Cloud Automated Security Audit

Jing (Jenny) Li
Department of Computer Science
New York Institute of Technology
Vancouver, Canada
jli140@nyit.edu

Nitin Tutlani
Department of Computer Science
New York Institute of Technology
Vancouver, Canada
ntutlani@nyit.edu

Samuel William Almeida
Department of Computer Science
New York Institute of Technology
Vancouver, Canada
salmei03@nyit.edu

***Abstract*—The demand and impact of cloud services have been significantly growing in global entities across various industries. Time and resources consuming traditional security auditing can no longer adapt to the fast-changing cloud environment. Automated security audit has become the big future for enterprises running businesses on the cloud. In this project, we develop an automated security audit application that can be used to report security check results against pre-configured compliance policies, including user identity, data, and end-point devices. This application will also provide recommendations for organizations for applying suggested security mechanisms, deploying appropriate security policies, or addressing the improvement action with third-party software or applications.**

Keywords—security audit, cloud, Microsoft 365, Microsoft Graph, Azure, secure score, policy, automation, C# code

I. INTRODUCTION

The benefits of Cloud Computing have been spurring a new norm in today's business operation solutions among various industries. Clients buy services from Cloud Service Providers (CSP) such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud, etc., without implementing or maintaining high-cost physical infrastructures. According to Gartner's report, the total spending on using public cloud services is expected to $494.7 billion in 2022 and will reach more. [1]

Under the profound impact of cloud services, security audit is inevitable. According to the National Institute of Standards and Technology (NIST), "security audit provides independent review and examination of a system's records and activities to determine the adequacy of system controls, ensure compliance with established security policy and procedure, detect breaches in security services, and recommend any changes that are indicated for countermeasures." [2] However, traditional security auditing is undertaken manually, involving interviewing employees, performing vulnerability scans, and evaluating policies regarding authentication, authorization, accounting, etc., which is heavily time and resources consuming.

For businesses running on the cloud, periodical and manual security audits and risk assessments become more inefficient in the fast-changing cloud environment. In the cloud, a simple misconfiguration, like an open API, may lead to a data breach attack, which is a true story happening: Optus, an Australian telecommunications company, is facing a $1 million ransom from a cybercriminal calming access to over 11 million records from customers including personal information, like passports. This is just because of the misconfiguration of valid authentication policies. [3] It is often found that companies would maintain a healthy state during the evaluation period of SOC 2 examination, ISO 27001 certification, or other security compliance audits, but security states become worse after that because there is no continuous audit in place. How to mitigate it?

The answer will be automation, which provides a feasible solution for continuous auditing. Using the proper automated audit tools, security teams can conduct a constant audit with real-time monitoring to check security risks in the cloud environment and take action accordingly. This means users must quickly gather, analyze, and report security-related data for on-demand assessment.

Unfortunately, most current tools like SolarWinds and Nessus are well designed for network security auditing under the on-premises landscape. The cloud-based automated security auditing tools are not prevalent or mature in this marketplace, not to mention an easy-to-use application, for that matter.

Microsoft 365 Defender measures compliance posture and provides Microsoft Secure Score based on the organization's configurations and policies, including cloud and on-premises environment. Thousands of security rules are measured, and the result Dashboard clearly demonstrates each item with a security concern comprehensively. However, this secure score will not update until 48 hours after any modifications or security actions are made, which means the security evaluation does not necessarily represent the current situation. This feature is great but has not had much popularity among cloud users.

In this project, we will take advantage of Microsoft's existing security audit capabilities and tools to design the new automated security audit application, which can be deployed on Microsoft 365 Marketplace and used among different tenants. This application will output assessment results as Pass or Fail against security criteria from established frameworks, such as Payment Card Industry Data Security Standard (PCI DSS), NIST, and other industry compliances. According to the failure information from the output, users can take actions in areas like User Identity Control, Device Management, etc., to improve security configurations and increase security scores.

II. METHODOLOGY AND RESOURCE

The project uses Microsoft cloud Azure and Microsoft 365 as the main platform. First, we build a virtual organization on

Azure and manage security configurations using applications under Microsoft 365 umbrella. Then, we utilize Microsoft Graph to fetch organizational security information set up by various applications. Finally, we create our own application to analyze the data we fetched from Microsoft Graph API and output security results according to different security rules. Figure 1 demonstrates the project methodology and workflow.

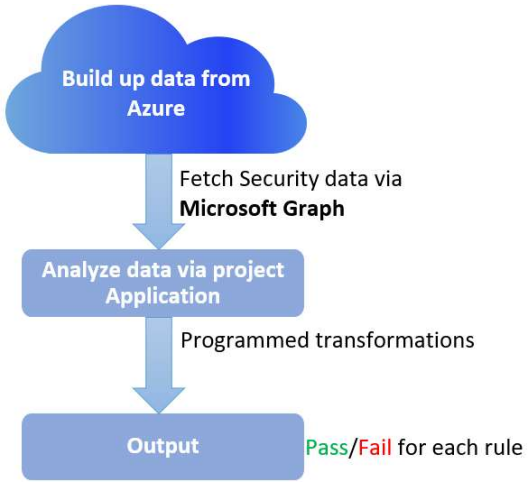


Fig. 1. Project Methodology & Workflow

The methodology being used is straightforward. We authenticate against Graph API using available libraries in a C# code project. After establishing the data access channel, we pull various data endpoints based on our research, analysis, and general documentation. Once we get hold of data containing the necessary information for assessing security, using code-based transformations, we generate a report based on individual security cases providing a comprehensive report to the platform users.

Thus, this project is predominantly working on the following resources and application platforms:

- Azure Portal, Azure Active Directory (AD)
- Microsoft 365: Office 365, Purview, Defender, Intune, 365 Admin Center
- Microsoft Graph
- C# Code

III. Build Up Data and Security Cases

A. *Build up Virtual Organization on Azure*

The purpose is to build up a virtual product organization from scratch to control it fully. This includes the following steps:

- Create tenant and organization
- Create users (employees)
- Create and assign Groups for users based on their job attributes

Figure 2. demonstrates that the organization "New York Institute of Technology" is created from the Azure portal for project test purposes.

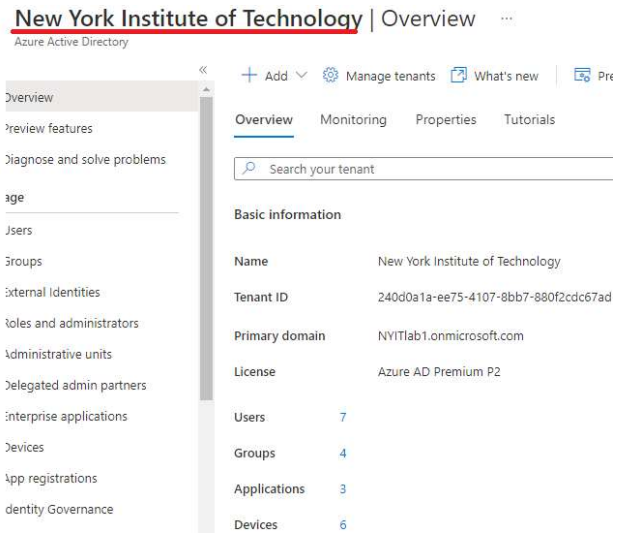


Fig. 2. Create Organization

Figure 3. demonstrates that different users are created within the organization.

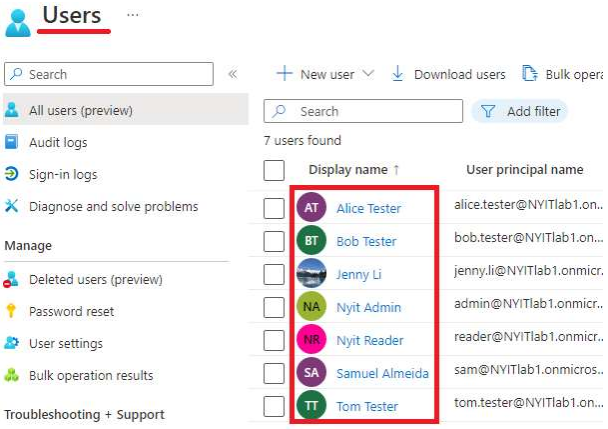


Fig. 3. Create Users

Figure 4. demonstrates that several groups are created; users are assigned to specific groups with different authorities granted.

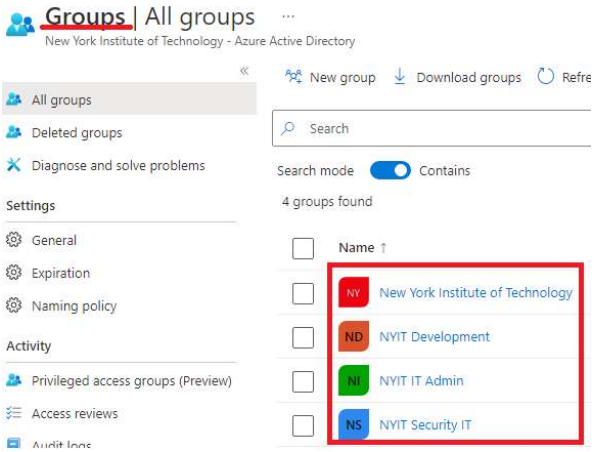


Fig. 4. Create and assign Groups

B. *Establish Seed Security Configurations*

The purpose is to ensure employees have the necessary authority to perform their daily job while following security policies in place, such as Multi-factor Authentication (MFA), Block Legacy authentication protocols, Enable Self-service password reset, etc.

Figure 5, Figure 6, and Figure 7 show an example of setting up MFA, Block Legacy authentication, and Self-service password reset policies in the Azure Active Directory admin center, respectively.

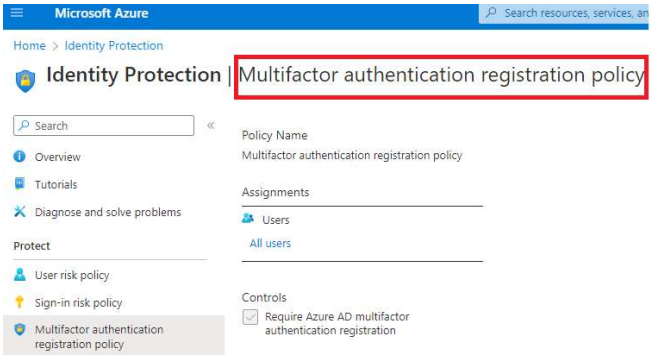


Fig. 5. MFA for users

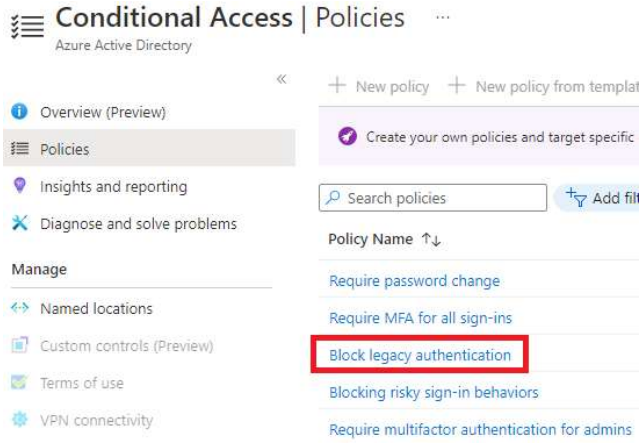


Fig. 6. Block legacy authentication policy

Note that Conditional Access Policies are enabled for testing purposes, as Figure 6 shown. Real users prefer Security Default in Azure AD, with this policy enabled by default, to avoid inadvertent misconfiguration. [4]

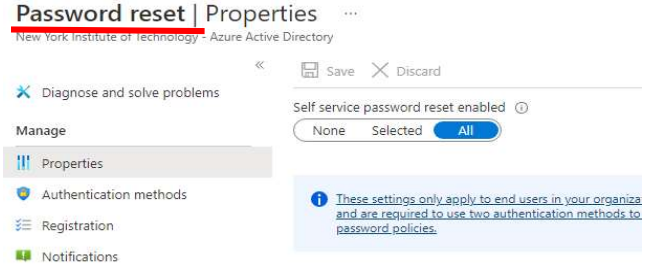


Fig. 7. Self-service password reset policy

C. *Fetch Security Information via Microsoft Graph*

The project relies on the fact that Microsoft Cloud Services consume Graph API infrastructure in the background. Microsoft Graph functions as the gateway to data and intelligence in Microsoft 365. The restful Graph API is the source of abundant information that flows through the cloud network to provide all its services, including Azure Active Directory, Microsoft Purview, Microsoft Intune, and Microsoft 365 Defender, to name a few. We rely on consuming this data securely and diligently to perform security analysis.

