache httpd 2.4.18.

Operating System: Ubuntu.

HTML Title: Not retrieved, indicating potential content restrictions or specific server configurations

**Observations:**

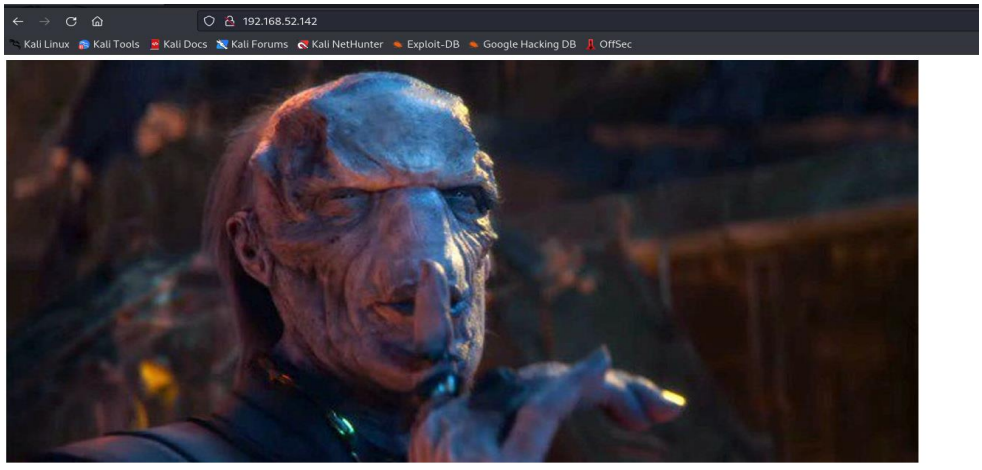
The detailed scan results provide valuable insights into the versions of the services and underlying operating systems, crucial for identifying potential vulnerabilities. The specific versions of vsftpd, OpenSSH, Postfix, and Apache httpd, coupled with their respective configurations, lay the groundwork for targeted vulnerability assessment. The absence of an HTML title for the HTTP service and the presence of SSL certificates suggest a conscious effort towards security, warranting a careful approach in further testing stages.

# III. Web Service Analysis

**Web Content Assessment on HTTP Service**

## 1. Exploration of HTTP Service (Port 80)

Following the identification of an operational HTTP service on the target IP address 192.168.52.142, our testing protocol involved directly accessing the webpage hosted on the server. This step was crucial for understanding the nature and purpose of the web content being served.

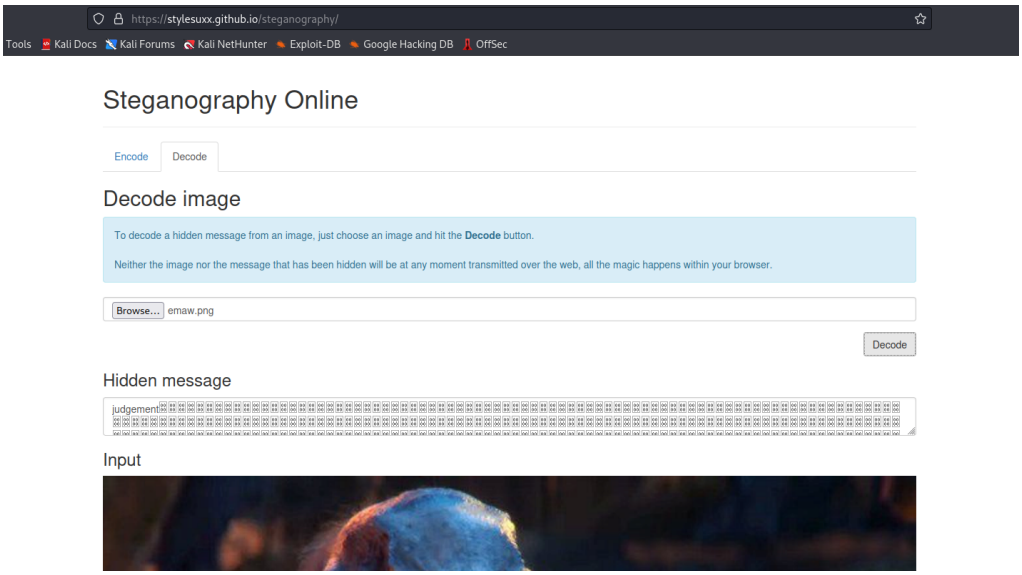


**Key Observations:**

**Steganographic Analysis, Hidden Message Investigation**

## 2. Analysis of Webpage Image

We delved into the possibility of steganography upon discovering a lone image on the website hosted at IP address 192.168.52.142. Using the online tool 'stylesuxx.github.io/steganography', we aimed to decode any concealed messages within the image.

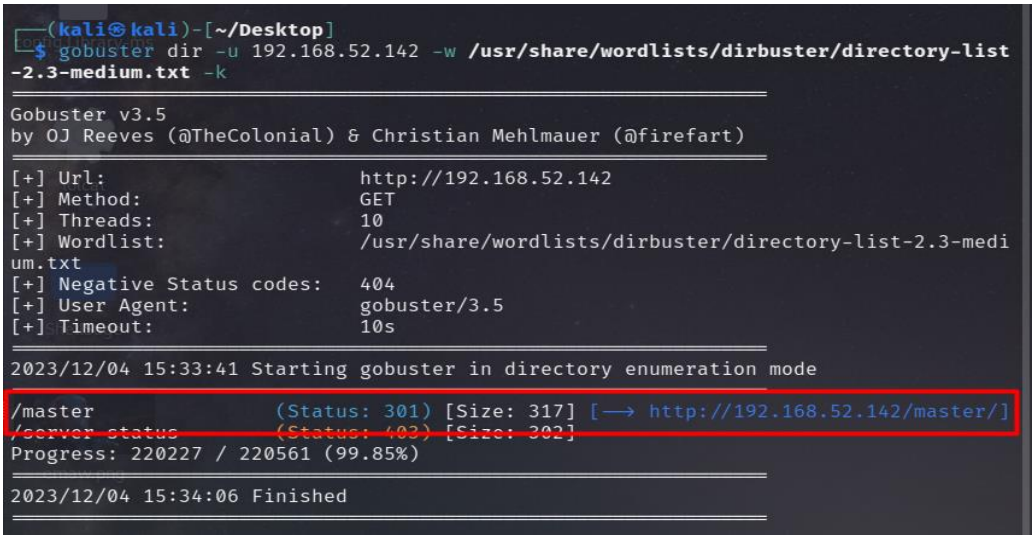



**Key Outcomes:**

**Discovery of Concealed String:** The tool revealed a hidden string "judgement" within the image.

**Potential Significance:** This string may act as a passphrase, password, or key, indicating layered security measures or hidden functionalities within the system.

**Implications**: The presence of a steganographically hidden message suggests advanced security features, requiring meticulous penetration testing to uncover and assess further vulnerabilities.



# IV. Directory Enumeration

## 1. Execution of Gobuster Scan (Gobuster Directory Search)

After discovering the string "judgement" within the website's image, we focused on directory enumeration for IP address 192.168.52.142. This was conducted using Gobuster version 3.5, a tool renowned for its efficiency in web content discovery.

**Procedure and Findings**:

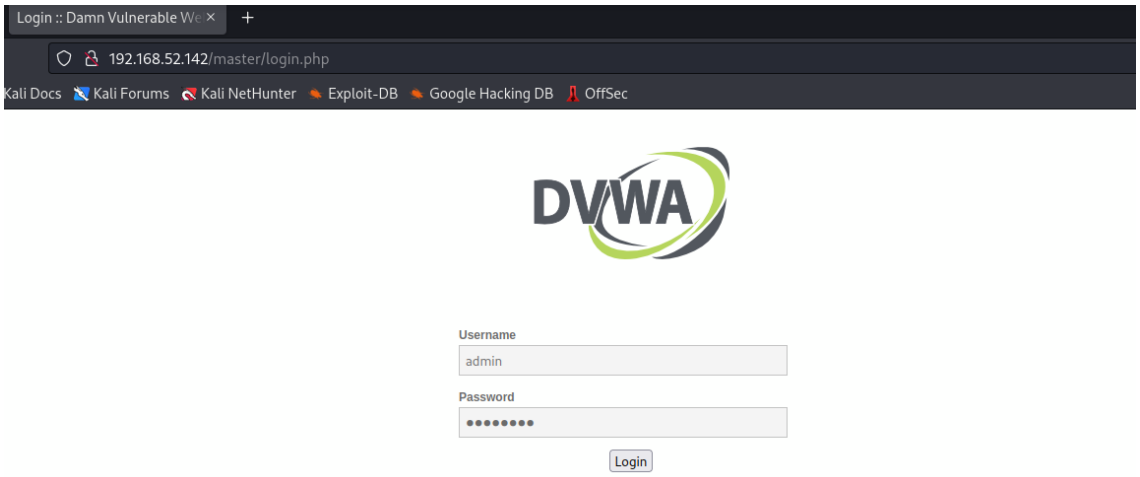
**Gobuster Command:** We utilized the 'directory-list-2.3-medium.txt' wordlist for a comprehensive directory search.

**Significant Discovery:** The scan identified a directory named '/master/'.

## 2.Attempted Authentication on '/master/' Directory

**Accessing the '/master/' Directory Login Page**

Post the enumeration of the '/master/' directory, we progressed to examining the login mechanism. This phase is critical in assessing the robustness of authentication protocols used by the target system.




**Key Actions and Results:**

**Credential Testing:**

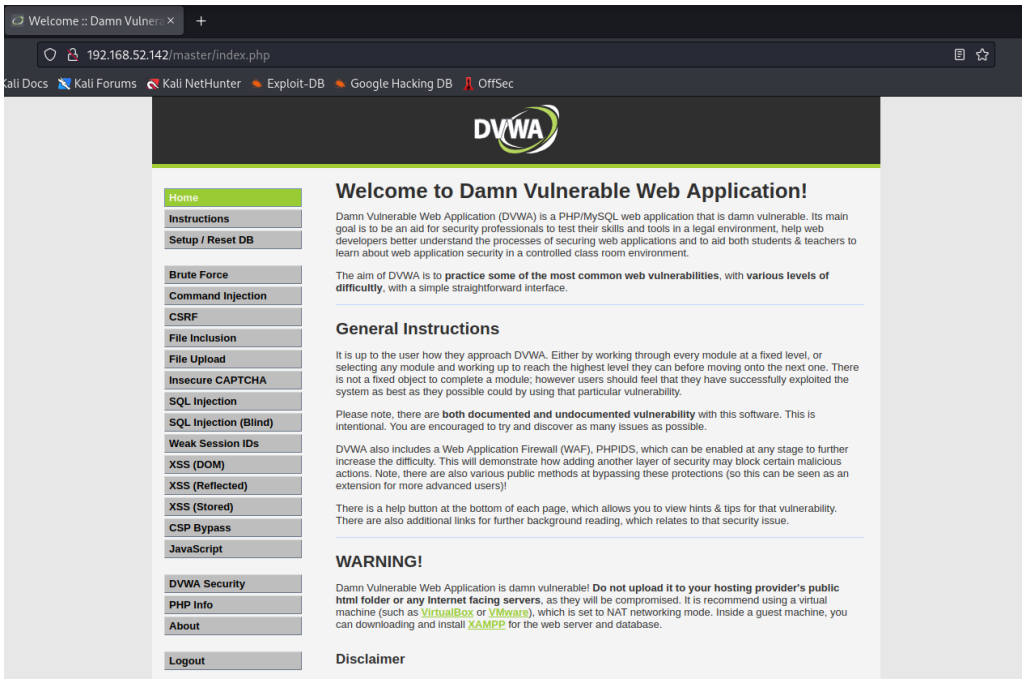
We initiated our testing with a common approach, using the widely recognized default credentials: username 'admin' and password 'password'.

This choice is based on the frequent observation that many systems retain default credentials, a notable security oversight.

