LAB GUIDE

Analyzing Microsoft 365 through PowerShell

Breakout Session Four – Calling the Graph API

Overview

This breakout session focuses on gathering credentials which can be used to connect with the Azure Portal systematically. The Azure API is continually growing to match the capabilities found in the PowerShell modules and much more.

There are two ways to access the Graph API: delegation and application access. In delegated access, OAuth + OIDC is leveraged to sign in a user and adopt their permissions for accessing the graph API. This method allows the application full access to the Graph APIs that your user account also has access to. The second method, Application access, gives a registered Azure AD application permission to query the API without end user interaction. Application access is not granted all the same APIs as an administrator as some API calls explicitly require end user interaction.

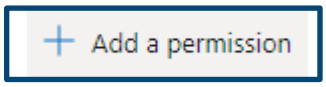
Finding valuate graph endpoints

The Microsoft Graph ecosystem is a rapidly growing ecosystem of valuable security information. In this lab, we focus on the authorization policy which controls default permissions for standard users and guest invititation. To start, let's look at the API we need.

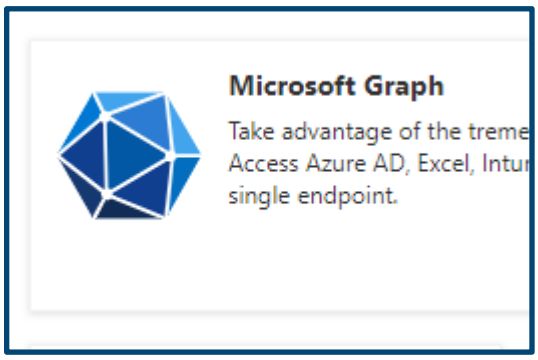
1. Open the API documentation here: https://docs.microsoft.com/sv-se/graph/api/authorizationpolicy-get?view=graph-rest-1.0
2. In the permissions section, we see that we need Policy.Read.All
3. We need to assign these permissions to the Service Principal

Granting permissions to the Application

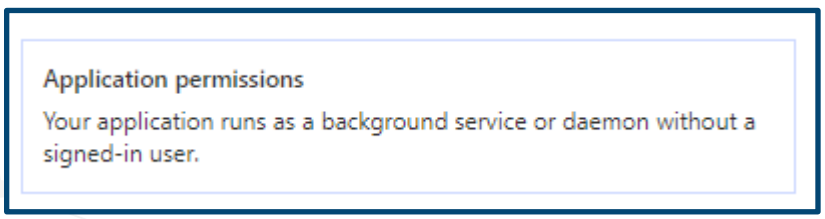
1. Login to the Azure AD portal: https://portal.azure.com/
2. In the top bar, search for "App Registrations" and select it.
3. Once the page loads, select the tab for "All Applications".
4. Search for the application name you specified in the previous labs.
5. In the left column of the application page, select "API Permissions".
6. Select "Add a permission" in the main content pain of the API Permissions page



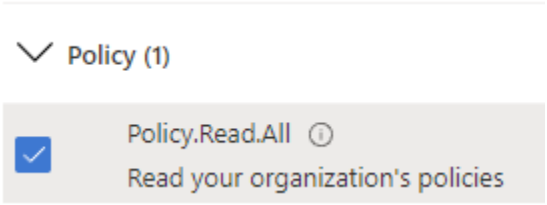
7. Select "Microsoft Graph"



8. Select "Application permissions"



9. Search for “Policy” and add Policy.Read.All to the applicatoin



10. Select “Add permissions” on the bottom of the permissions page.
11. Finally, select “Grant admin consent…” to approve the programmatic access to the graph API.

Using the Permissions

With Microsoft's PowerShell module for Microsoft Graph installed, we can call graph endpoints easily and effectively as the service principal. The following code snippet takes these steps:

1. It gathers the credential information
2. The credentials are sent to the Microsoft Graph for Authentication
3. We switch to the preferred Graph endpoint for the command (Beta)
4. We call the Authroization Policy. To determine the command:
    a. Start with the actual Graph Url:
    `https://graph.microsoft.com/v1.0/policies/authorizationPolicy`
    b. The command can be deduced as Get-MgPoliciesAuthorizationPolicy

```
# Gather Service Principal Credentials
$Cert = New-Object
System.Security.Cryptography.X509Certificates.X509Certificate('.\TenantScanner.cer')
$ThumbPrint = $Cert.GetCertHashString()
$TenantId = Get-Content .\TenantId.txt
$AppId = Get-Content .\AppId.txt
$Thumbprint = $Cert.GetCertHashString()

# Connect to Microsoft Graph
Connect-Graph -TenantId $TenantId -ClientId $AppId -CertificateThumbPrint $ThumbPrint

# Switch to the beta api
Select-MgProfile -Name beta

# Get the authorization policy information
Get-MgPolicyAuthorizationPolicy
```

With the authorization policy in hand, we can see interesting properties.

- Default Guest Role
    - Learn what this is and what it could be here.
- Self-Service Password Reset status
- Default User roles
    - Rights to create applications
    - Rights to create security groups
    - Right to view other user profiles
- AllowInvitesFrom (Who can invite external users)

Continue Developing

Go forth and implement these new checks into the tool and explore additional API endpoints. Some interesting endpoints include:

- Secure Score
- Directory Audit