In this project, we went through massive documentation on the topics of Microsoft 365, Azure AD and Microsoft Graph, trying to figure out security checks (cases) regarding policies, configurations, and settings from Microsoft 365 applications. The data we fetch from the cloud includes users' information, device information, conditional policies saved on the cloud, etc. We categorized our security cases into the following three areas:

- Identity Security
- Device Security
- Data Security

For each security case, we nailed down the specific Graph API to find possible security data according to security cases.

Figure 8 and figure 9 demonstrate the user's information in the organization. We use Graph API "Get all users in the organization" (https://graph.microsoft.com/beta/users) to find out basic employee information and security information, such as if the multifactor service is enabled or not.


```
"id": "eae4bfc7-435e-4ced-9a4e-044fec11205f",
"deletedDateTime": null,
"accountEnabled": true,
"ageGroup": null,
"businessPhones": [],
"city": "Vancouver",
"createdDateTime": "2022-08-15T21:04:43Z",
"creationType": null,
"companyName": null,
"consentProvidedForMinor": null,
"country": "Canada",
"department": "IT",
"displayName": "Jenny Li",
"employeeId": null,
"employeeHireDate": null,
"employeeLeaveDateTime": null,
"employeeType": null,
"faxNumber": null,
"givenName": "Jenny",
"imAddresses": [
    "jenny.li@nyitlab1.onmicrosoft.com"
],
"infoCatalogs": [],
"isManagementRestricted": null,
"isResourceAccount": null,
"jobTitle": "Network Admin",
"legalAgeGroupClassification": null,
"mail": "jenny.li@NYITlab1.onmicrosoft.com",
```


Fig. 8. Get all users in the organization: employee's basic information [5]

```
"assignedDateTime": "2022-08-15T21:04:43Z",
"capabilityStatus": "Enabled",
"service": "MultiFactorService",
"servicePlanId": "8a256a2b-b617-496d-b51b-e76466e88db0"
```


Fig. 9. Get all users in the organization: multifactor service state [5]

Figure 10 demonstrates Azure conditional access policies information. We use Graph API "Get all Conditional Access policies" https://graph.microsoft.com/v1.0/identity/conditionalAccess/policies) to query all conditional policies set in place, such as the block legacy authentication policy shown in figure 10.


```
"id": "33b546e4-1356-467c-b384-eae826e31ca5",
"displayName": "Block legacy authentication",
"createdDateTime": "2022-09-03T23:14:14.2783731Z",
"modifiedDateTime": "2022-10-07T03:58:42.9078482Z",
"state": "disabled",
"sessionControls": null,
"conditions": {
    "userRiskLevels": [],
    "signInRiskLevels": [],
    "clientAppTypes": [
        "all"
    ],
    "servicePrincipalRiskLevels": [],
    "platforms": null,
    "locations": null,
    "devices": null,
    "clientApplications": null,
    "applications": {
        "includeApplications": [
            "None"
        ],
        "excludeApplications": [],
        "includeUserActions": [],
        "includeAuthenticationContextClassReferences": []
    },
    "users": {
        "includeUsers": [
            "All"
        ],
```


Fig. 10. Get all Conditional Access policies: block legacy authentication [5]

Figure 11 demonstrates a secure score measured from Microsoft 365 Defender. We use Graph API "Get secure scores" (https://graph.microsoft.com/beta/security/secureScores?$top=1) to get a non-real-time update comprehensive security output, such as the user risk policy shown in figure 11.


```
{
    "controlCategory": "Identity",
    "controlName": "UserRiskPolicy",
    "description": "With the user risk policy turned on, Azure Active Directory detects the probability that a user account has been compromised. As an administrator, you can configure a user risk Conditional Access policy to automatically respond to a specific user risk level. For example, you can block access to your resources or require a password change to get a user account back into a clean state.",
    "score": 0,
    "count": "7",
    "IsApplicable": "true",
    "IsEnforced": "false",
    "implementationStatus": "You have 7 users out of 7 that do not have user risk policy enabled.",
    "total": "7",
    "controlState": "active",
    "scoreInPercentage": 0,
    "lastSynced": "2022-10-26T00:00:00Z"
```


Fig. 11. Get secure scores: risky user policy [5]

Figure 12 and figure 13 demonstrate end-point device management information fetched from Microsoft Intune via Graph API "Get all azure active directory devices" (https://graph.microsoft.com/beta/deviceManagement/auditEvents), including all listed options. Here are rich security data we can take, such as bit locker settings shown in figure 13.

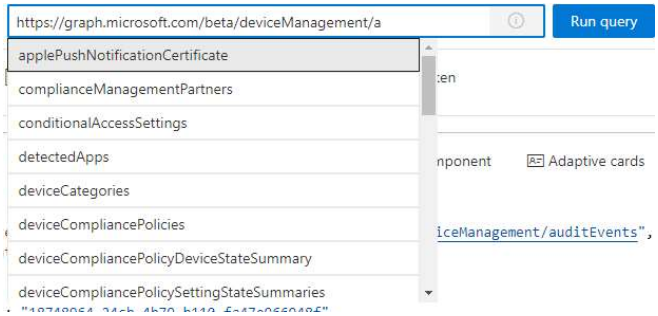


Fig. 12. Get all azure active directory devices: options


```
    "applicationId": "5926fc8e-304e-4f59-8bed-58ca97cc39a4",
    "applicationDisplayName": "Microsoft Intune portal extension",
    "userPrincipalName": "sam@NYITlab1.onmicrosoft.com",
    "servicePrincipalName": null,
    "ipAddress": null,
    "userId": "e0375dbc-ee6d-4b9b-92b2-502b428d74e8",
    "remoteTenantId": null,
    "remoteUserId": null,
    "userRoleScopeTags": []
},
"resources": [
    {
        "displayName": "Bit locker",
        "type": "DeviceManagementConfigurationPolicy",
        "auditResourceType": "DeviceManagementConfigurationPolicy",
        "resourceId": "a731ee3b-70c3-417a-b077-a5f20183f68b",
        "modifiedProperties": []
    },
    {
        "displayName": "<null>",
        "type": "DeviceManagementConfigurationPolicyAssignment",
        "auditResourceType": "DeviceManagementConfigurationPolicyAssignment"
```


Fig. 13. Get all azure active directory devices: bit locker

By extracting data from Microsoft Graph, we constructed 42 security rules as security testing cases for this project. (You can find detailed information on each case in the Appendix Security Cases.) The following content of this report explains how to configure and implement the new automated security audit application based on the security cases we demonstrated above. Note that there could be significantly more security cases designed than the number 42, which could be very useful for a security baseline setup and assessment.

IV. Application Configuration

A. *Create and Register Application in Azure Portal*

First, we register a new application in the Azure portal that is needed to grant the authorization to read data from Graph API. To do this, we follow the tutorials from [6].

Figure 14 shows the app we created in this project, with the following items listed that are critical for downstream operation:

- Application Display Name: Security and Compliance Score

- Application (client) ID: f9f12bec-7f61-4cfc-81ca-c423f69a0ccc
- Directory (tenant) ID: 240d0a1a-ee75-4107-8bb7-880f2cdc67ad

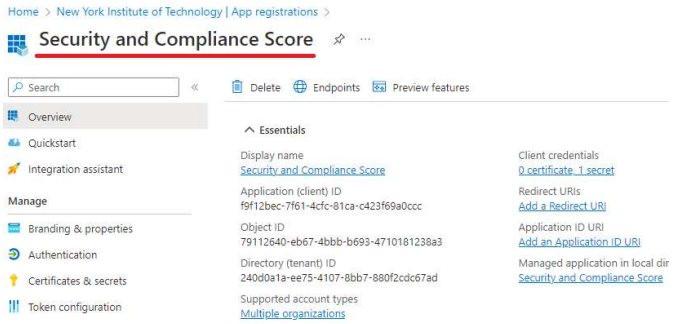


Fig. 14. Project Application: Security and Compliance Score

B. *Configure Application Permissions for Graph API*

It is essential to mention that most of our research involved using the tool Microsoft Graph Explorer. We basically communicate with Graph API, from which we extract relevant data. We need Microsoft Graph to grant our "Security and Compliance Score" app Read permissions. To achieve this, we can go to API permissions in the Azure portal and request permissions needed.

Figure 15 demonstrates Read API permissions from Microsoft Graph. Figure 16 lists permissions granted after the request.

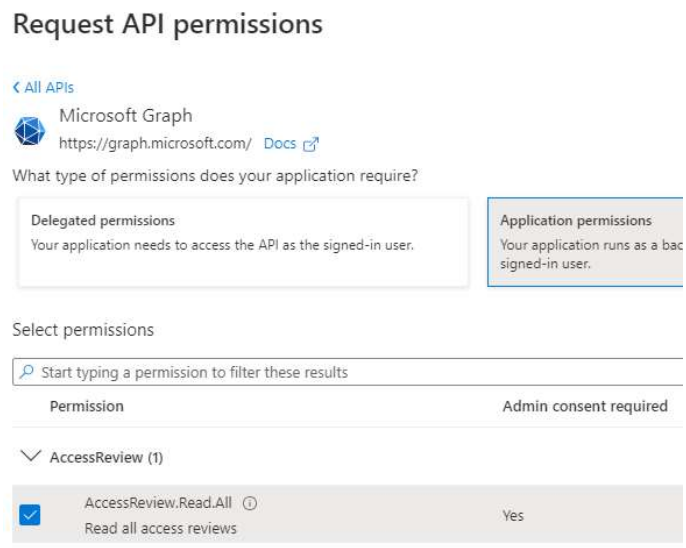


Fig. 15. Request API permissions for Microsoft Graph

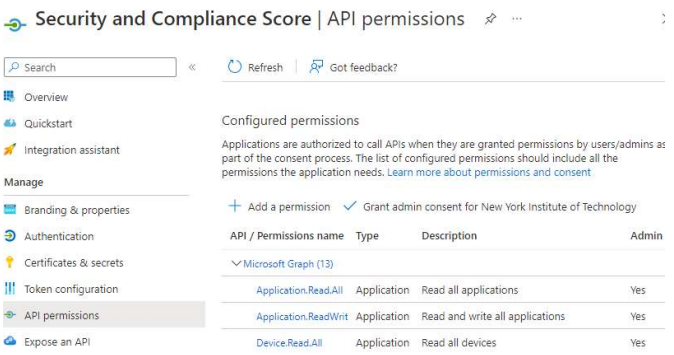


Fig. 16. Read permissions granted to Security and Compliance Score App

C. *Configure Application Authentication*

There are several ways to use the new project application, which are supported by the Microsoft Authentication Library (MSAL). The documentation in [7] clearly explains the authentication scenarios, and for this project, we are looking at using the Client credentials method, which is often referred to as daemons or service accounts, compared to user accounts that are needed to authenticate manually. To achieve the Client credentials authentication, we can either configure it using certificates (more secure) or Application secrets. This project used the Application secrets method. [7]

In general, our purpose is to allow authorized users to use this application to examine their organization's security status. There are two things to be done:

- Create and configure Client Secret
- Allow application for multi-tenants

Figure 17 demonstrates how to create a Client Secret key in the Azure AD admin center, which the tenant admin will use to grant application permissions, so users don't have to be required to get permissions manually when using this app.