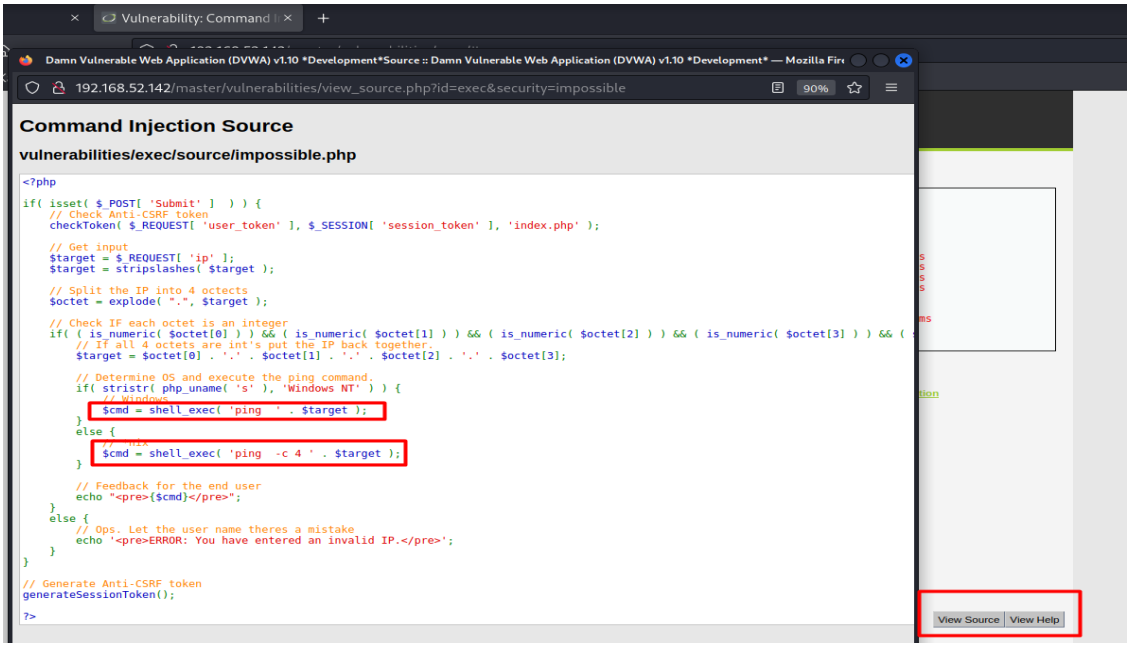
**Successful Access:**

Surprisingly, this basic credential combination proved effective.

We gained access to the Damn Vulnerable Web Application (DVWA) admin panel, a significant breakthrough in the penetration test.




The image illustrates the homepage of the Damn Vulnerable Web Application (DVWA) after logging in.



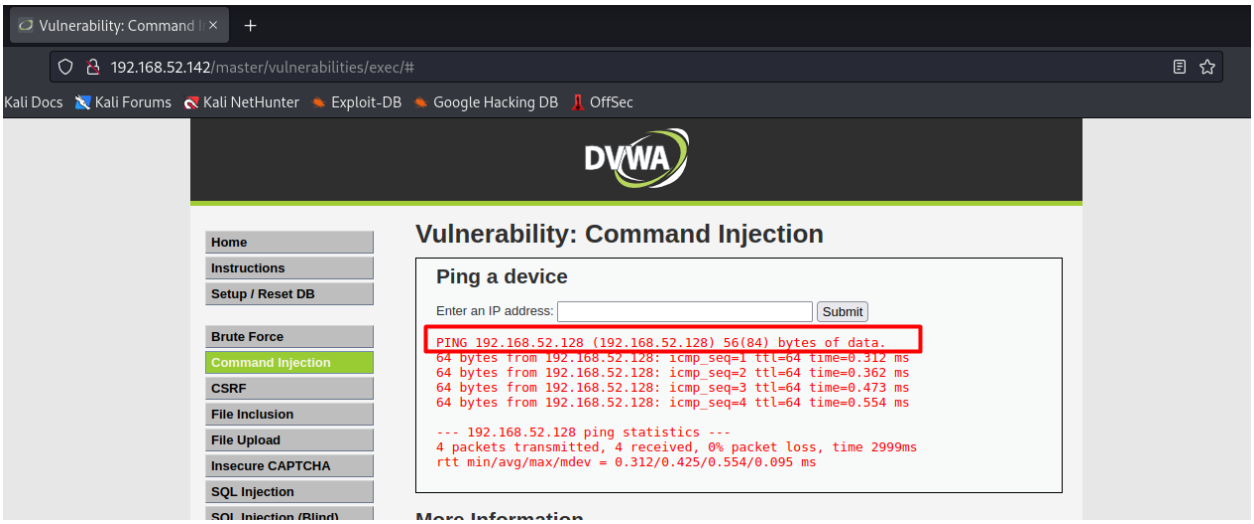
# V. Code Analysis and Vulnerability Identification

## 1.Examination of DVWA Command Injection Pa

The source code revealed the implementation of the shell_exec function. This function is commonly used to execute system commands within PHP scripts.

Its presence is a focal point for assessing command injection vulnerabilities.

**Command Execution Mechanism:**




The source code highlighted the use of the ping command.

The command incorporates a variable $target, which is indicative of accepting user input.

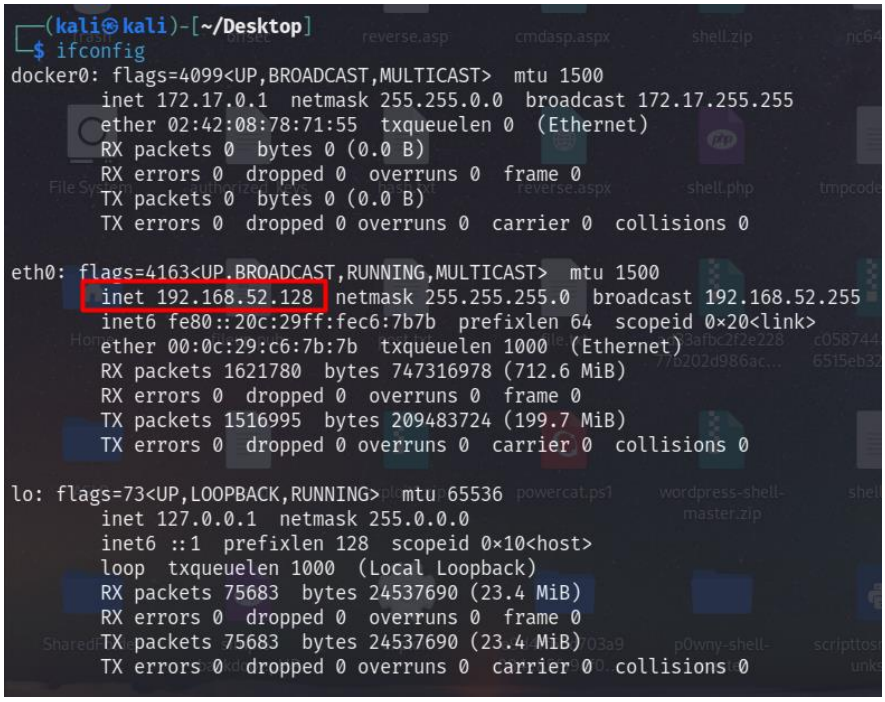
**Implications:**

The combination of the shell_exec function and user-input integration in the ping command raises significant security concerns. This setup suggests a potential vulnerability to command injection attacks, where an attacker could manipulate the $target variable to execute arbitrary system commands. Such vulnerabilities are critical and require immediate attention to mitigate risks of unauthorized system access or control.

## 2.Network Connectivity Test

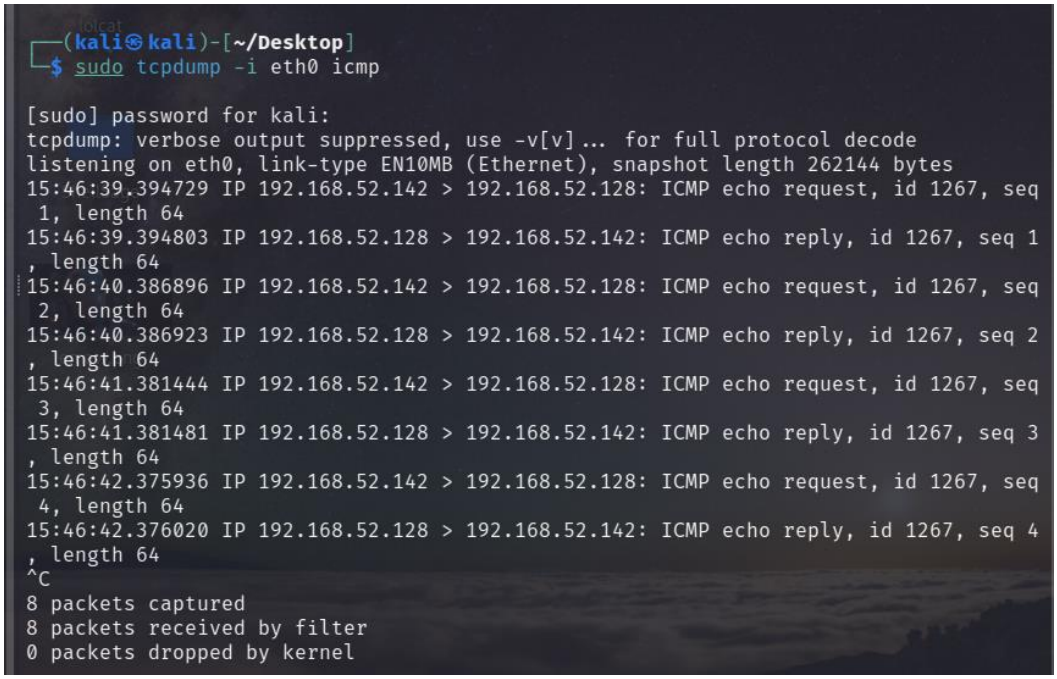
**Command Injection and ICMP Echo Analysis**

### a. Command Injection Page Test:




Conducted a ping test to local IP (192.168.52.128) from DVWA host (192.168.52.142).

Confirmed successful transmission and reception of four ICMP echo requests.

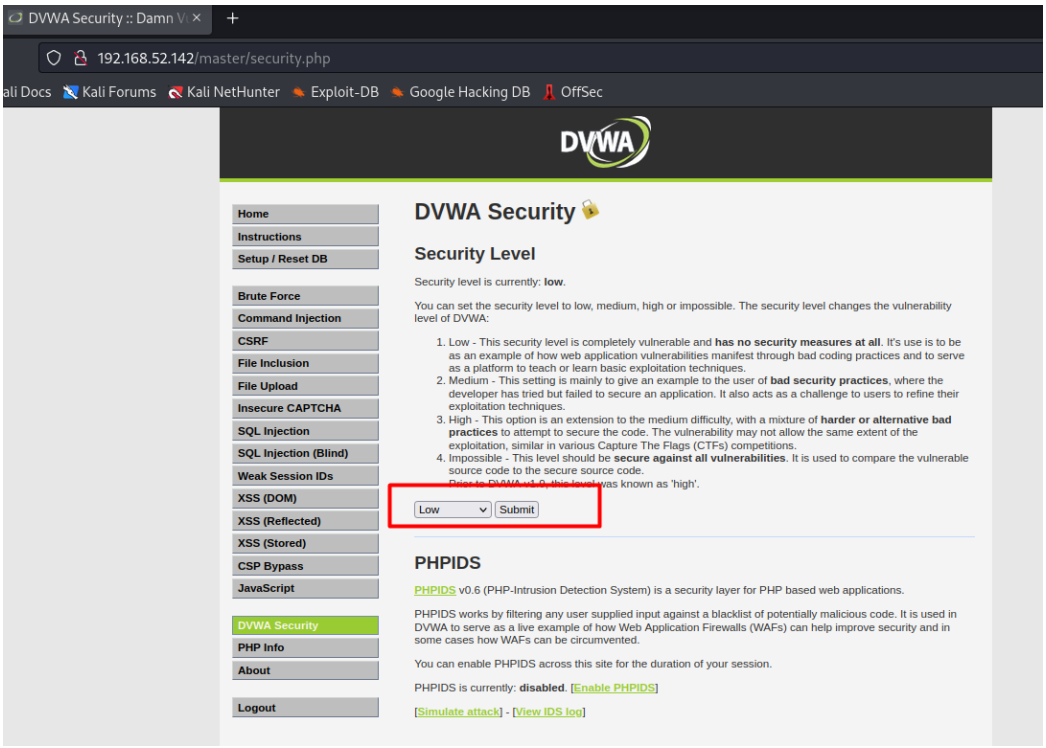


## b. ICMP Packet Exchange Observation:

Noted consistent ICMP echo requests and replies between .142 and .128, with no packet loss.

**Implications:**

Verified application's standard command response, indicating network connectivity.




# 3. Security Level Modification and Vulnerability Exploitation

## a. Security Level Change(DVWA Security Level Adjustment):

Modified DVWA Security Level from 'Impossible' to 'Low'.

The 'Low' setting implies minimal security measures, exposing the application to vulnerabilities.

J. Source Code Analysis at Low Security Level

## b. PHP Source Code Review:


Examined the Command Injection exercise source code.

Identified use of shell_exec for Unix ping command execution.

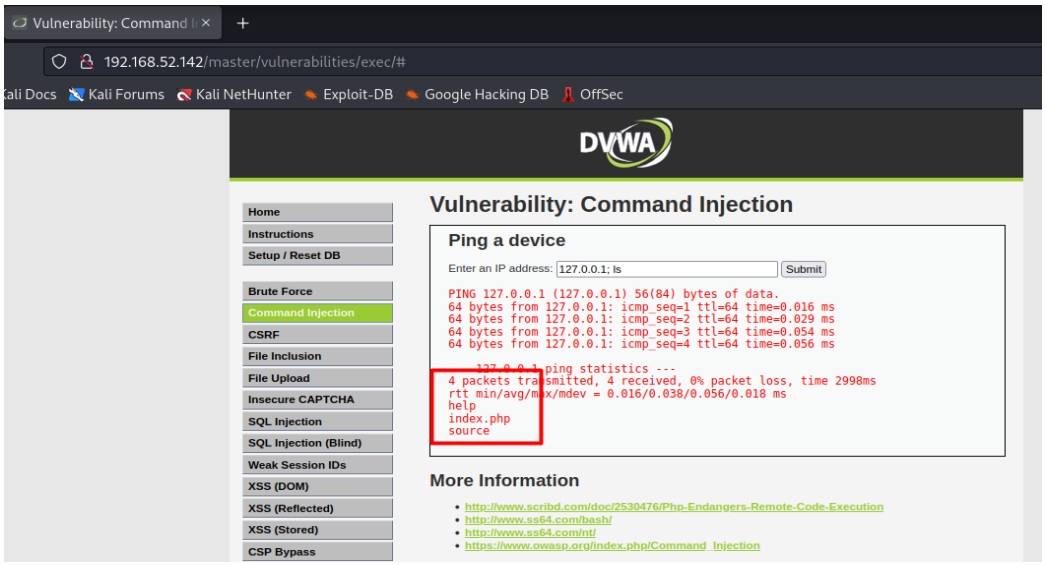
K. Command Injection Test

## c. Command Injection Experimentation:

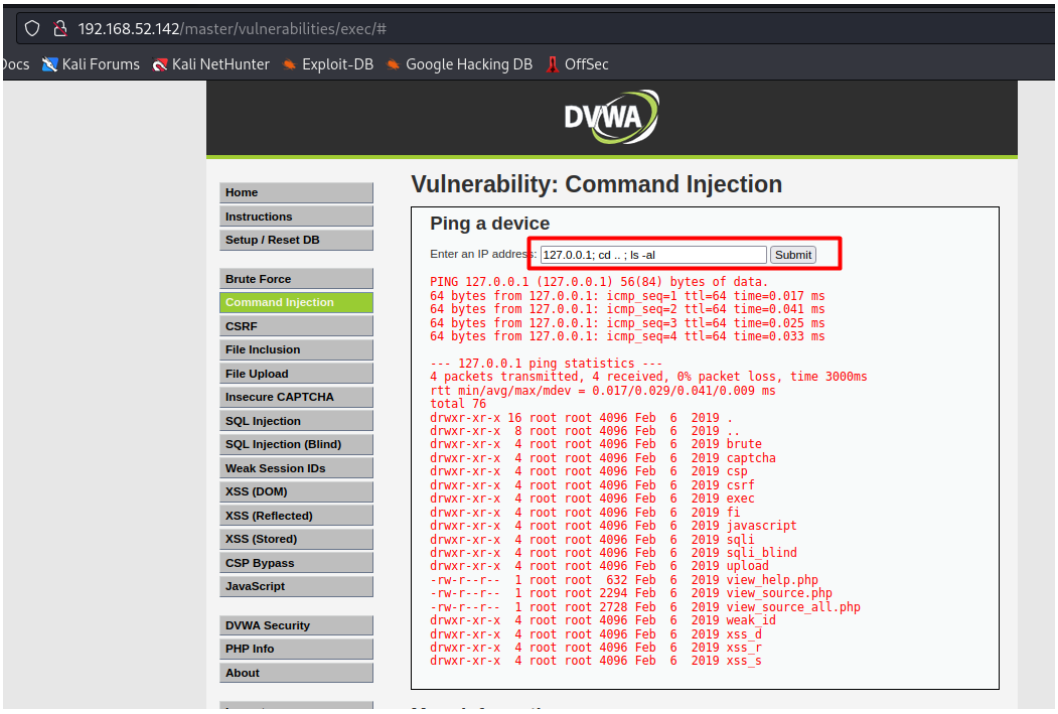
Performed a command injection with ls command.

Unexpected output, including directory listings, confirmed successful execution.

L. Advanced Command Injection and Directory Listing



## d. Complex Payload Execution:




Injected a payload combining loopback IP with directory change (cd) and listing (ls -al) commands.

Successfully listed contents of the parent directory, revealing sensitive files and directories.

# VI.Reverse Shell Connection and Listener Setup

## 1. Reverse Shell Execution Attempt

**Reverse Shell Command Construction:**

Utilized a complex shell command for reverse shell establishment:

mkfifo /tmp/s: Created a named pipe for bidirectional communication.

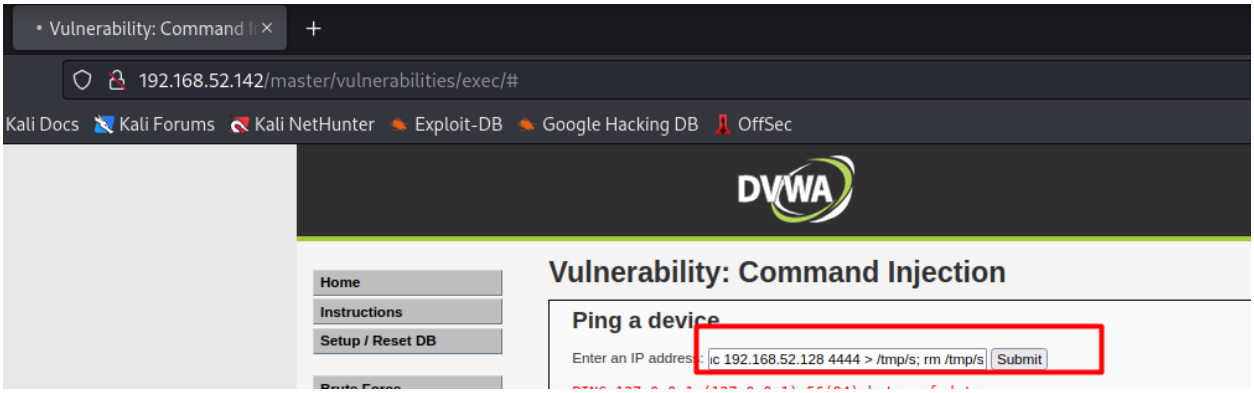
/bin/sh -i: Requested an interactive shell session.

I/O Redirection: Managed input/output via the named pipe.

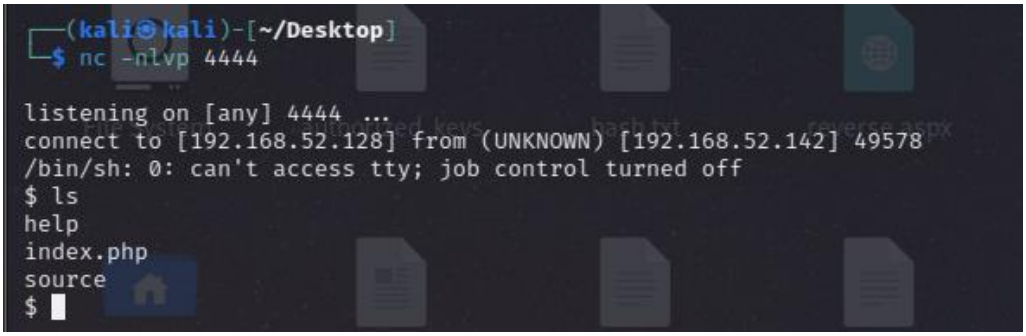
Netcat (nc) Usage: Attempted a TCP connection to our machine on port 4444 for shell control.

Cleanup Step: Scheduled removal of the named pipe post-session.

N. Local Listener Configuration




## 2. Local Machine Listener Initialization:




Set up a listener on our local machine with nc -nlvp 4444 to capture the reverse shell.

Configured Netcat to listen verbosely on port 4444 for real-time connection feedback.

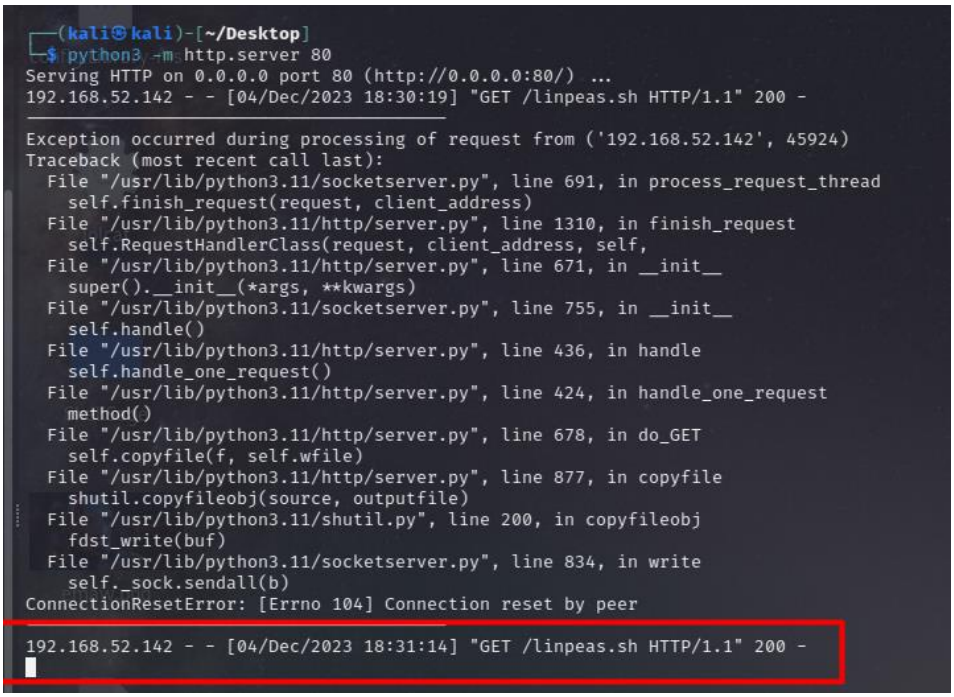
### 3. Confirmation of Successful Connection:

Observed a successful reverse shell connection from the target server (192.168.52.142).