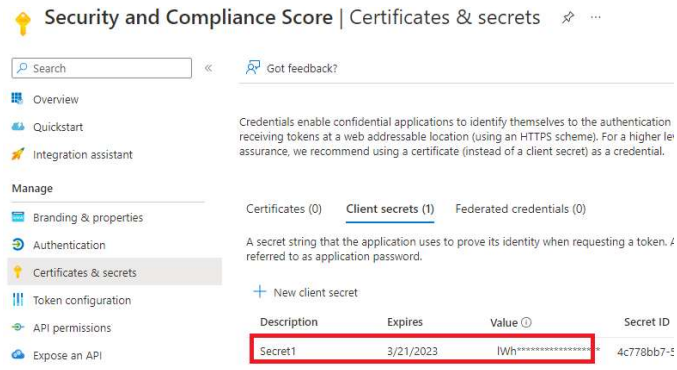


Fig. 17. Create Secrets in Azure AD admin center

Note that Client secret values cannot be viewed except immediately after creation. Be sure to save it before leaving the creation page.

Figure 18 shows how to allow the new applications to be used among all organizations running operations on Azure.

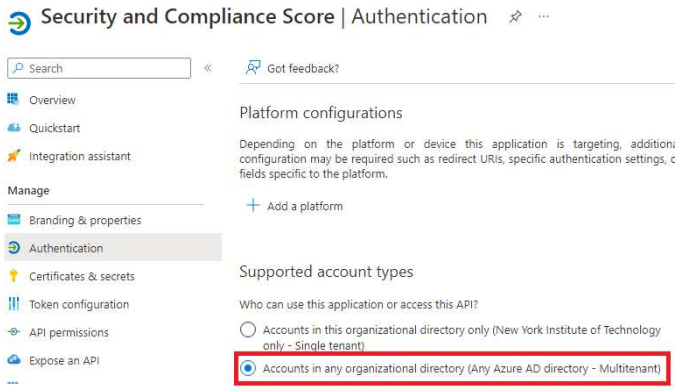


Fig. 18. Allow app to be used by all organizations

Now, we have completed all basic configurations for our new application, "Security and Compliance Score". In the next section, we will explain how to implement the C# code to analyze the data fetched from Graph API and output Pass/Fail results with necessary suggestions to improve the organization's security condition.

V. Code Implementation and Output

Ideally, developing a user-friendly UI will be more attractive; for now, we only implemented as command line interface (CLI) to show the functional prototype. We use C# Code for all the programming work.

A. Application Code

In this project, we have implemented ten security cases from the best practices on Microsoft 365 tenant security, including

- MFA enabled
- Block legacy protocol authentication
- Enable Risky Users/Risky Sign In policies
- Tenant Audit enabled
- Enable Self Links
- Enable Self Attachments
- Enable Self-service Password Reset
- Implement DMARC/DKIM protection for domain names
- Do not allow users to grant consent to unmanaged apps
- Regular review of security logs and reports (we should be able to see the last access date on reports) [8]

The code comprises three interfaces/classes: ITestCase, TestHelper, and Command.

1) ITestCase: To extend all security cases, see figure 19.

```
interface ITestCase
{
    5 references
    string name
    {
        get;
    }
    5 references
    string solution
    {
        get;
    }

    5 references
    Task<bool> Test(GraphServiceClient appClient)
}
```

Fig. 19. Interface of ITestCase

2) TestHelper: To handle application authentication, see figure 20.

```
class TestHelper
{
    // Settings object
    private static Settings? _settings;

    // App-ony auth token credential
    private static ClientSecretCredential? _clientSecretCredential;
    // Client configured with app-only authentication
    private static GraphServiceClient? _appClient;

    1 reference
    public static void InitializeGraph(Settings settings)
    {
        _settings = settings;

        // Ensure settings isn't null
        _ = _settings ??
            throw new System.NullReferenceException("Settings cannot be null");

        if (_clientSecretCredential == null)
        {
            _clientSecretCredential = new ClientSecretCredential(
                _settings.TenantId, _settings.ClientId, _settings.ClientSecret);
        }

        if (_appClient == null)...
    }

    1 reference
    public static Task<bool> Test(ITestCase testCase)...
}
```

Fig. 20. Class of TestHelper

3) Command: Responsible for the console runner and consumer of the security cases, see figure 21.

```
Console.WriteLine("Security and compliance score - CLI\n");

var settings = Settings.LoadSettings();

// Initialize Graph
TestHelper.InitializeGraph(settings);

int choice = -1;

while (choice != 0)
{
    Console.WriteLine("Please choose one of the following options:");
    Console.WriteLine("0. Exit");
    Console.WriteLine("1. Authentication");
    Console.WriteLine("10. Password Authentication");
    Console.WriteLine("2. Conditional Access");
    Console.WriteLine("20. Require MFA for Admins");
    Console.WriteLine("21. Block Legacy Authentication");
    Console.WriteLine("3. Secure Scores");
    Console.WriteLine("30. Role Overlap");
    Console.WriteLine("31. One Admin");
    Console.WriteLine("32. Legacy Authentication");
    Console.WriteLine("33. Admin MFA");

    try...

    ITestCase? testCase = null;

    switch (choice)...

    if (testCase != null)...
}
```

Fig. 21. Class of Command

According to different security cases, specific Graph API(s) will be called using the Microsoft Graph SDKs. To make it work, we follow the instructions in [9].

B. Run and Output Result

This project application runs on CLI, which can be launched from Windows Command Prompt or PowerShell. We use .NET to run the web app implemented by C# code. Before running the application, it is necessary to add the package "UserSecrets" to use Client Secrete authentication, see figure 22.

```
C:\Users\lj_xs>PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\lj_xs> cd Project870
PS C:\Users\lj_xs\Project870> dotnet add package Microsoft.Extensions.Configuration.UserSecrets
```

Fig. 22. Add package UserSecrets

Run the application by "dotnet run .NET Graph Tutorial". There is no need for the user to do any authentication to run the app because the "secret" has already been configured for the web application. To check the result of any security cases, users just need to choose the number of specific security cases according to the instruction.

In figure 23, we run a security check for the case of Admin MFA, and the application tells us the organization has this security check Passed.

In figure 24, we run a security check for the case of Block Legacy Authentication, and the application reported a failure check result with the recommendation that "A policy that restricts legacy Authentication must be enabled."

```
Please choose one of the following options:
0. Exit
1. Authentication
10. Password Authentication
2. Conditional Access
20. Require MFA for Admins
21. Block Legacy Authentication
3. Secure Scores
30. Role Overlap
31. One Admin
32. Legacy Authentication
33. Admin MFA
33
Secure scores - AdminMFAV2 : PASS
```

Fig. 23. Run and Output Security Case of Admin MFA - PASS

```
21. Block Legacy Authentication
3. Secure Scores
30. Role Overlap
31. One Admin
32. Legacy Authentication
33. Admin MFA
21
Conditional Access - Block legacy authentication : FAIL (A policy that
restricts Legacy Authentication must be enabled)
```

Fig. 24. Run and Output Security Case of Block Legacy Authentication – FAIL

Note that in many cases, Microsoft Graph may not capture the settings or values directly but instead verifies the availability of relevant policy. For example, if we check the bit locker data from Graph API, it will not simply give you if a specific end-point device has bit locker enabled or not. Microsoft 365 ecosystem relies heavily on the availability of policies and status reports against policy enforcement. This is a fundamental concept in dealing with Users, Roles, Groups and Devices. This helps us categorize security test cases mainly into two aspects:

- when a policy is missing
- when a policy is not enforced

The application can safely report a "FAIL" result if the relevant policy is missing, which should be an alert that is well worthy of attenuation from the organization's security team.

VI. CONCLUSION

During this project, we covered most areas that can be used to build a security score for the production. Even though the project does not provide full coverage, we validated the implementation and identified measures to help a team continue working on this task. We explored massive APIs in Microsoft Graph and also provided code examples for accessing and analyzing related data for security case transformations.

In our findings, Microsoft 365 APIs can be safely categorized on three mechanisms: state-driven, policy-driven and log (report)-driven. Leveraging security cases on each STRIDE category involves a deep understanding of the available Graph APIs and the mechanism with which the APIs are structured.

Organizations can keep an eye on and improve the security of their Microsoft 365 identities, apps, and devices according to the security audit feedback from the project application. Secure Score assists businesses by reporting on the organization's security posture as it stands today. You can strengthen their security posture by giving them discoverability, visibility,

direction, and control. Create key performance indicators and compare them against benchmarks (KPIs).

For future tasks, here are some points we suggest that could be further worked on. First, in our project, to evaluate our devices and user identities, we manually create the policies and configure settings from various Microsoft 365 application platforms. But a future extension to this project can be a scenario where the policies are scripted and automatically created and deployed using our API. This will be more efficient and better to review when running the automated security audit application, and also guarantee that all devices are evaluated against the suitable compliance standard and not the one that the organization has created. Because anyone in the organization can modify the policy, but if it has been automated to install with our API it will be tamperproof. Second, we only create CLI for our project purpose, but one can develop web API or web App in the future. More, we only constructed 42 security checks in this project, but there could be much more to dig in. Microsoft has provided us meaningful security output scattered in various application services. Organizations will benefit from having a single consolidated application covering all such security checks, providing a centralized comprehensive report, and automatically updating or deploying proper configurations and policies.

REFERENCES

[1] "Gartner Forecasts Worldwide Public Cloud End-user spending to reach nearly $500 billion in 2022," *Gartner*:
https://www.gartner.com/en/newsroom/press-releases/2022-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-500-billion-in-2022

[2] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, A. Hahn, "Guide to Industrial Control Systems (ICS) Security", pp. 132, 2015
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf

[3] E. Kedrosky, "Optus Faces $1Million Ransom Due to Cloud Misconfiguration", sonrai, 2022.9:
https://sonraisecurity.com/blog/optus-faces-1million-ransom-due-to-cloud-misconfiguration/

[4] Security defaults in Azure AD, Microsoft:
https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/concept-fundamentals-security-defaults

[5] Microsoft Graph, Microsoft:
https://developer.microsoft.com/en-us/graph/graph-explorer

[6] Register the app in the portal, Microsoft Graph, Microsoft:
https://learn.microsoft.com/en-us/graph/tutorials/dotnet?tabs=aad&tutorial-step=1

[7] Authentication flow support in MSAL, Azure, Microsoft, 2022:
https://learn.microsoft.com/en-us/azure/active-directory/develop/msal-authentication-flows

[8] Configure your Microsoft 365 tenant for increased security, Microsoft 365, Microsoft, 2022:
https://learn.microsoft.com/en-us/microsoft-365/security/office-365-security/tenant-wide-setup-for-increased-security?view=o365-worldwide

[9] Make API calls using the Microsoft Graph SDKs, Microsoft Graph, Microsoft, 2022:
https://learn.microsoft.com/en-us/graph/sdks/create-requests?source=recommendations&tabs=CS

APPENDIX: SECURITY CASES

A. Data Security

1) Case 1: Data Loss Prevention

Description: Data Loss Prevention (DLP) policies allow content in multiple locations, such as, devices, Exchange online and Teams chats to be scanned for specific types of data like social security numbers, credit card numbers, or passwords.
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Pass
Reason: DLP is enabled.


```
{
    "controlCategory": "Data",
    "controlName": "dlp_datalossprevention",
    "description": "Data Loss Prevention (DLP) policies allows content in multiple locations,
such as, devices, Exchange online and Teams chats to be scanned for specific types of data like social security
numbers, credit card numbers, or passwords.",
    "score": 5,
    "on": "true",
    "scoreInPercentage": 100,
    "implementationStatus": "",
    "IsApplicable": "true",
    "lastSynced": "2022-10-01T14:02:21Z",
    "source": "ingestion"
},
```


Configuration: go to Microsoft Purview->Data loss prevention->Policies, “Default Office 365 DLP policy”, Status: On