Executed a test command ls, confirming operational control with file listings.

## VII. File Transfer via Reverse Shell

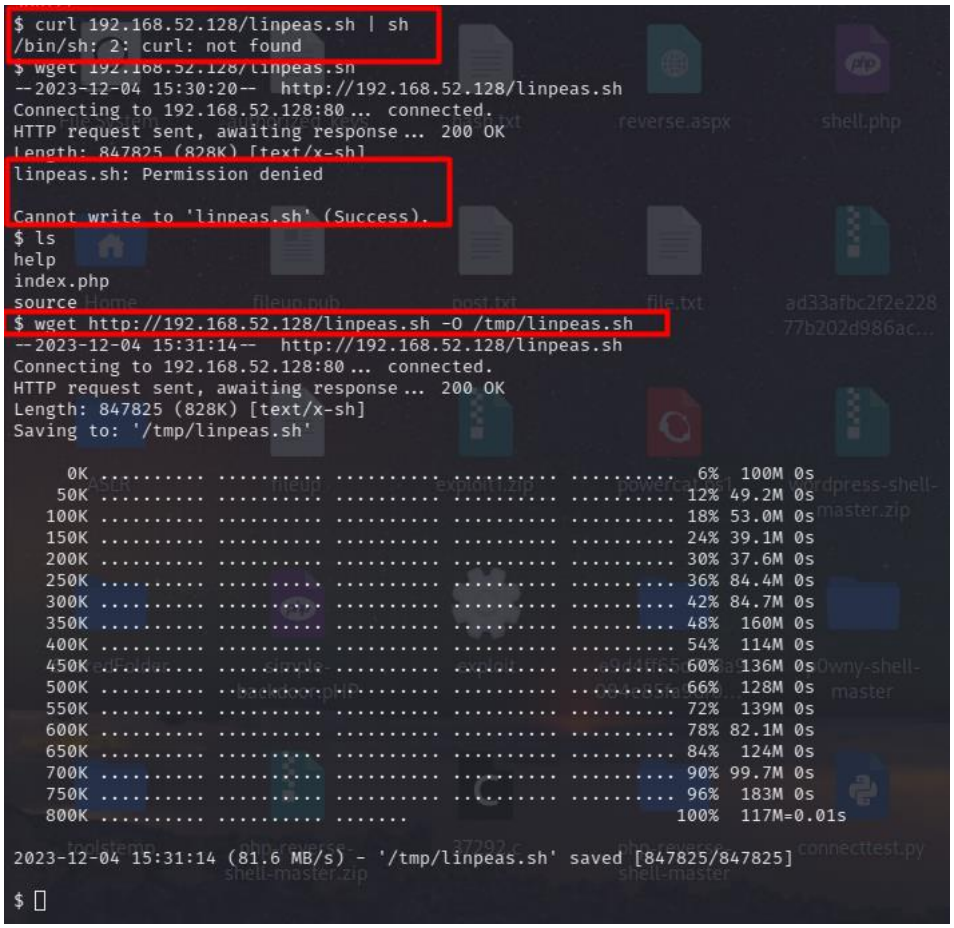
### 1. Local HTTP Server Initiation:




Started a Python HTTP server using python3 -m http.server 80.

Made the local directory accessible over HTTP on port 80 for file transfers.

## 2. Transferring linpeas.sh Script:




Within the reverse shell, executed wget to transfer linpeas.sh from Kali to the target system.

Attempt to use curl failed due to its absence on the target system.

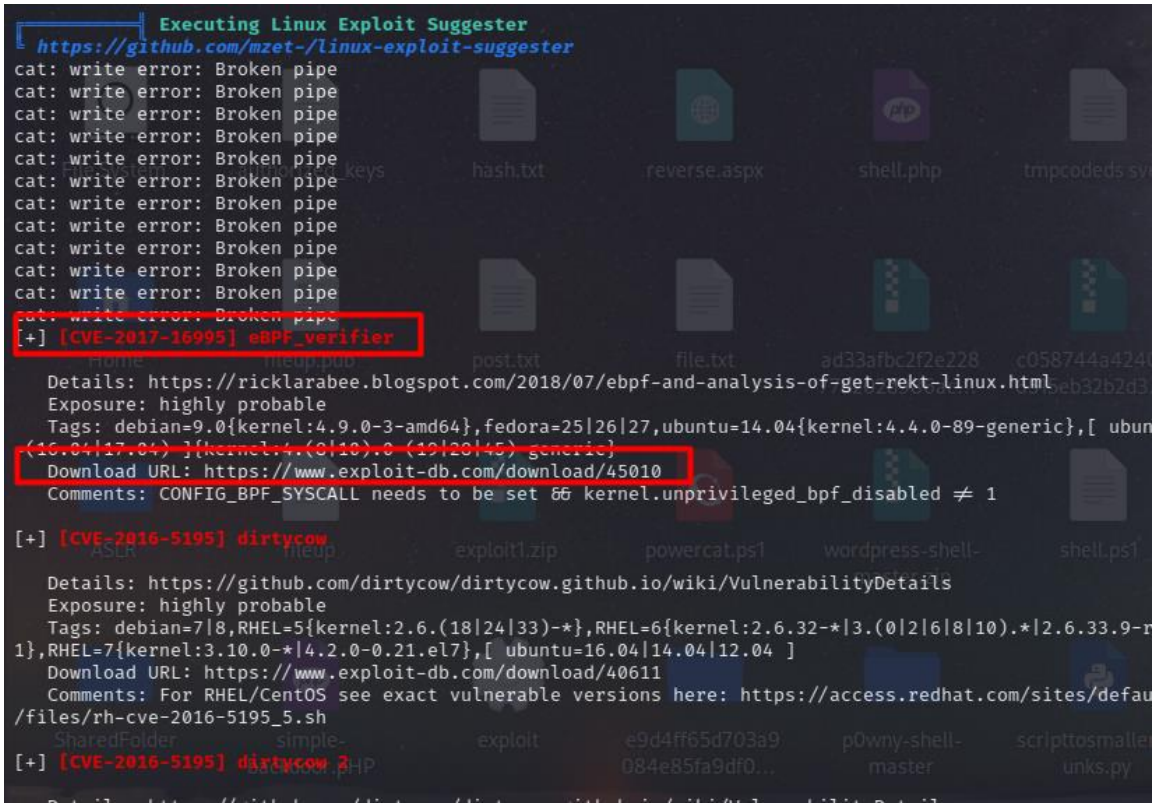
## 3. Confirmation of Script Transfer:

Screenshots show successful retrieval of linpeas.sh using wget.

Script saved in the /tmp directory of the target system.

# VIII. Vulnerability Assessment and Exploitation

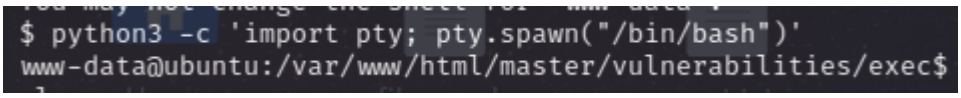
## 1. Execution of linpeas.sh Script:




Utilized linpeas.sh for comprehensive vulnerability scanning.

Identified key vulnerabilities including CVE-2017-16995 (eBPF) and CVE-2016-5195 (Dirty COW).

## 2. Enhanced Shell Access:




Executed python3 -c 'import pty; pty.spawn("/bin/bash")' for a stable, interactive shell.

Facilitated precise execution of targeted exploits.

### 3. Exploiting CVE-2017-16995:

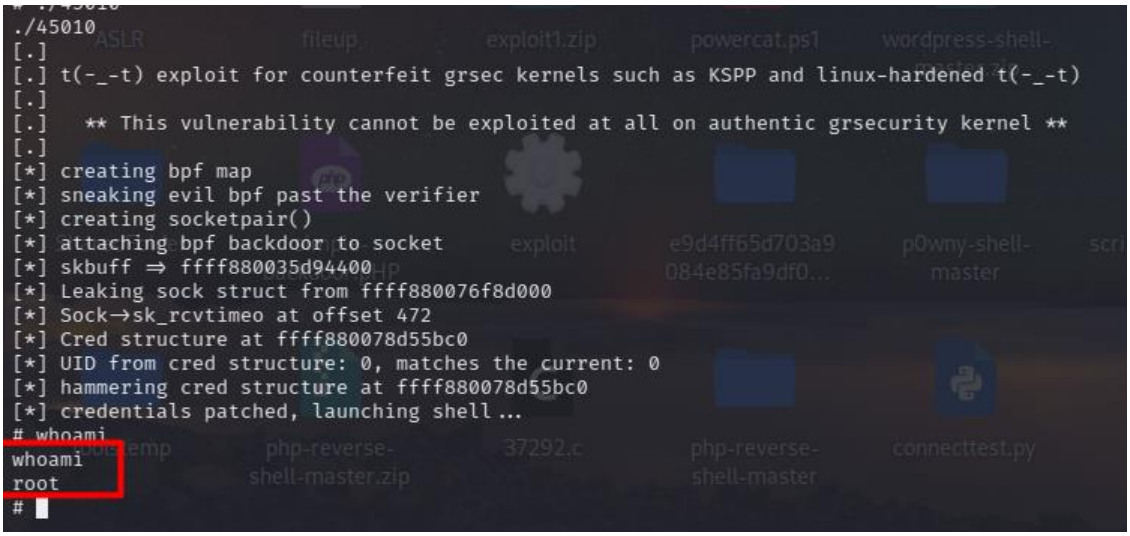
Leveraged the stable shell to initiate exploitation of the eBPF vulnerability (CVE-2017-16995).

### 4. Downloading and Executing Exploit:

Acquired relevant exploit script from Exploit Database (exploitdb).

Executed the exploit on the target system.

### 5. Confirmation of Privilege Escalation:




Post-exploit execution, used whoami command.

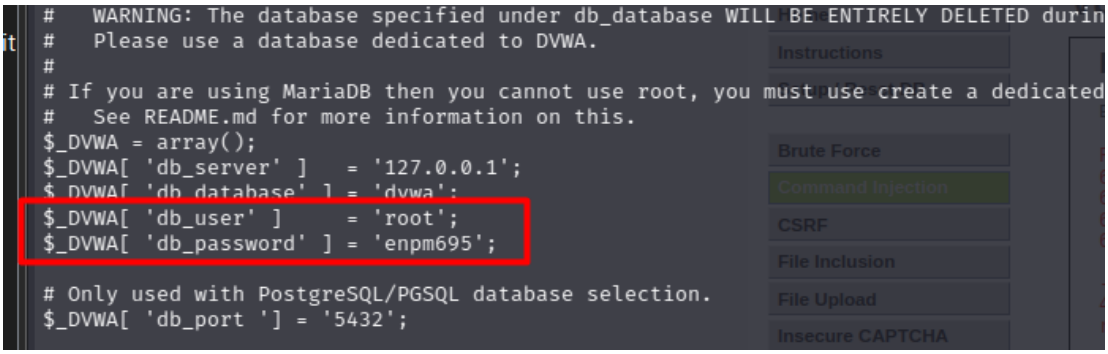
Confirmed successful elevation to root user.