2) Case 2: Purview Label Consent

Description: Applying sensitivity labels to keep data secure by stating how sensitive certain data is in your organization.
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Pass


```
{
    "controlCategory": "Data",
    "controlName": "mip_purviewlabelconsent",
    "description": "        To get work done, people in your organization collaborate with others
both inside and outside the organization. Data doesn't always stay in your cloud, and often roams
everywhere—across  devices, apps, and services. When your data roams, you still want it to be secure in a way that
meets your organization's business and compliance policies.        <br/>        <br/>        Applying sensitivity
labels to your content helps you keep your data secure by stating how sensitive certain data is in your
organization. It also abstracts the data itself, letting you track the type of data without exposing sensitive data
on other platforms.        <br/>        <br/>        For example, applying the sensitivity label 'highly
confidential' to a document that contains social security numbers and credit card numbers helps you identify the
sensitivity of the document without knowing the actual data in the document.        <br/>        <br/>        The
sensitivity labels created in Microsoft Purview Information Protection can also be extended to the Microsoft
Purview data map. When you apply a label on an office document and then scan it into the Microsoft Purview data
map, the label will be applied to the data asset.     ",
    "score": 1,
    "on": "true",
    "scoreInPercentage": 100,
    "implementationStatus": "The setting is properly enabled.",
    "IsApplicable": "true",
    "lastSynced": "2022-10-01T14:02:21Z",
    "source": "ingestion"
},
```

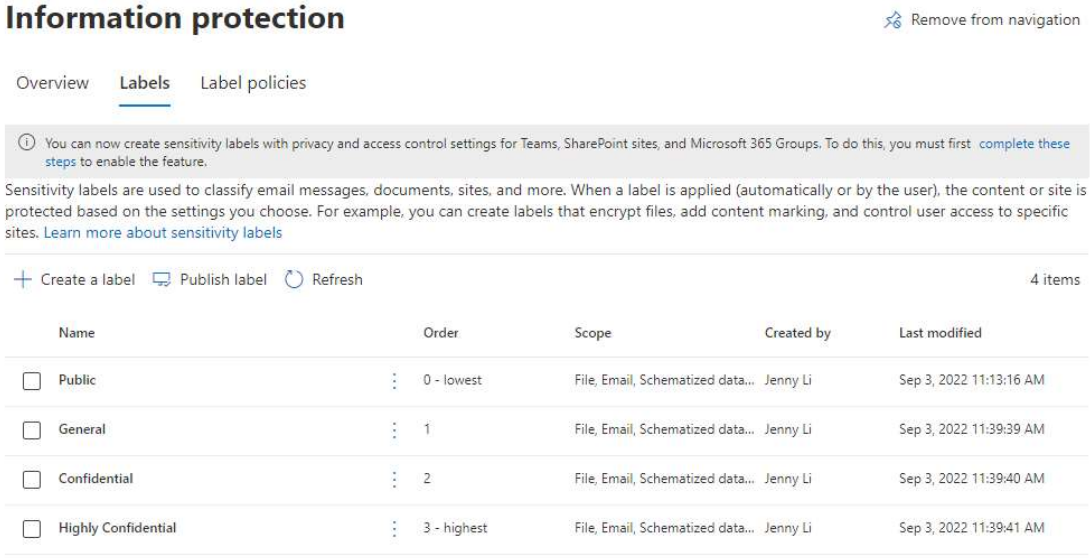

Reason: The setting is properly enabled.
Configuration: Go to Microsoft Purview->Information protection->Turn on now

3) Case 3: Sensitivity Labels Policy

Description: Set up and use data classification policies on data stored in your users' Office apps (like Outlook and Word), SharePoint sites, and Office 365 groups.
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Pass
Reason: Policies applied on all users.

```
{
    "controlCategory": "Data",
    "controlName": "mip_sensitivitylabelspolicies",
    "description": "Set up and use data classification policies on data stored in your users'
Office apps (like Outlook and Word), SharePoint sites, and Office 365 groups. <br/><br/>The policies will help
categorize your most important data so you can effectively protect it from illicit access and will help make it
easier to investigate discovered breaches. <br/><br/>Creation of data classification policies will not cause a
significant impact to an organization. However, ensuring long term adherence with policies can potentially be a
significant training and ongoing compliance effort across an organization. Organizations should ensure that
training and compliance planning is part of the classification policy creation process.<br/><br/><i>This
information was taken from Center for Internet Security (CIS).</i>    ",
    "score": 2,
    "total": "7",
    "scoreInPercentage": 100,
    "implementationStatus": "Policies were published on 7 of the 7 users",
    "IsApplicable": "true",
    "lastSynced": "2022-10-01T14:02:21Z",
    "source": "ingestion",
    "count": "7"
},
```


Configuration: Go to Microsoft Purview->Information protection; create security label policy and publish to all users



4) *Case 4: Auto sensitivity labels policies (should be improved now)*

Description: Create auto-labeling policies to automatically apply sensitivity labels to email messages or OneDrive and SharePoint files that contain sensitive info.

API: https://graph.microsoft.com/beta/security/secureScores?$top=1

Result: Fail

Reason: Policies were not published to any users.


```
{
    "controlCategory": "Data",
    "controlName": "mip_autosensitivitylabelspolicies",
    "description": "       Create auto-labeling policies to automatically apply sensitivity labels
to email messages or OneDrive and SharePoint files that contain sensitive info.       <br/>       This ability to
apply sensitivity labels to content automatically is important because:       <br/>       You don't need to train
your users on the appropriate way to use each of your classifications.       <br/>       You don't need to rely
on users to classify all content correctly.       <br/>       Users no longer need to know about your
policies—they can instead focus on their work.    ",
    "score": 0,
    "total": "7",
    "scoreInPercentage": 0,
    "implementationStatus": "Policies were published on 0 of the 7 users",
    "IsApplicable": "true",
    "lastSynced": "2022-10-01T14:02:21Z",
    "source": "ingestion",
    "count": "0"
},
```


Recommendation: Create and publish auto-labeling policy properly

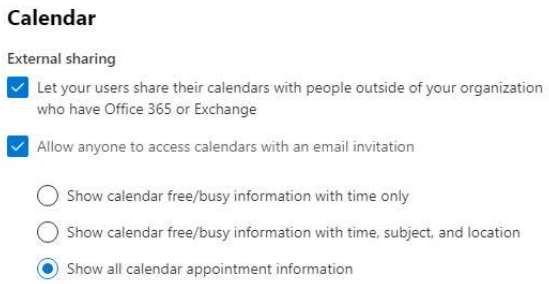
Configuration: Go to Microsoft Purview->Information protection; create a new label, turn on auto label option, add sensitive info condition, and publish the new policy

5) *Case 5: Calendar sharing with external users (should be improved now)*

Description: Users should not be allowed to share the full details of their calendars with external users.
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Fail
Reason: no such control implemented


```
{
    "controlCategory": "Apps",
    "controlName": "exo_individualsharing",
    "description": "Users should not be allowed to share the full details of their calendars with external users.",
    "score": 0,
    "on": "false",
    "IsApplicable": "true",
    "implementationStatus": "",
    "scoreInPercentage": 0,
    "lastSynced": "2022-10-02T09:31:42Z",
    "source": "ingestion"
},
```


Recommendation: Change the default to not allow calendar sharing with external users
Configuration: Go to Microsoft 365 admin center->Org settings->Calendar->turn off "Let your users share their calendars with people outside of your organization who have Office 364 or Exchange" and Save



6) *Case 6: Restrict anonymous users from joining meetings*

Description: By restricting anonymous users from joining Microsoft Teams meetings, you have full control over meeting access. Anonymous users may not be from your organization and could have joined for malicious purposes, such as gaining information about your organization through conversations.
API: API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Fail
Reason: No such control enabled for anonymous users joining meetings


```
        {
            "controlCategory": "Apps",
            "controlName": "meeting_restrictanonymousjoin_v1",
            "description": "By restricting anonymous users from joining Microsoft Teams meetings, you have
full control over meeting access. Anonymous users may not be from your organization and could have joined for
malicious purposes, such as gaining information about your organization through conversations.",
            "score": 0,
            "lastSynced": "2022-09-14T17:24:14Z",
            "source": "ingestion",
            "IsApplicable": "true",
            "implementationStatus": "current status: On",
            "scoreInPercentage": 0,
            "on": "false"
        },
```


Recommendation: Enable control to restrict anonymous users from joining meetings
Configuration: Go to Microsoft Teams admin center->Meeting settings->turn off "Anonymous users can join a meeting"

B. *App Security*

1) *Case 1: Idle session timeout (should be set now)*

Description: Idle session timeout signs users automatically out of Office web apps after a period of inactivity.
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Fail
Reason: The setting is not compliant.

```
{
    "controlCategory": "Apps",
    "controlName": "spo_idle_session_timeout",
    "description": "Idle session sign-out lets you specify a time at which users are warned and are later
signed out of Microsoft 365 after a period of browser inactivity in SharePoint and OneDrive.           <br/>This policy is
one of several you can use with SharePoint and OneDrive to balance security and user productivity and help keep your data
safe, regardless  of where users access the data from, what device they're working on, and how secure their network
connection is.",
    "score": 0,
    "lastSynced": "2022-10-02T22:02:27Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "implementationStatus": "The setting is not compliant.",
    "scoreInPercentage": 0,
    "on": "false"
},
```


Recommendation: Set Idle session timeout control properly

Configuration:

Go to Microsoft 365 admin center->Org settings->Security & privacy->Idle session timeout->turn on and set a proper time

Go to SharePoint admin center->Access control->Idle session sign-out->turn on Idle session sign-out

2) *Case 2: Mailbox Intelligence*

Description: Turns on artificial intelligence (AI) that identifies users' email patterns with their frequent contacts to spot potential phishing attempts.

API: https://graph.microsoft.com/beta/security/secureScores?$top=1

Result: Pass

Reason: Mailbox AI is enabled and applied to all users.


```
{
    "controlCategory": "Apps",
    "controlName": "mdo_enablemailboxintelligence",
    "description": "     Turns on artificial intelligence (AI) that identifies users' email patterns with their
frequent contacts to spot potential phishing attempts.    ",
    "score": 8,
    "lastSynced": "2022-10-02T09:31:42Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "total": "7",
    "implementationStatus": "",
    "scoreInPercentage": 100,
    "noPolicies": "false",
    "mdoImplementationStatus": "{\"Policies\":[{\"PolicyName\":\"Office365 AntiPhish Default\",\"Status\":2,
\"UsersInPolicy\":7,\"MissingDomains\":null}],\"TotalUsers\":7}",
    "count": "7"
},
```


Configuration: Go to Microsoft 365 Defender->Anti-phishing->Office365 AntiPhish Default is on by default

3) *Case 3: Automatically admit people in meetings*

Description: Users who aren't invited to a meeting shouldn't be let in automatically, because it increases the risk of data leaks, inappropriate content being shared, or malicious actors joining. If only invited users are automatically admitted, then users who weren't invited will be sent to a meeting lobby. The host can then decide whether or not to let them in.

API: https://graph.microsoft.com/beta/security/secureScores?$top=1

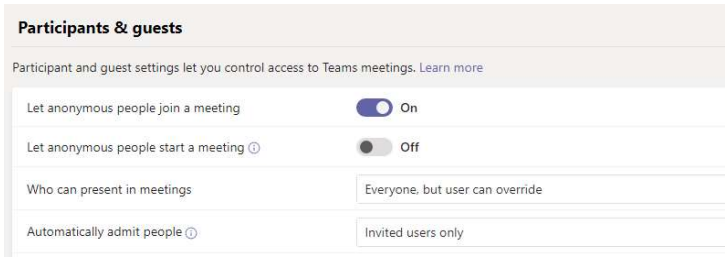
Result: Fail

Reason: meeting in automatically for only invited users is not implemented


```
{
    "controlCategory": "Apps",
    "controlName": "meeting_autoadmitusers_v1",
    "description": "Users who aren't invited to a meeting shouldn't be let in automatically,
because it increases the risk of data leaks, inappropriate content being shared, or malicious actors joining. If
only invited users are automatically admitted, then users who weren't invited will be sent to a meeting lobby.
The host can then decide whether or not to let them in.",
    "score": 1,
    "lastSynced": "2022-09-14T17:24:14Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "implementationStatus": "",
    "scoreInPercentage": 50
},
```


Recommendation: implement control to only allow invited users to join meetings automatically

Configuration: Go to Microsoft Teams admin center->Meeting policies->New a policy and set Automatically admit people as Invited users only



4) *Case 4: Share content during meetings*

Description: Only allow users with presenter rights to share content during meetings. Restricting who can present limits meeting disruptions and reduces the risk of unwanted or inappropriate content being shared.

API: https://graph.microsoft.com/beta/security/secureScores?$top=1

Result: Fail

Reason: no such policy to control content sharing during meetings


```
{
    "controlCategory": "Apps",
    "controlName": "meeting_designatedpresenter_v1",
    "description": "Only allow users with presenter rights to share content during meetings.
Restricting who can present limits meeting disruptions and reduces the risk of unwanted or inappropriate content
being shared.  ",
    "score": 0,
    "lastSynced": "2022-09-14T17:24:14Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "implementationStatus": "",
    "scoreInPercentage": 0
},
```


Recommendation: implement policy to control content sharing during meetings properly

Configuration: Go to Microsoft Teams admin center->Meeting policies->New a policy and configure present and sharing properly.

5) *Case 5: Firewall Log collector*

Description: Log collectors provide visibility into cloud app usage so you can identify if there are any apps that run without official approval, or if there is anomalous behavior. Log collectors automatically upload reports and parse the firewall/ proxy traffic logs to see if there is a match with your services in the Cloud App Catalog.

API: https://graph.microsoft.com/beta/security/secureScores?$top=1

Result: Fail

Reason: no such feature in place


```
{
    "controlCategory": "Apps",
    "controlName": "McasFirewallLogUpload",
    "description": "Log collectors provide visibility into cloud app usage so you can identify
if there are any apps that run without official approval, or if there is anomalous behavior. Log collectors
automatically upload reports and parse the firewall/ proxy traffic logs to see if there is a match with your
services in the Cloud App Catalog.",
    "score": 0,
    "lastSynced": "2022-10-02T08:27:46Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "implementationStatus": "Feature in place: false.",
    "scoreInPercentage": 0,
    "on": "false"
},
```


Recommendation: Manage in Cloud App Security to enable such feature

Configuration: Go to Microsoft Defender for Cloud Apps->Settings->Automatic log upload->Log collectors->Add log collector properly

6) *Case 6: Suspicious usage alert (should be improved to some extent now)*

Description: Activity policies help you monitor specific activities carried out by users, or follow unexpectedly high rates of certain types of activities. After you set an activity detection policy, it starts to generate alerts. Alerts are only generated on activities that occur after you create the policy.

API: https://graph.microsoft.com/beta/security/secureScores?$top=1

Result: Fail

Reason: no such policy in place

```
{
    "controlCategory": "Apps",
    "controlName": "McasCutomActivityPolicy",
    "description": "Activity policies help you monitor specific activities carried out by users,
or follow unexpectedly high rates of certain types of activities. After you set an activity detection policy, it
starts to generate alerts. Alerts are only generated on activities that occur after you create the policy.",
    "score": 0,
    "lastSynced": "2022-10-02T08:27:46Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "implementationStatus": "Policy in place: false.",
    "scoreInPercentage": 0,
    "on": "false"
},
```


Recommendation:
Manage in Cloud App Security to enable such policy
Configuration: Go to Microsoft Defender for Cloud Apps->Create activity policy

7) *Case 7: Identity and notify new and trending cloud apps*

Description: App discovery policies can notify you when new apps or abnormal usage is observed within your organization, based on traffic logs data.
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Fail
Reason: no such policy in place


```
{
    "controlCategory": "Apps",
    "controlName": "McasCloudAppNotification",
    "description": "App discovery policies can notify you when new apps or abnormal usage is
observed within your organization, based on traffic logs data.",
    "score": 0,
    "lastSynced": "2022-10-02T08:27:46Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "implementationStatus": "Policy in place: false.",
    "scoreInPercentage": 0,
    "on": "false"
},
```


Recommendation: Manage in Cloud App Security to enable such policy
Configuration: Go to Microsoft Defender for Cloud Apps->Create app discovery policy

8) *Case 8: Notify new OAuth applications (should be improved to some extent now)*

Description: OAuth app policies can help you manage app permission and notify you when a user or an admin consents to a new Open Authorization (OAuth) app. With this information, you can investigate which permissions each app requested and which users authorized them.
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Fail
Reason: no such feature in place


```
{
    "controlCategory": "Apps",
    "controlName": "McasOAuthAppNotification",
    "description": "OAuth app policies can help you manage app permission and notify you when a
user or an admin consents to a new Open Authorization (OAuth) app. With this information, you can investigate
which permissions each app requested and which users authorized them.",
    "score": 0,
    "lastSynced": "2022-10-02T08:27:46Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "implementationStatus": "Feature in place: false.",
    "scoreInPercentage": 0,
    "on": "false"
},
```


Recommendation: Manage in Cloud App Security to enable such feature
Configuration: Go to Microsoft Defender for Cloud Apps->Create OAuth app policy

9) *Case 9: Notify new OAuth applications (should be improved to some extent now)*

Description: OAuth app policies can help you manage app permission and notify you when a user or an admin consents to a new Open Authorization (OAuth) app. With this information, you can investigate which permissions each app requested and which users authorized them.

API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Fail
Reason: no such feature in place


```
{
    "controlCategory": "Apps",
    "controlName": "McasOAuthAppNotification",
    "description": "OAuth app policies can help you manage app permission and notify you when a
user or an admin consents to a new Open Authorization (OAuth) app. With this information, you can investigate
which permissions each app requested and which users authorized them.",
    "score": 0,
    "lastSynced": "2022-10-02T08:27:46Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "implementationStatus": "Feature in place: false.",
    "scoreInPercentage": 0,
    "on": "false"
},
```


Recommendation: Manage in Cloud App Security to enable such feature
Configuration: Go to Microsoft Defender for Cloud Apps->Create OAuth app policy

10) Case 10: Spam detection

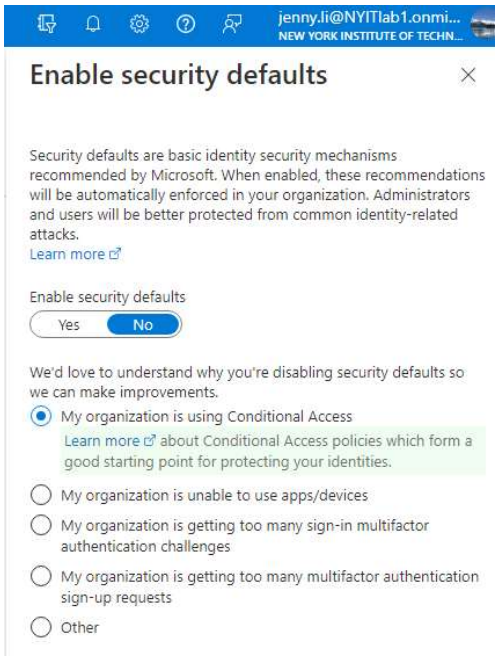
Description: Set the action that will be taken on high confidence spam detection.
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Fail
Reason: no such policy applied


```
{
    "controlCategory": "Apps",
    "controlName": "mdo_highconfidencespamaction",
    "description": "Set the action that will be taken on high confidence spam detection.",
    "score": 0,
    "lastSynced": "2022-10-02T09:31:42Z",
    "source": "ingestion",
    "IsApplicable": "true",
    "total": "7",
    "implementationStatus": "",
    "scoreInPercentage": 0,
    "noPolicies": "false",
    "mdoImplementationStatus": "{\"Policies\":[{\"PolicyName\":\"Default\",\"Status\":1,
\"UsersInPolicy\":7,\"MissingDomains\":null}],\"TotalUsers\":7}",
    "count": "0"
},
```


Recommendation: create such high confidence spam detection policy properly
Configuration: Go to Microsoft 365 Defender->manage Anti-spam policies

C. Identity Security

For testing purpose, disable security defaults to allow Conditional Access Policies.



1) *Case 1: One Admin (already modified, now only one global admin)*

- Security description: Having more than one global administrator helps if you are unable to fulfill the needs or obligations of your organization. It's important to have a delegate or an emergency account someone from your team can access if necessary. It also allows admins the ability to monitor each other for signs of a breach.
- API: https://graph.microsoft.com/beta/security/secureScores?$top=1
- Result: Fail
- Reason: You currently have 3 global admins.


```
{
    "controlCategory": "Identity",
    "controlName": "OneAdmin",
    "description": "Having more than one global administrator helps if you are unable to fulfill
the needs or obligations of your organization. It's important to have a delegate or an emergency account someone
from your team can access if necessary. It also allows admins the ability to monitor each other for signs of a
breach.",
    "score": 1,
    "implementationStatus": "You currently have 3 global admins.",
    "scoreInPercentage": 100,
    "controlState": "active",
    "IsApplicable": "true",
    "lastSynced": "2022-10-01T00:00:00Z",
    "IsEnforced": "false",
    "count": "3"
},
```


- Recommend: go to Microsoft 365 admin center, change permissions based on least privilege rule and reduce overlap privilege for users.

2) *Case 2: Block Legacy Authentication*

- Security description: Today, most compromising sign-in attempts come from legacy authentication. Older office clients such as Office 2010 don't support modern authentication and use legacy protocols such as IMAP, SMTP, and POP3. Legacy authentication does not support multifactor authentication (MFA). Even if an MFA policy is configured in your environment, bad actors can bypass these enforcements through legacy protocols.
- Prerequisite: Disable security default; create "Block legacy authentication" policy
- API: https://graph.microsoft.com/v1.0/identity/conditionalAccess/policies
- Result: Fail

- Reason: enabled, but No user is included for this policy. This means all users don't have legacy authentication blocked.
- Recommendation: Manage "Block legacy authentication" policy at Azure Active Directory->Security->Conditional Access Polices to include all users (may specify exclude condition if needed).

3) Case 3: Require MFA for regular sign-ins

- Description: Adding more authentication methods, such as the Microsoft Authenticator app or a phone number, increases the level of protection if one factor is compromised.
- API: https://graph.microsoft.com/v1.0/identity/conditionalAccess/policies
- Result: Pass
- Reason: enabled; All users are required MFA, except three users on purpose.

```
{
    "id": "1869ea52-4d8c-464e-ac7e-1697bf3d90a8",
    "displayName": "Require MFA for all sign-ins",
    "createdDateTime": "2022-09-03T04:23:21.4918712Z",
    "modifiedDateTime": "2022-10-01T21:44:43.1397667Z",
    "state": "enabled",
    "sessionControls": null,
    "conditions": {
        "userRiskLevels": [],
        "signInRiskLevels": [],
        "clientAppTypes": [ ···
        ],
        "servicePrincipalRiskLevels": [],
        "platforms": null,
        "locations": null,
        "devices": null,
        "clientApplications": null,
        "applications": { ···
        },
        "users": {
            "includeUsers": [
                "All"
            ],
            "excludeUsers": [
                "eae4bfc7-435e-4ced-9a4e-044fec11205f",
                "e0375dbc-ee6d-4b9b-92b2-502b428d74e8",
                "cba5642b-54d8-495b-a0fa-0db1eaec2490"
            ],
```


- Configuration: create “Require MFA for all sign-ins” conditional access policy at Azure AD
- Prerequisite: disable security default to enable conditional access policy

4) Case 4: MFA for Admin

Description: Requiring multifactor authentication (MFA) for administrative roles makes it harder for attackers to access accounts.