## IX. Post-Privilege Escalation Activities

**Sensitive File Discovery and Database Credential Extraction**

### 1. Configuration File Examination:

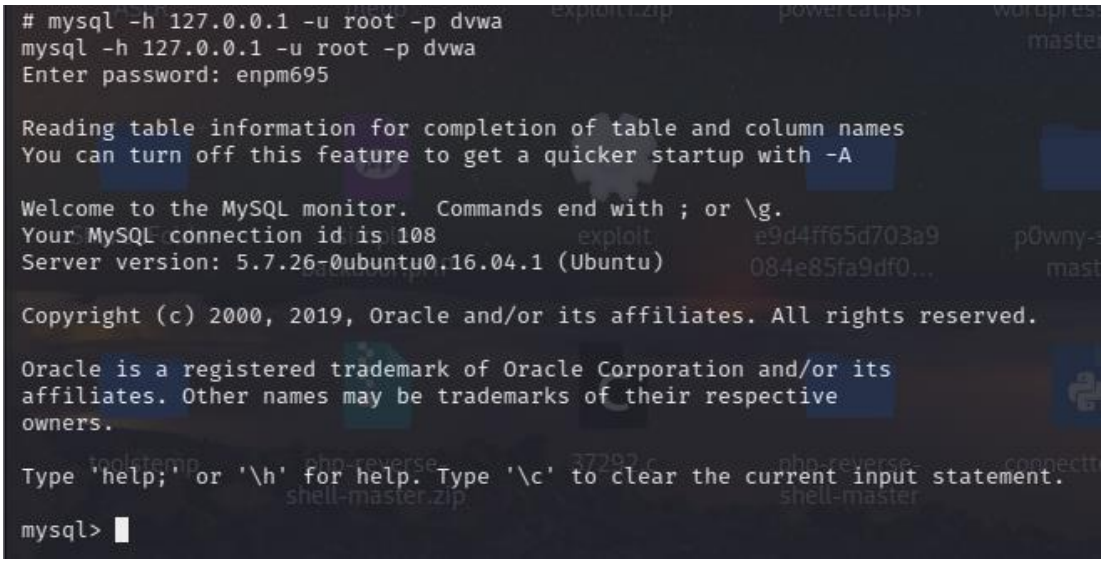
```
$ pwd
/var/www/html/master/config
$ ls
config.inc.php
config.inc.php.dist
$ cat config.inc.php
```




Post privilege escalation, identified a configuration file in the working directory.

Discovered MySQL database credentials within the file.

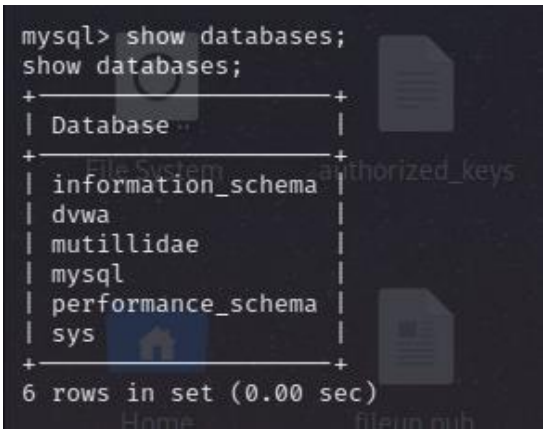
## 2. MySQL Server Access:




Accessed MySQL using mysql -h 127.0.0.1 -u root -p.

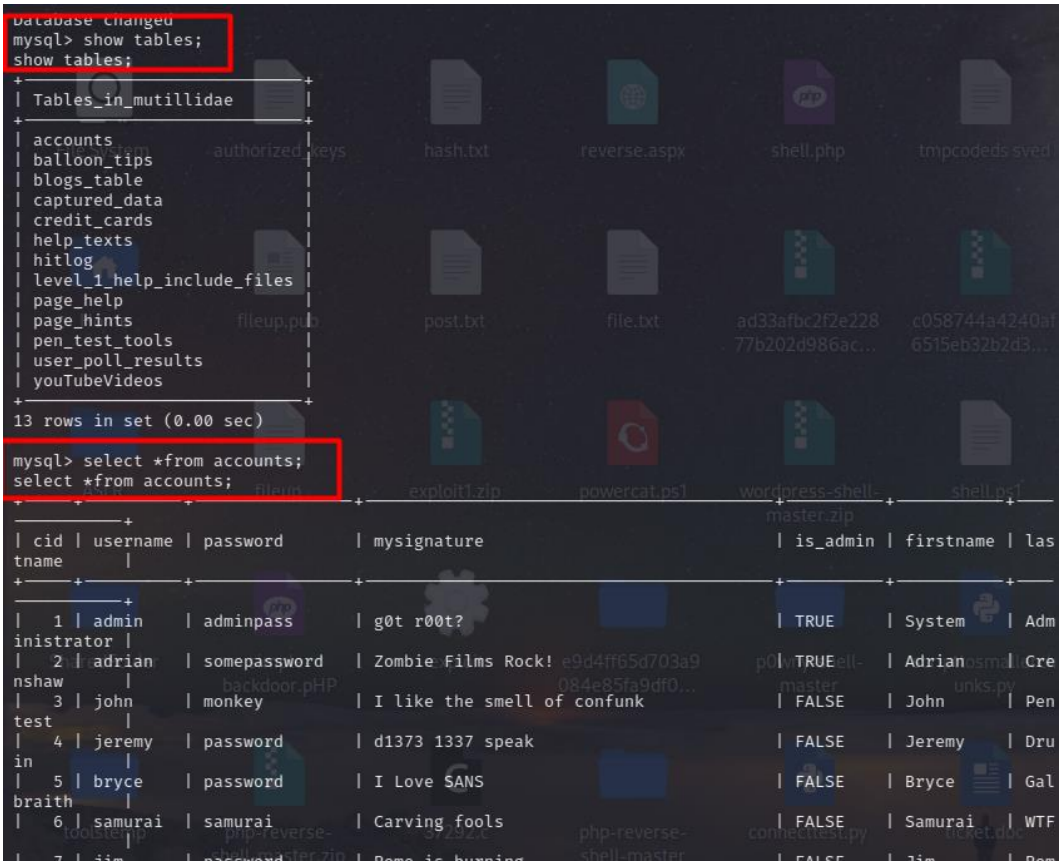
Leveraged credentials from the configuration file for login.

## 3. Interaction with MySQL Database:




Established a direct interface for database queries.
Opportunity to investigate data within the MySQL environment.

## 4. Database Enumeration Command:

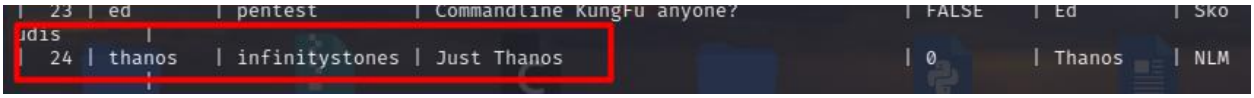



Executed SHOW DATABASES; in MySQL.
Identified databases: information_schema, dvwa, mutillidae, mysql, performance_schema, sys.

### 5. Accounts Table Analysis in Mutillidae Database:

Located an "accounts" table in the mutillidae database.

Executed SELECT * FROM accounts; to reveal usernames and passwords.



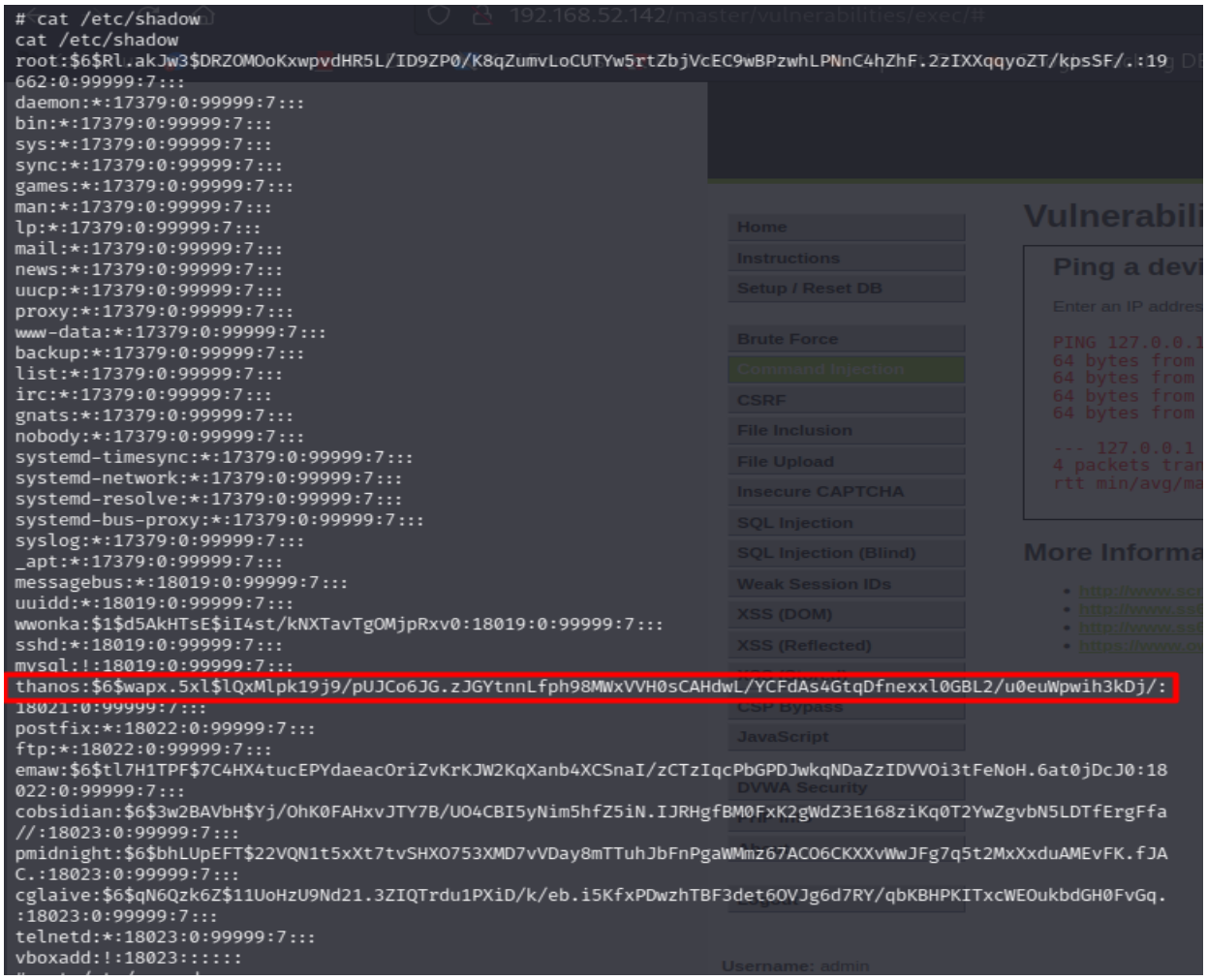

The identification of a username "thanos" with the associated password "infinity stones" within the compromised credentials seems particularly interesting.

## X. User Authentication Data Analysis and System Access

### 1. Accessing /etc/shadow (Examination of /etc/shadow File):

Retrieved user authentication data from /etc/shadow.

Found hashed passwords for users: "thanos," "cglaive," "emaw," "pmidnight," "cobsidian," and "wwonka."

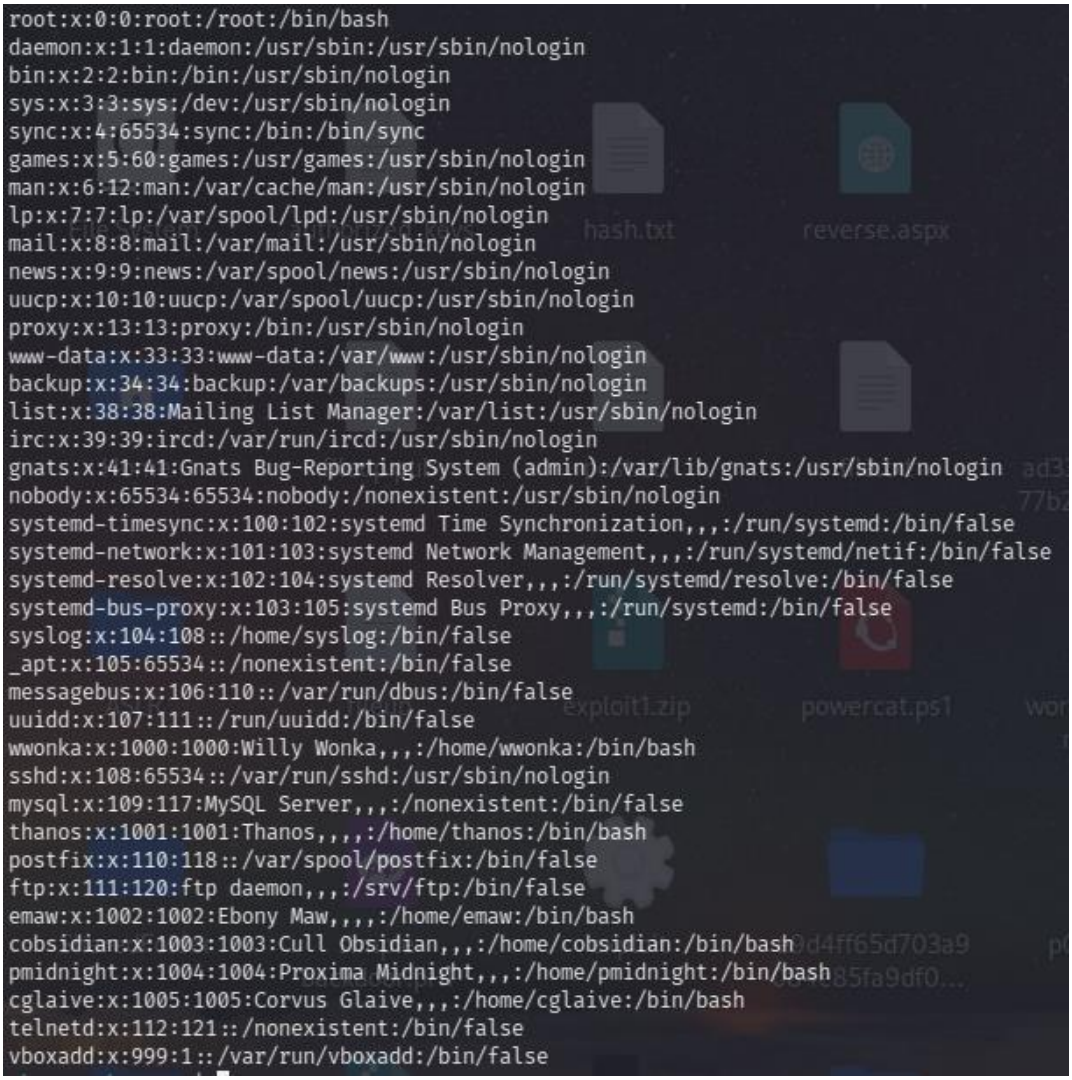


## 2. Exploring /etc/passwd:

Accessed /etc/passwd to obtain user account details.

Identified users, user IDs, group IDs, home directories, and shell information.

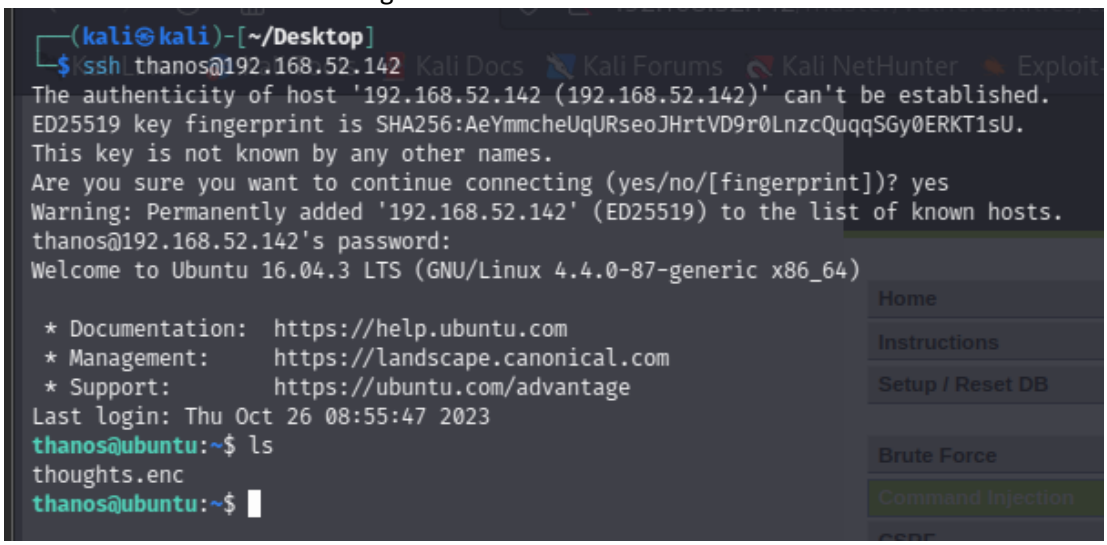
Noted a user "Ebony Maw" with username "emaw".



## 3. System Access with Thanos Account:

Logged in with "thanos" and "infinitystones".

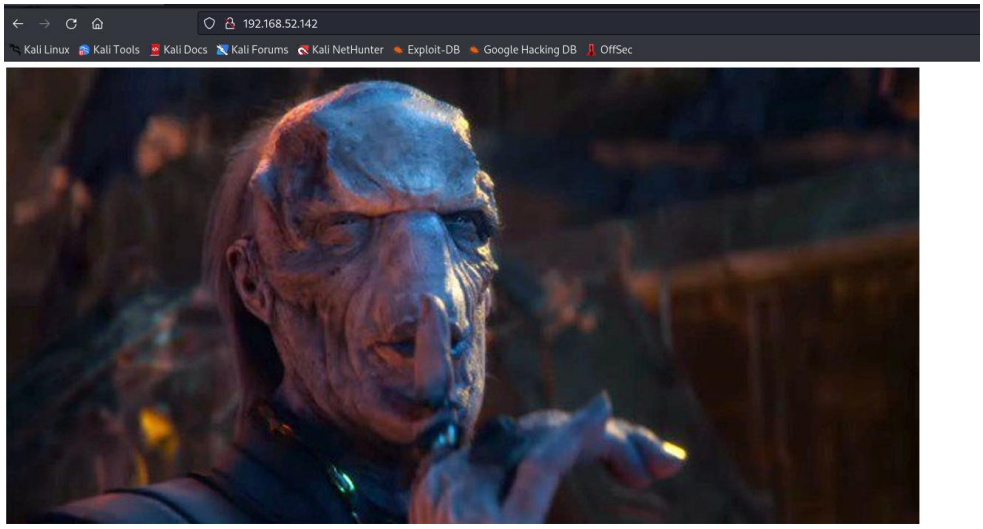
Used ls command to reveal a file thoughts.enc.

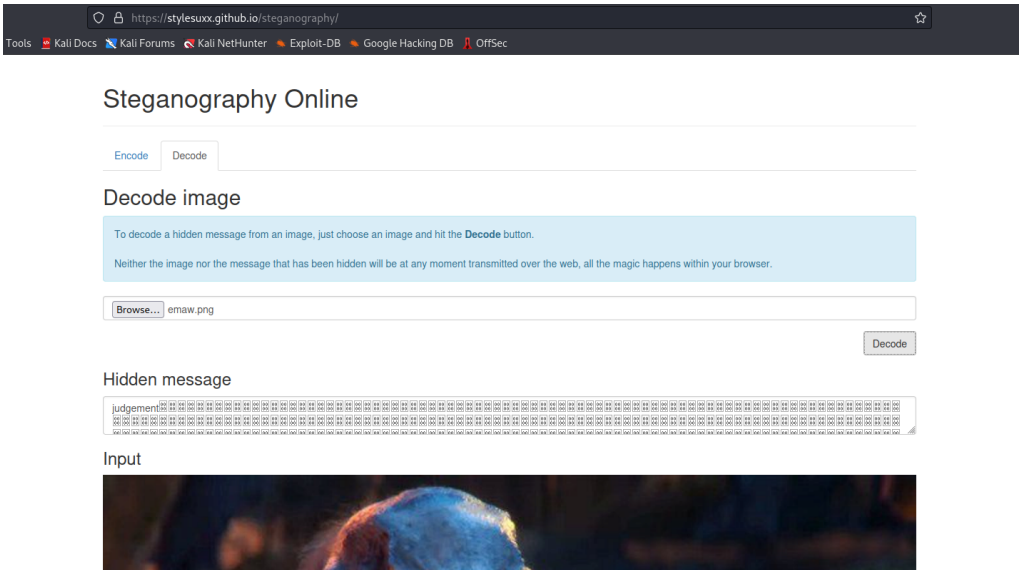


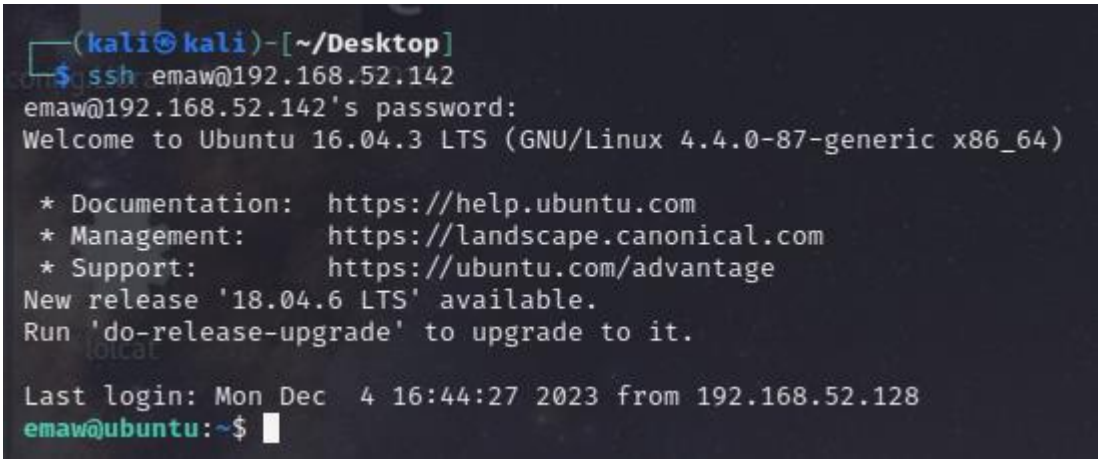

## 4. SSH Access Using emaw Account:

Attempted SSH with "emaw" username and "judgement" password.

Successfully gained access for further system exploration.