API: https://graph.microsoft.com/beta/security/secureScores?$top=1

Result: Fail

Reason: No MFA for Admins policy applied to admins


```
            {
                "controlCategory": "Identity",
                "controlName": "AdminMFAV2",
                "description": "<p>Requiring multifactor authentication (MFA) for administrative roles makes it
harder for attackers to access accounts. Administrative roles have higher permissions than typical users. If any of
those accounts are compromised, your entire organization is exposed. At a minimum, protect the following roles: 
</p><ul><li>Global administrator </li><li>Authentication administrator </li><li>Billing administrator 
</li><li>Conditional Access administrator </li><li>Exchange administrator </li><li>Helpdesk administrator&
nbsp;</li><li>Security administrator </li><li>SharePoint administrator </li><li>User administrator </
li></ul>",
                "score": 10,
                "lastSynced": "2022-10-02T00:00:00Z",
                "controlState": "active",
                "IsApplicable": "true",
                "total": "3",
                "IsEnforced": "false",
                "implementationStatus": "You have 0 out of 3 users with administrative roles that aren't
registered and protected with MFA.",
                "scoreInPercentage": 100,
                "count": "0"
            },
```


Recommendation: Go to Azure AD->Security->Conditional Access->Policies->new a policy to require MFA for admins and apply to all admins

Prerequisite: disable security default to enable conditional access policy

5) *Case 5: Require password change for high risky users*

- Prerequisite: Disable security default; create “Require password change” policy
- API: https://graph.microsoft.com/v1.0/identity/conditionalAccess/policies
- Result: Pass
- Reason: enabled; Under conditions “high” and “medium” risk level, all users are required password change, except one specific user on purpose.


```
{
    "id": "410c3585-c6db-4f19-96b2-44a277b9f7b5",
    "displayName": "Require password change",
    "createdDateTime": "2022-09-02T21:15:54.1464965Z",
    "modifiedDateTime": "2022-10-01T22:02:34.7778666Z",
    "state": "enabled",
    "sessionControls": null,
    "conditions": {
        "userRiskLevels": [
            "high",
            "medium"
        ],
        "signInRiskLevels": [],
        "clientAppTypes": [ ···
        ],
        "servicePrincipalRiskLevels": [],
        "platforms": null,
        "locations": null,
        "devices": null,
        "clientApplications": null,
        "applications": { ···
        },
        "users": {
            "includeUsers": [
                "All"
            ],
            "excludeUsers": [
                "eae4bfc7-435e-4ced-9a4e-044fec11205f"
            ],
```


6) *Case 6: Check Security Alerts*

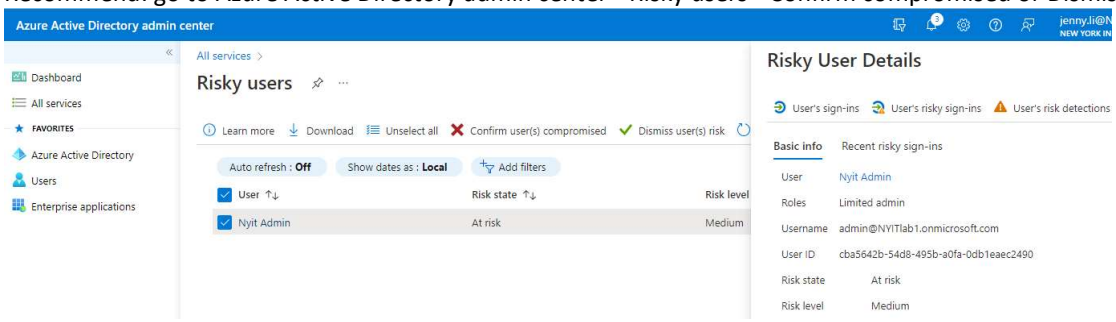
API: https://graph.microsoft.com/v1.0/security/alerts?$top=1
Result: Fail
Reason: has one sign-in alert

```
"value": [
    {
        "id": "e8fe7b2280e024bce1ac44679363564a90708c16fcb23c29df7823ae73c73603",
        "azureTenantId": "240d0a1a-ee75-4107-8bb7-880f2cdc67ad",
        "azureSubscriptionId": null,
        "riskScore": null,
        "tags": [],
        "activityGroupName": null,
        "assignedTo": null,
        "category": "UnfamiliarLocation",
        "closedDateTime": null,
        "comments": [],
        "confidence": null,
        "createdDateTime": "2022-09-09T01:52:04.594554Z",
        "description": "Sign-in with properties we have not seen recently for the given user",
        "detectionIds": [],
        "eventDateTime": "2022-09-09T01:52:04.594554Z",
        "feedback": null,
        "incidentIds": [],
        "lastEventDateTime": null,
        "lastModifiedDateTime": "2022-09-09T01:54:47.6407519Z",
        "recommendedActions": [],
        "severity": "low",
        "sourceMaterials": [],
        "status": "newAlert",
        "title": "Unfamiliar sign-in properties",
```


The alert is from user: admin


```
"userStates": [
    {
        "aadUserId": "cba5642b-54d8-495b-a0fa-0db1eaec2490",
        "accountName": "admin",
        "domainName": "NYITlab1.onmicrosoft.com",
        "emailRole": "unknown",
        "isVpn": null,
        "logonDateTime": "2022-09-09T01:52:04.594554Z",
        "logonId": null,
        "logonIp": "72.143.232.143",
        "logonLocation": "Toronto, Ontario, CA",
        "logonType": null,
        "onPremisesSecurityIdentifier": null,
        "riskScore": null,
        "userAccountType": null,
        "userPrincipalName": "admin@NYITlab1.onmicrosoft.com"
    }
```


Recommend: go to Azure Active Directory admin center->Risky users->Confirm compromised or Dismiss risk



7) *Case 7: Check if any users do not belong to anyone group*

Description: Attribute based security control; assign user(s) to specific group(s) based on the job attribute to manage permissions

Step1: Get all groups:

API: https://graph.microsoft.com/v1.0/groups

Group 1: NYIT Security IT; "id": "2c1178c1-15e4-46fc-aad5-3db719f67e66"
Group 2: NYIT IT Admin; "id": "9b005d59-33e0-43ca-a633-828ab73639d2"
Group 3: NYIT Development; "id": "cd7e26a6-ec7e-435e-82e1-c1ca56666bdd"
Step2: Get direct members of a group
API: https://graph.microsoft.com/v1.0/groups/{group-id}/members?$count=true
Group 1: NYIT Security IT; "id": "2c1178c1-15e4-46fc-aad5-3db719f67e66"
Users belong to this group:
Nyit Admin; "id": "cba5642b-54d8-495b-a0fa-0db1eaec2490"
Samuel Almeida; "id": "e0375dbc-ee6d-4b9b-92b2-502b428d74e8"
Jenny Li: "id": "eae4bfc7-435e-4ced-9a4e-044fec11205f"

Group 2: NYIT IT Admin; "id": "9b005d59-33e0-43ca-a633-828ab73639d2"
Users belong to this group:
Nyit Admin; "id": "cba5642b-54d8-495b-a0fa-0db1eaec2490"
Jenny Li: "id": "eae4bfc7-435e-4ced-9a4e-044fec11205f"

Group 2: NYIT IT Admin; "id": "9b005d59-33e0-43ca-a633-828ab73639d2"
Users belong to this group:
Alice Tester; "id": "1b848aa9-bd4e-4d12-b487-0980875e6f9c"
Tom Tester; "id": "3b5b865b-15bb-4043-b9fb-43ae2219eaf2"
Bob Tester: "id": "ae4d3b4d-54b7-4d58-b7d4-1b5d0413e925"

Step3: Get all users in
API: https://graph.microsoft.com/v1.0/users
Users: Nyit Admin, Alice Tester, Bob Tester, Jenny Li, Nyit Reader, Samuel Almeida, Tom Tester
Count all users: 7
API: https://graph.microsoft.com/v1.0/users/$count

Step4: Compare output from step2 and step3

Result: Fail
Reason: User “Nyit Reader” does not belong to any group
Recommendation: assign targeted users into specific group(s) at Azure Active Directory

8) Case 8: Password Age Policy

Description: Research has found that when periodic password resets are enforced, passwords become less secure. Users tend to pick a weaker password and vary it slightly for each reset. If a user creates a strong password (long, complex and without any pragmatic words present) it should remain just as strong in the future as it is today. It is Microsoft's official security position to not expire passwords periodically without a specific reason, and recommends that cloud-only tenants set the password policy to never expire.
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Pass
Reason: Your current policy is set to never let passwords expire.


```
{
    "controlCategory": "Identity",
    "controlName": "PWAgePolicyNew",
    "description": "Research has found that when periodic password resets are enforced, passwords
become less secure. Users tend to pick a weaker password and vary it slightly for each reset. If a user creates a
strong password (long, complex and without any pragmatic words present) it should remain just as strong in the
future as it is today. It is Microsoft's official security position to not expire passwords periodically without a
specific reason, and recommends that cloud-only tenants set the password policy to never expire.",
    "score": 8,
    "controlState": "active",
    "expiry": "2147483647",
    "scoreInPercentage": 100,
    "implementationStatus": "Your current policy is set to never let passwords expire.",
    "IsApplicable": "true",
    "lastSynced": "2022-10-01T00:00:00Z",
    "IsEnforced": "false"
},
```

9) *Case 9: User consent policy (should be improved now)*

Description: To reduce the risk of malicious applications attempting to trick users into granting them access to your organization's data, turn off user consent applications that are not published by a verified publisher. Only admin can consent applications.

API: https://graph.microsoft.com/beta/security/secureScores?$top=1

Result: Fail

Reason: You have no user consent policy in place.


```
{
    "controlCategory": "Identity",
    "controlName": "IntegratedApps",
    "description": "To reduce the risk of malicious applications attempting to trick users into
granting them access to your organization's data, we recommend that you allow user consent only for applications that
have been published by a verified publisher.",
    "score": 0,
    "lastSynced": "2022-10-02T00:00:00Z",
    "controlState": "active",
    "IsApplicable": "true",
    "IsEnforced": "false",
    "implementationStatus": "You have no user consent policy in place.",
    "scoreInPercentage": 0,
    "on": "false"
},
```


Recommendation: Create and apply user consent policy properly.

Configuration: Go to Microsoft 365 admin center->Org settings->Services->User consent to apps->turn off User consent to apps and Save

10) *Case 10: Self Service Password Reset (already enabled now)*

Description: With self-service password reset in Azure Active Directory, users no longer need to engage help desk to reset passwords. This feature works well with Azure AD dynamically banned passwords, which prevents easily guessable passwords from being used.

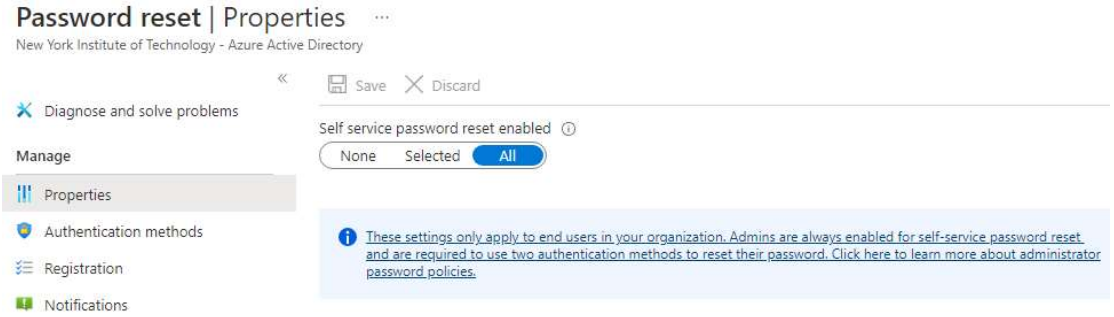
API: https://graph.microsoft.com/beta/security/secureScores?$top=1

Result: Fail

Reason: Self-service password reset is not applied for any users.


```
{
    "controlCategory": "Identity",
    "controlName": "SelfServicePasswordReset",
    "description": "With self-service password reset in Azure Active Directory, users no longer
need to engage help desk to reset passwords. This feature works well with Azure AD dynamically banned passwords,
which prevents easily guessable passwords from being used.",
    "score": 0,
    "total": "7",
    "controlState": "active",
    "scoreInPercentage": 0,
    "implementationStatus": "You have 7 of 7 users who don't have self-service password reset
enabled.",
    "IsApplicable": "true",
    "lastSynced": "2022-10-01T00:00:00Z",
    "count": "7",
    "IsEnforced": "false"
},
```


Recommendation: Enable “self-service password reset” at Azure Active Directory admin center



11) *Case 11: Sign in Risk Policy (state might be changed now)*

Description: Turning on the sign-in risk policy ensures that suspicious sign-ins are challenged for multifactor authentication (MFA).