## 5. Cracking the password for other users

Initial Hash Cracking Attempt: Utilized the default rockyou.txt wordlist for cracking hashes extracted from the /etc/shadow file, yielding no significant results.

```
root@ubuntu:/# cd /etc/
root@ubuntu:/etc# cat shadow
root:$6$Rl.akJw3$DRZOMOoKxwpvdHR5L/ID9ZP0/K8qZumvLoCUTYw5rtZbjVcEC9wBPzwhLPNnC4hZhF.2zIXXqqyoZT/kpsSF/.:19662:0:99999:7:::
daemon:*:17379:0:99999:7:::
bin:*:17379:0:99999:7:::
sys:*:17379:0:99999:7:::
sync:*:17379:0:99999:7:::
games:*:17379:0:99999:7:::
man:*:17379:0:99999:7:::
lp:*:17379:0:99999:7:::
mail:*:17379:0:99999:7:::
news:*:17379:0:99999:7:::
uucp:*:17379:0:99999:7:::
proxy:*:17379:0:99999:7:::
www-data:*:17379:0:99999:7:::
backup:*:17379:0:99999:7:::
list:*:17379:0:99999:7:::
irc:*:17379:0:99999:7:::
gnats:*:17379:0:99999:7:::
nobody:*:17379:0:99999:7:::
systemd-timesync:*:17379:0:99999:7:::
systemd-network:*:17379:0:99999:7:::
systemd-resolve:*:17379:0:99999:7:::
systemd-bus-proxy:*:17379:0:99999:7:::
syslog:*:17379:0:99999:7:::
_apt:*:17379:0:99999:7:::
messagebus:*:18019:0:99999:7:::
uuidd:*:18019:0:99999:7:::
wwonka:$1$d5AkHTsE$iI4st/kNXTavTgOMjpRxv0:18019:0:99999:7:::
sshd:*:18019:0:99999:7:::
mysql:!:18019:0:99999:7:::
thanos:$6$wapx.5xl$lQxMlpk19j9/pUJCo6JG.zJGYtnnLfph98MWxVVH0sCAHdwL/YCFdAs4GtqDfnexxl0GBL2/u0euWpwih3kDj/:18021:0:99999:7:::
postfix:*:18022:0:99999:7:::
ftp:*:18022:0:99999:7:::
emaw:$6$tl7H1TPF$7C4HX4tucEPYdaeacOriZvKrKJW2KqXanb4XCSnaI/zCTzIqcPbGPDJwkqNDaZzIDVVOi3tFeNoH.6at0jDcJ0:18022:0:99999:7:::
cobsidian:$6$3w2BAVbH$Yj/OhK0FAHxvJTY7B/UO4CBI5yNim5hfZ5iN.IJRHgfBM0FxK2gWdZ3E168ziKq0T2YwZgvbN5LDTfErgFfa//:18023:0:99999:7:::
pmidnight:$6$bhLUpEFT$22VQN1t5xXt7tvSHXO753XMD7vVDay8mTTuhJbFnPgaWMmz67ACO6CKXXvWwJFg7q5t2MxXxduAMEvFK.fJAC.:18023:0:99999:7:::
cglaive:$6$qN6Qzk6Z$11UoHzU9Nd21.3ZIQTrdu1PXiD/k/eb.i5KfxPDwzhTBF3det6OVJg6d7RY/qbKBHPKITxcWEOukbdGH0FvGq.:18023:0:99999:7:::
telnetd:*:18023:0:99999:7:::
vboxadd:!:18023::::::
root@ubuntu:/etc#
```

hash

home > kali > sec_os_final > hash

```
1 $6$tl7H1TPF$7C4HX4tucEPYdaeacOriZvKrKJW2KqXanb4XCSnaI/zCTzIqcPbGPDJwkqNDaZzIDVVOi3tFeNoH.6at0jDcJ0
2 $6$3w2BAVbH$Yj/OhK0FAHxvJTY7B/UO4CBI5yNim5hfZ5iN.IJRHgfBM0FxK2gWdZ3E168ziKq0T2YwZgvbN5LDTfErgFfa//
3 $6$bhLUpEFT$22VQN1t5xXt7tvSHXO753XMD7vVDay8mTTuhJbFnPgaWMmz67ACO6CKXXvWwJFg7q5t2MxXxduAMEvFK.fJAC.
4
```

Investigation through .bash_history: Investigated user creation details from the /root/.bash_history file, revealing associations with the Black Order group from Marvel.

```
root@ubuntu:/root# cat .bash_history | grep 'useradd'
useradd
useradd -u 1001 -s /bin/bash -c "Thanos,,,," thanos
useradd -u 1001 -m -s /bin/bash -c "Thanos,,,," thanos
useradd -u 1002 -m -s /bin/bash 'c "Ebony Maw,,,," emaw
useradd -u 1002 -m -s /bin/bash -c "Ebony Maw,,,," emaw
useradd -u 1003 -c "Cull Obsidian,,," -m -s /bin/bash cobsidian
useradd -u 1004 -c "Proxima Midnight,,," -m -s /bin/bash pmidnight
useradd -u 1005 -c "Corvus Glaive,,," -m -s /bin/bash cglaive
root@ubuntu:/root# cd ..
```

Custom Wordlist Creation: Developed a custom wordlist exclusively comprising lowercase characters, inspired by the user creation details found.

```
┌──(kali㉿kali)-[~/sec_os_final]
└─$ cewl 'https://en.wikipedia.org/wiki/Black_Order_(comics)' --lowercase -d 1 -w blck_order.txt
CeWL 6.1 (Max Length) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
```

Exhaustive Brute Forcing and Hash Cracking: Executed exhaustive brute force techniques using the custom wordlist, followed by a python script tailored to combine words while considering password length restrictions and enforcing a lowercase word regex condition.

iter_script.py | new_iter_script.py

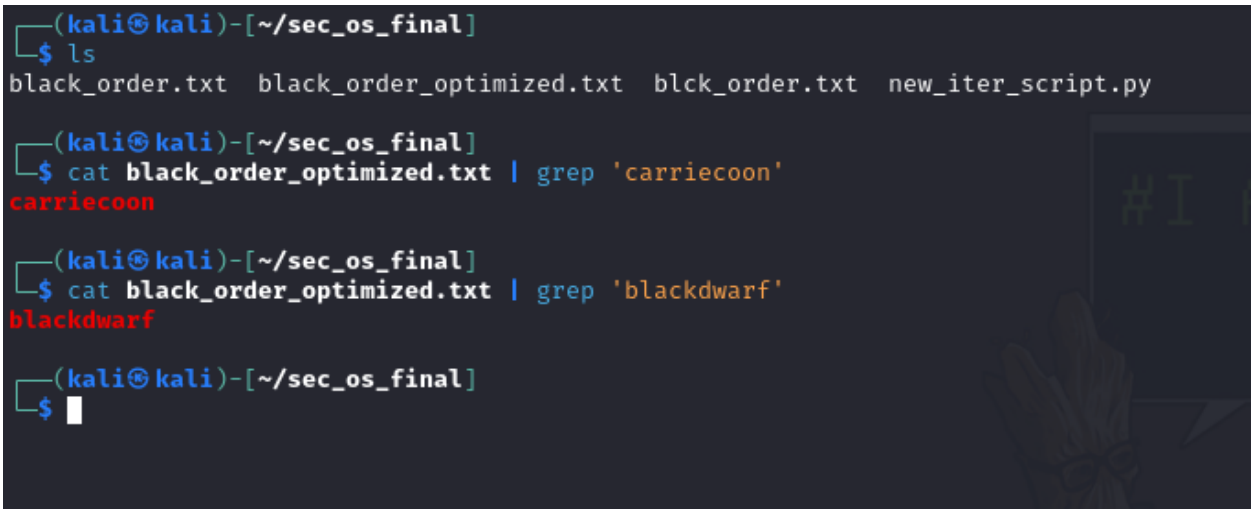
```python
import re

def contains_only_lowercase_english(word):
    return bool(re.match('^[a-z]+$', word))

def combine_words(input_file, output_file):
    with open(input_file, 'r') as file_in, open(output_file, 'w') as file_out:
        words = [line.strip() for line in file_in if contains_only_lowercase_english(line.strip())]

        for i in range(len(words)):
            for j in range(i + 1, len(words)):
                combined = words[i] + words[j]
                if 10 == len(combined):
                    file_out.write(combined + '\n')

input_filename = 'black_order.txt'
output_filename = 'black_order_optimized.txt'

combine_words(input_filename, output_filename)

```

Optimization for Performance: Modified the script to prioritize space over memory, enhancing performance in handling intensive operations.

iter_script.py | new_iter_script.py

```python
from itertools import product

def contains_only_lowercase(word):
    return all(char.islower() for char in word)

def combine_words(input_file, output_file):
    with open(input_file, 'r') as file:
        words = [line.strip() for line in file if contains_only_lowercase(line.strip())]

    unique_combinations = set()

    combinations = list(product(words, repeat=2))
    for combo in combinations:
        combined = ''.join(combo)
        if 8 <= len(combined) <= 16:
            unique_combinations.add(combined)

    with open(output_file, 'w') as file:
        for unique_combo in unique_combinations:
            file.write(unique_combo + '\n')

input_filename = 'black_order.txt'
output_filename = 'black_order_optimized.txt'

combine_words(input_filename, output_filename)

```

Execution Duration: The exhaustive brute forcing and hash cracking efforts spanned approximately 4 hours, resulting in successful password cracking for the "pmidnight" and "cobsidion" user accounts.

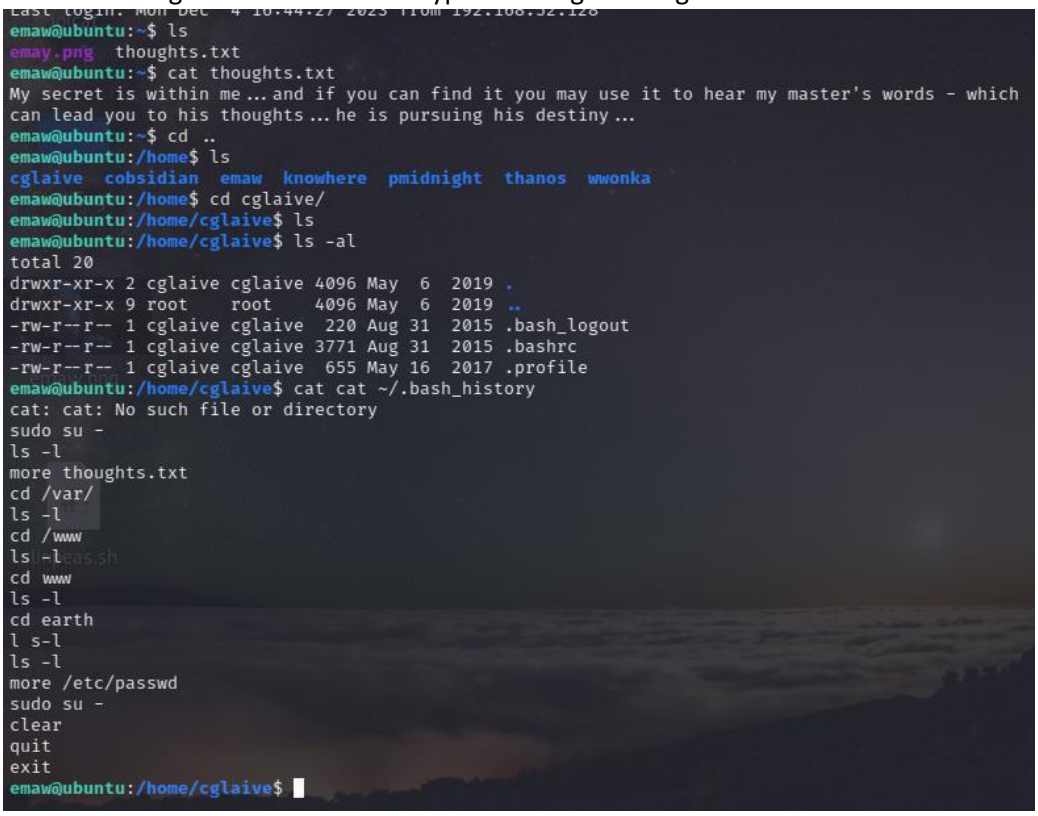

(kali㉿kali)-[~/sec_os_final]
$ ls
black_order.txt  black_order_optimized.txt  blck_order.txt  new_iter_script.py
(kali㉿kali)-[~/sec_os_final]
$ cat black_order_optimized.txt | grep 'carriecoon'
carriecoon
(kali㉿kali)-[~/sec_os_final]
$ cat black_order_optimized.txt | grep 'blackdwarf'
blackdwarf
(kali㉿kali)-[~/sec_os_final]
$

# XI. Deep Dive into System Directories and Decoding Messages

## 1. Unveiling Directory Contents(Directory Content Analysis):

Found "emaw.png" and "thoughts.txt" in a directory.

cat command on "thoughts.txt" revealed a cryptic message hinting at hidden information.