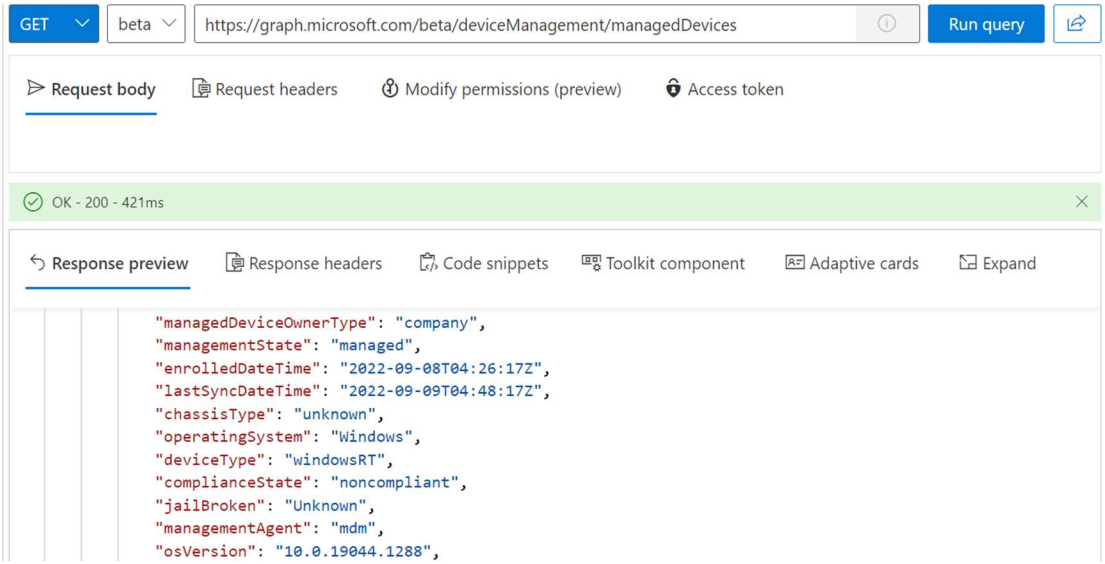
API: https://graph.microsoft.com/beta/security/secureScores?$top=1
Result: Fail
Reason: There is one user that doesn't have the sign-in risky policy turned on.


```
{
    "controlCategory": "Identity",
    "controlName": "SigninRiskPolicy",
    "description": "Turning on the sign-in risk policy ensures that suspicious sign-ins are
challenged for multifactor authentication (MFA).",
    "score": 6,
    "total": "7",
    "controlState": "active",
    "scoreInPercentage": 85.71,
    "implementationStatus": "You have 1 of 7 users that don't have the sign-in risky policy turned
on.",
    "IsApplicable": "true",
    "lastSynced": "2022-10-01T00:00:00Z",
    "count": "1",
    "IsEnforced": "false"
},
```

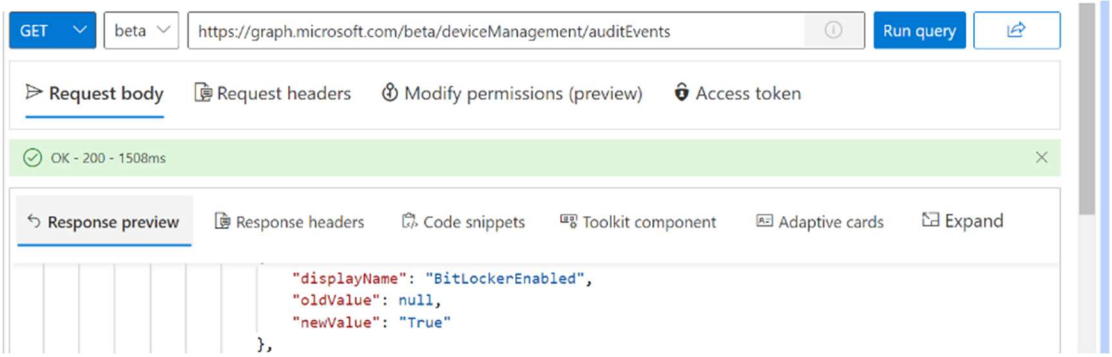

Recommendation: Apply sign-in risky policy for the target user at Azure Active Directory.

D. Device Test cases

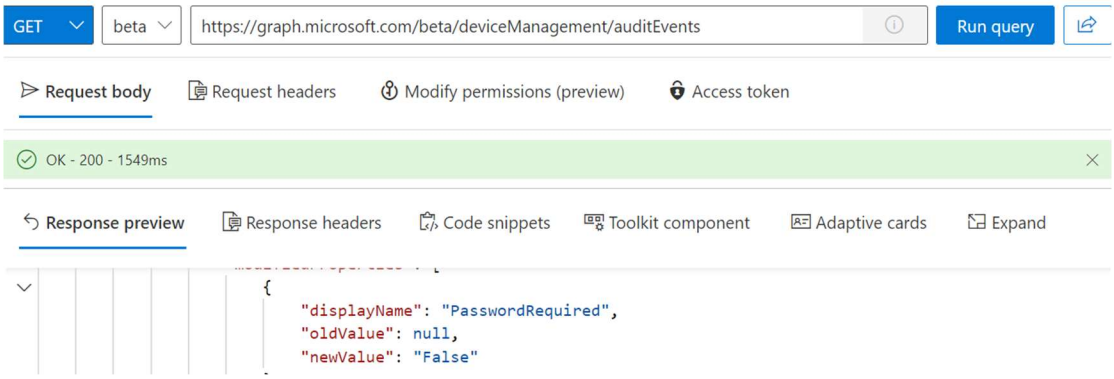
1) Case1 - Check if device is complaint or not based on the policies set in endpoint manager



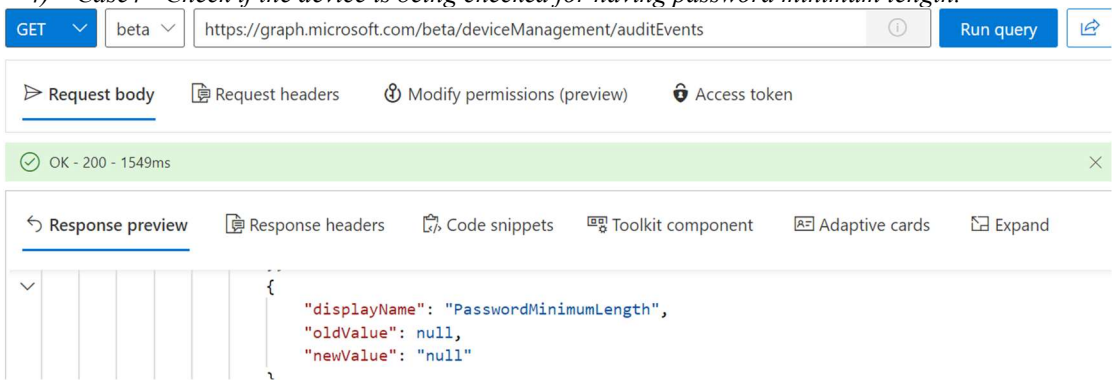
2) Case2 - Check if the device is checked for having bit locker enabled



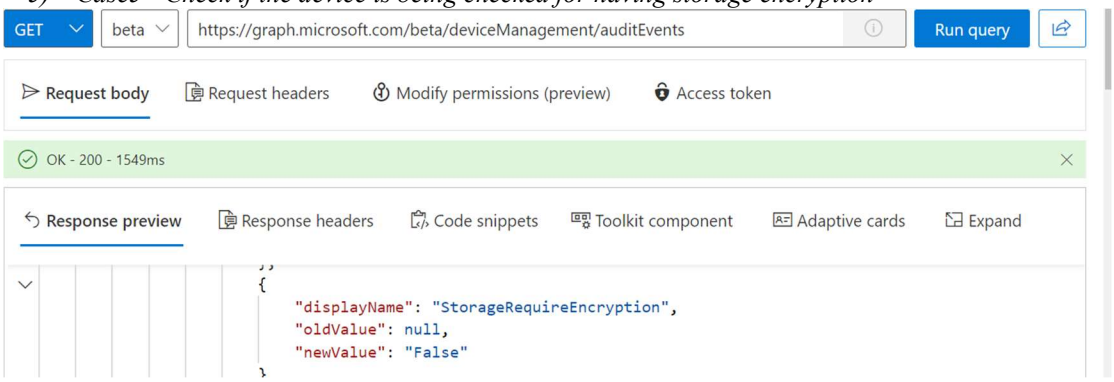
3) Case3 - Check if the device is being checked for having password required



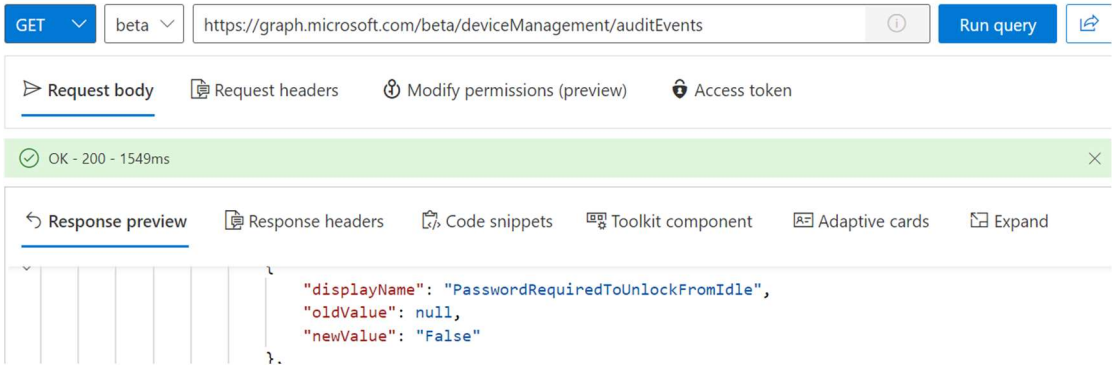
4) Case4 - Check if the device is being checked for having password minimum length.