## 2. User Directory Inspection:

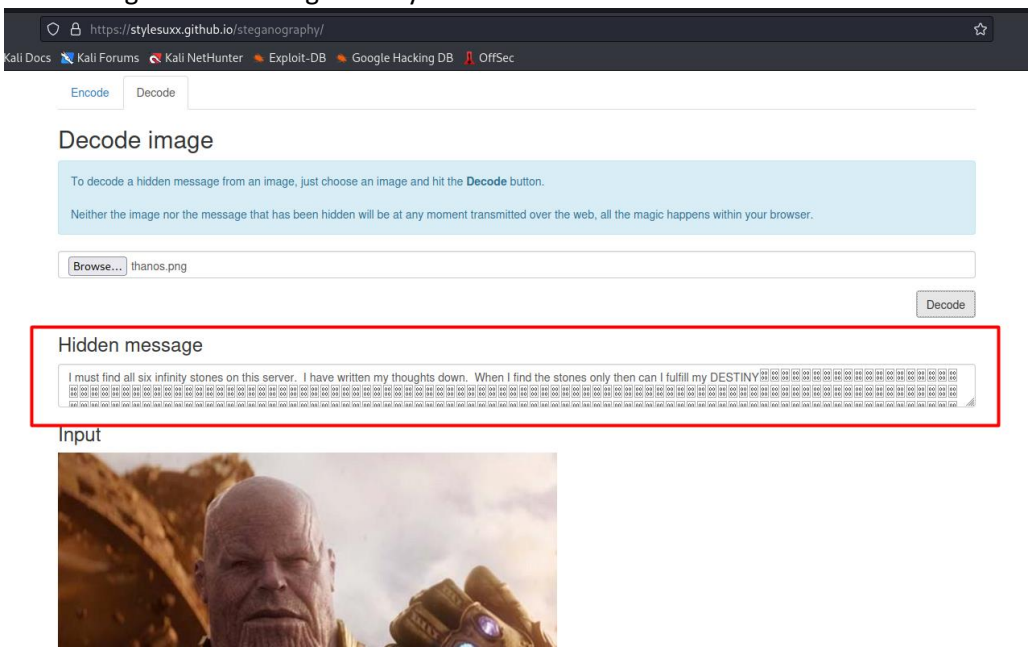
Explored individual user folders; "cglaive," "cobsidian," and "pmidnight" were empty.

Discovered encrypted files in "knowhere" ("stone3.enc") and "thanos" ("thanos.enc"), and "thanos.png" in "wwonka".

## 3. Decoding Message in Image File:

Used 'stylesuxx.github.io/steganography' on "thanos.png".

Uncovered a message about finding "infinity stones" on the server.



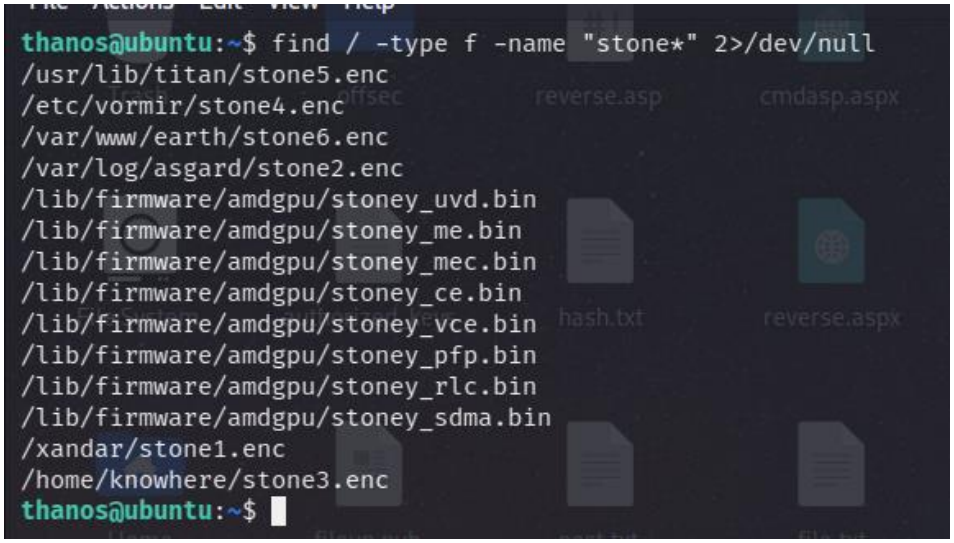

## 4. Regaining Root Access:

Reacquired root access using the previously employed payload 45010.c.

## 5. Search for "Infinity Stones":

Executed find / -type f -name "stone*" 2>/dev/null.

Located all six "infinity stones" within the server.



```
root@ubuntu:/root# cat .bash_history | grep "stone"
vi stone1.txt
vi stone2.txt
vi stone3.txt
vi stone4.txt
vi stone5.txt
vi stone6.txt
openssl aes-256-cbc -in stone1.txt -out stone1.enc
openssl aes-256-cbc -in stone2.txt -out stone2.enc
openssl aes-256-cbc -in stone3.txt -out stone3.enc
openssl aes-256-cbc -in stone4.txt -out stone4.enc
openssl aes-256-cbc -in stone5.txt -out stone5.enc
openssl aes-256-cbc -in stone6.txt -out stone6.enc
mv ../wwonka/stone1.enc .
mv stone1.enc /
mv /stone1.enc /xandar/
mv stone2.enc /var/log/asgard
mv stone3.enc /home/knowhere
mv stone4.enc /etc/vormir
mv stone5.enc /usr/lib/titan
mv stone6.enc /var/www/earth
more stone2.enc
root@ubuntu:/root#
```

## XII. Decryption of Encrypted Files and Message Interpretation

**Bash History Review and Encryption Insights**

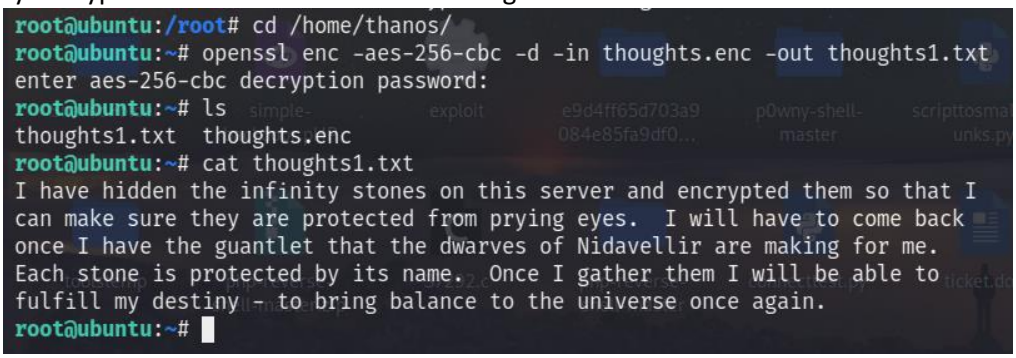
### 1. Bash History Analysis:

Confirmed use of "openssl" commands for encrypting "stone" files.

```
root@ubuntu:/root# cat .bash_history | grep "openssl"
which openssl
openssl aes-256-cbc -in stone1.txt -out stone1.enc
openssl aes-256-cbc -in stone2.txt -out stone2.enc
openssl aes-256-cbc -in stone3.txt -out stone3.enc
openssl aes-256-cbc -in stone4.txt -out stone4.enc
openssl aes-256-cbc -in stone5.txt -out stone5.enc
openssl aes-256-cbc -in stone6.txt -out stone6.enc
openssl aes-256-cbc -in thoughts.txt -out thoughts.enc
openssl aes-256-cbc -in thanos.png -out thanos.enc
openssl enc -aes-256-cbc -d -in thanos.enc -out thanos1.png
openssl enc -aes-256-cbc -d -in thoughts.enc -out thoughts1.txt
root@ubuntu:/root#
```

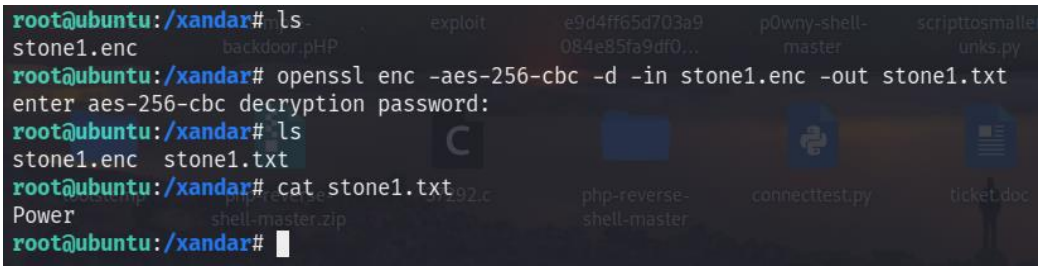
## 2. Decrypting "thoughts.enc":

Executed openssl enc -aes-256-cbc -d -in thoughts.enc -out thoughts1.txt.
Successfully decrypted and stored contents in "thoughts1.txt".

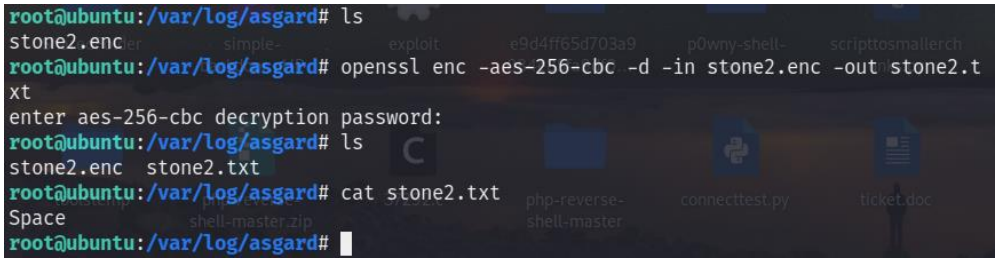



## 3. Decoding Strategy Based on Movie Theme:

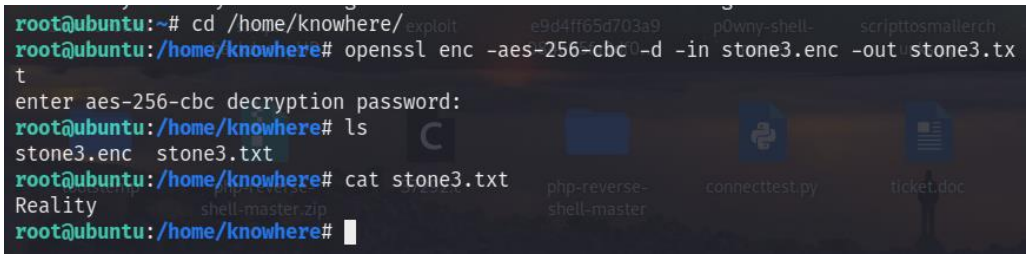
Decrypted "stone1.enc" with key "Power"; content stored in "stone1.txt":




Decrypted "stone2.enc" with key "Space"; content stored in "stone2.txt":




Decrypted "stone3.enc" with key "Reality"; content stored in "stone3.txt":




Decrypted "stone4.enc" with key "Soul"; content stored in "stone4.txt":

```
root@ubuntu:/etc/vormir# ls
stone4.enc
root@ubuntu:/etc/vormir# openssl enc -aes-256-cbc -d -in stone4.enc -out stone4.txt
enter aes-256-cbc decryption password:
root@ubuntu:/etc/vormir# ls
stone4.enc  stone4.txt
root@ubuntu:/etc/vormir# cat stone4.txt
Soul
root@ubuntu:/etc/vormir#
```

Decrypted "stone5.enc" with key "Time"; content stored in "stone5.txt":

```
root@ubuntu:/usr/lib/titan# ls
stone5.enc
root@ubuntu:/usr/lib/titan# openssl enc -aes-256-cbc -d -in stone5.enc -out stone5.tx
t
enter aes-256-cbc decryption password:
root@ubuntu:/usr/lib/titan# ls
stone5.enc  stone5.txt
root@ubuntu:/usr/lib/titan# cat stone5.txt
Time
```

Decrypted "stone6.enc" with key "Mind"; content stored in "stone6.txt":

```
root@ubuntu:/var/www/earth# ls
stone6.enc
root@ubuntu:/var/www/earth# openssl enc -aes-256-cbc -d -in stone6.enc -out stone6.tx
t
enter aes-256-cbc decryption password:
root@ubuntu:/var/www/earth# ls
stone6.enc  stone6.txt
root@ubuntu:/var/www/earth# cat stone6.txt
Mind
```