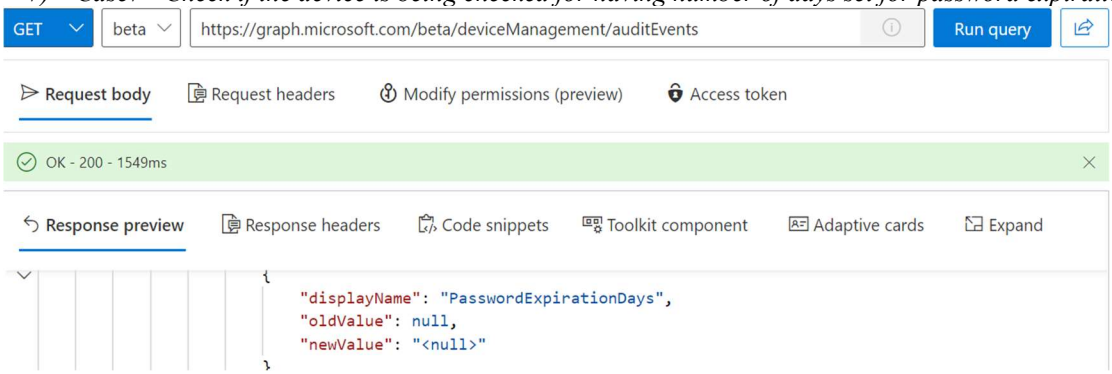
5) Case5 - Check if the device is being checked for having storage encryption



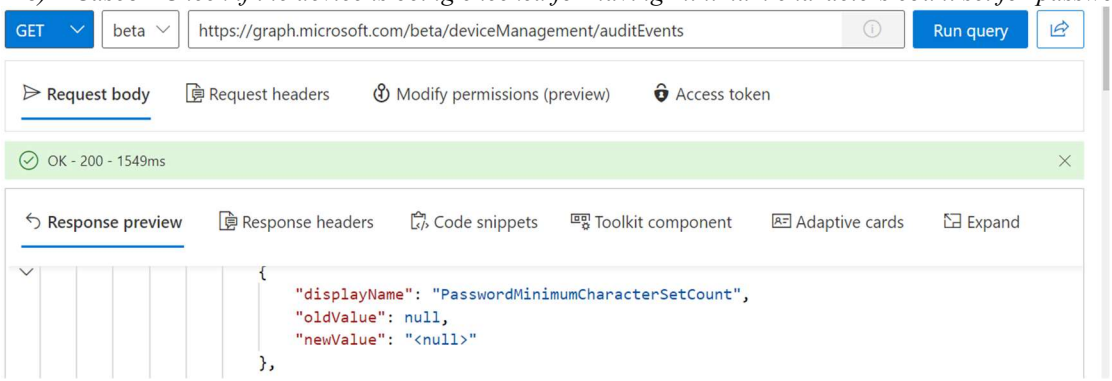
6) Case6 - Check if the device is being checked for requiring password to unlock from idle



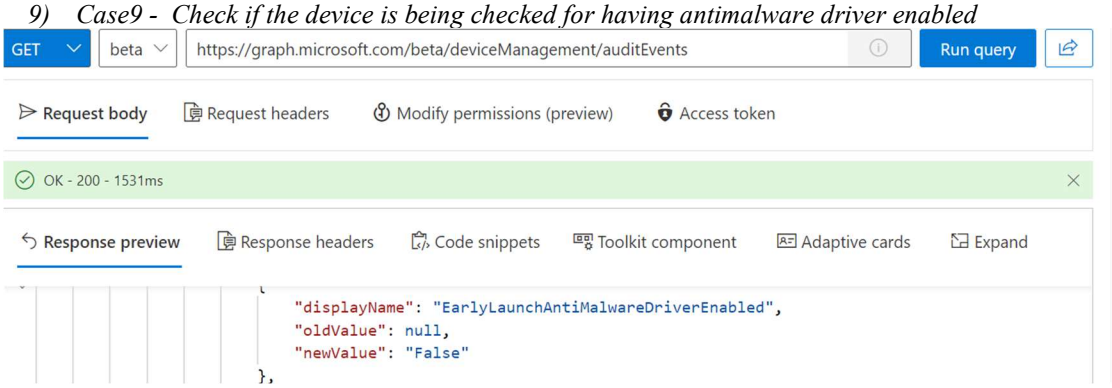
7) *Case7 - Check if the device is being checked for having number of days set for password expiration.*



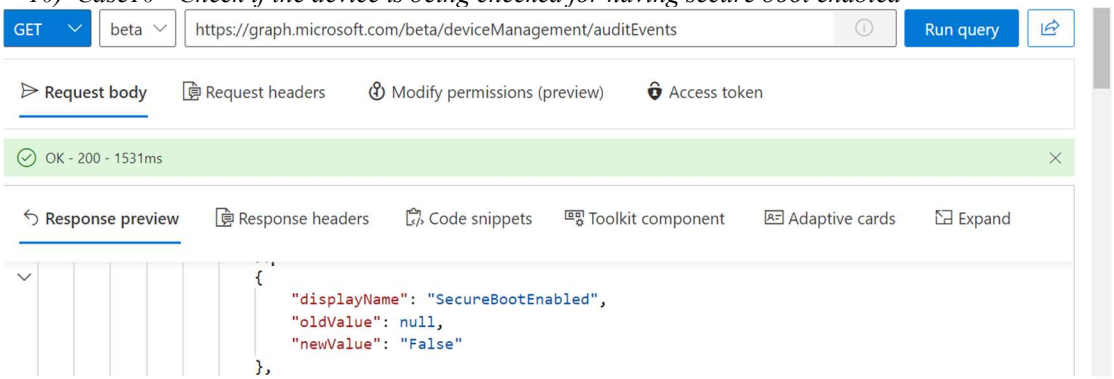
8) *Case8 - Check if the device is being checked for having minimum characters count set for password*



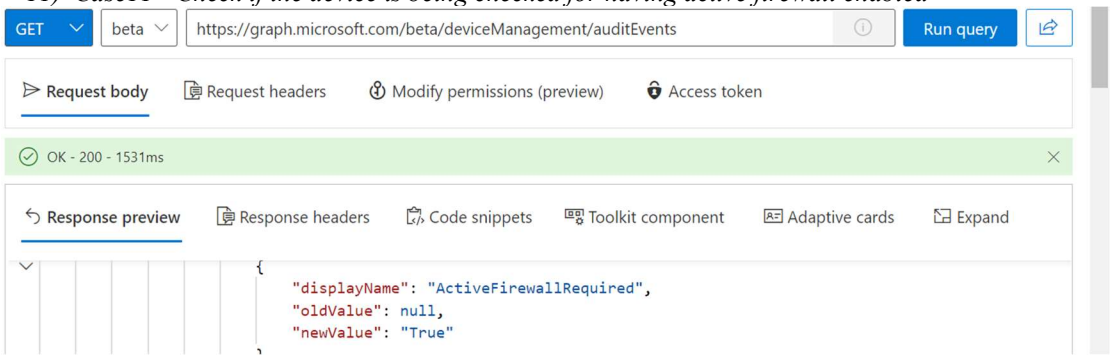
9) *Case9 - Check if the device is being checked for having antimalware driver enabled*



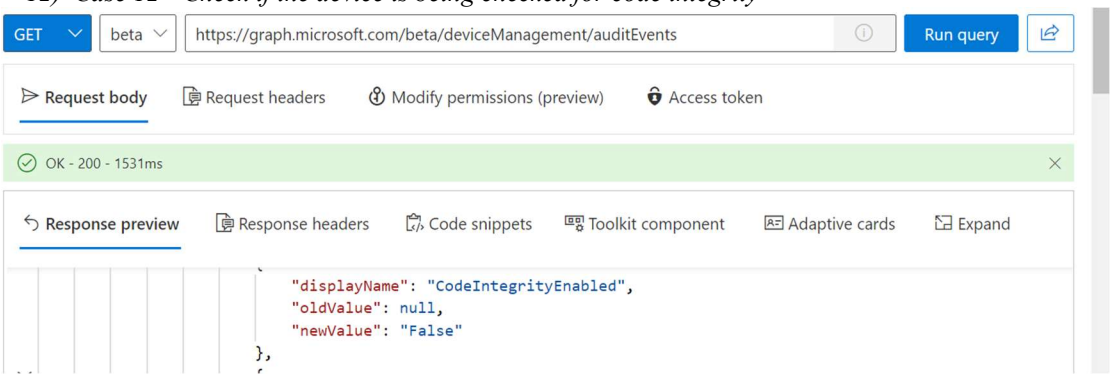
10) *Case10 - Check if the device is being checked for having secure boot enabled*



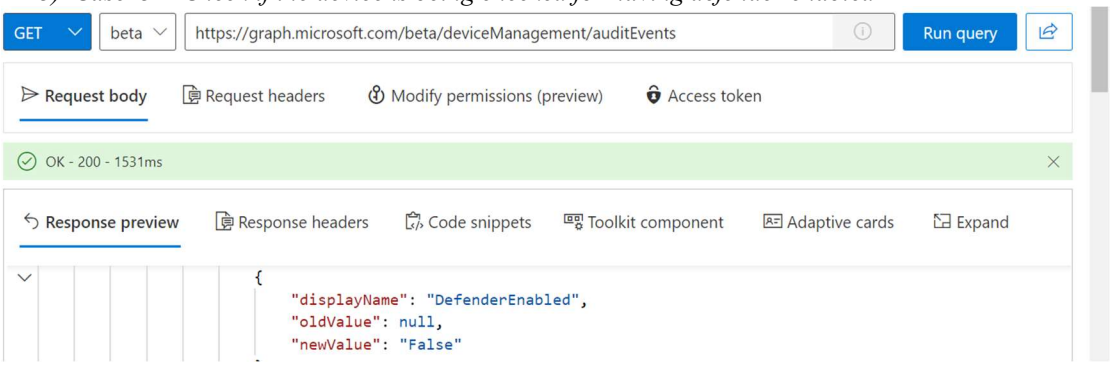
11) Case11 - Check if the device is being checked for having active firewall enabled



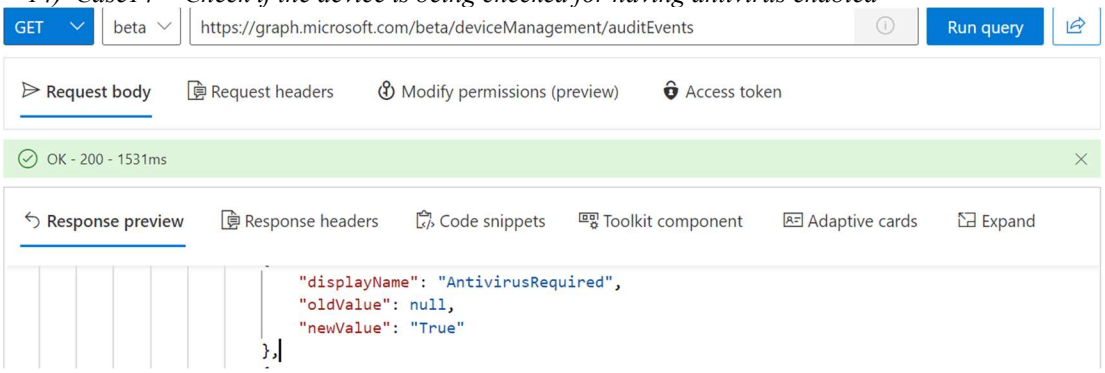
12) Case 12 - Check if the device is being checked for code integrity



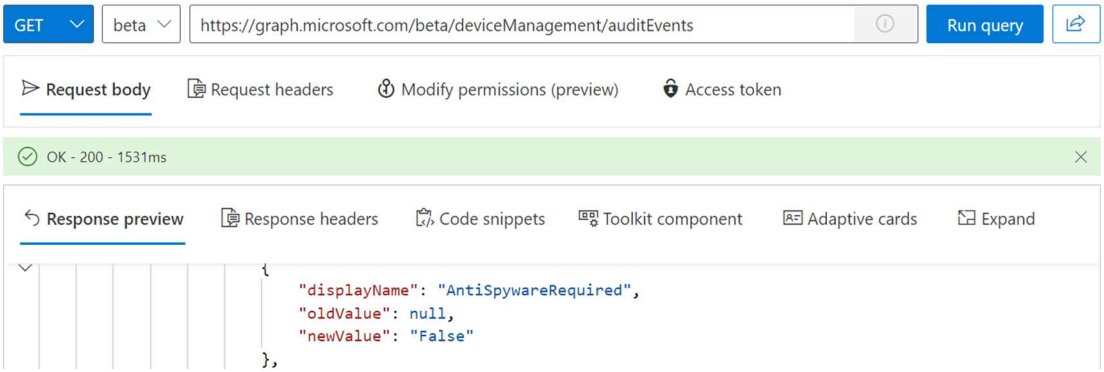
13) Case13 - Check if the device is being checked for having defender enabled



14) Case14 - Check if the device is being checked for having antivirus enabled



15) Case15 - Check if the device is being checked for having antispyware enabled



E. Considerations

1. Based on API output, narrow down the content, make a Pass/Fail decision
2. For Identity Security Case 7, need to organize content based on output from multiple APIs to make a decision.