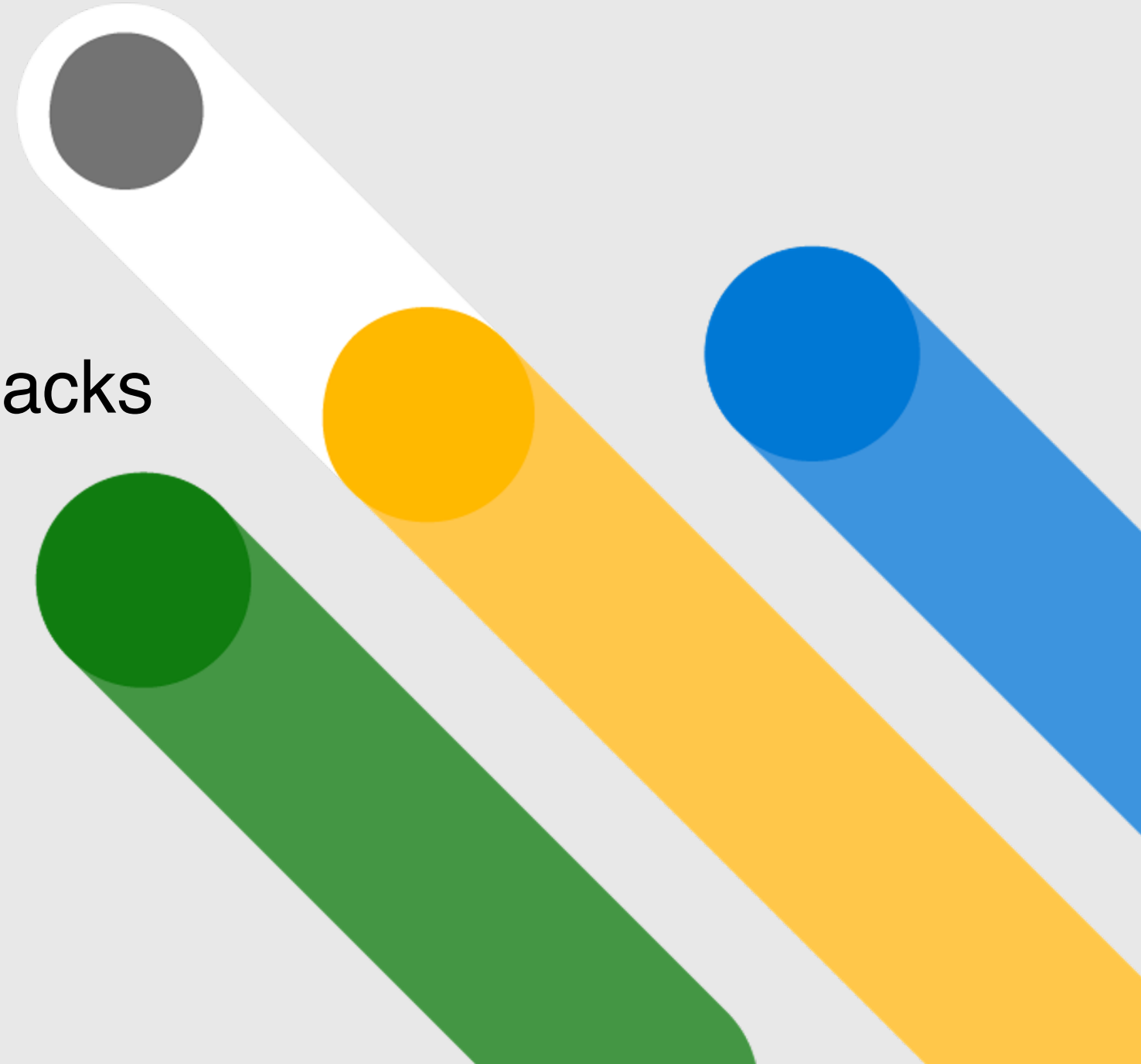
Microsoft Security

Graph API Mastery – Logs to Real World Attacks



About Us

- Over 10 years in Security
- Sr. Threat Intel Analyst @ Atlassian
- Previously worked in Symantec/Microsoft etc.
  in various CyberSec roles.
- “Food is bae”

- Over 10 years in Security
- Sr. Security Researcher @ Microsoft
- Previously worked in Symantec/Anaplan etc. in various CyberSec roles.
- “Gaming is an escape from reality”

Agenda

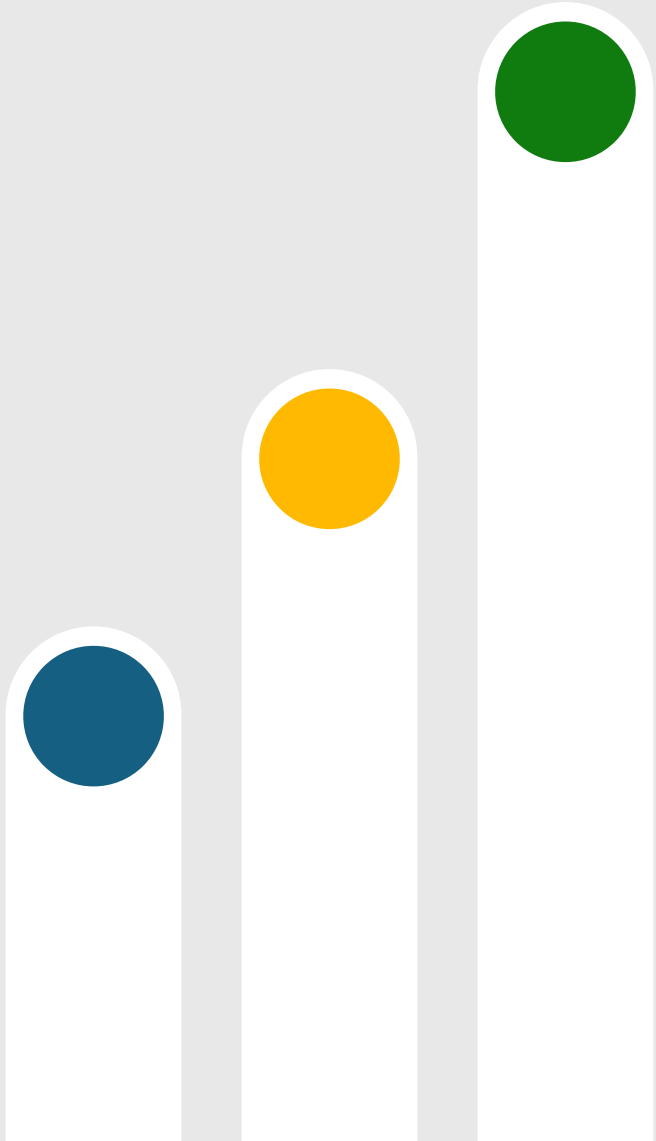
#1 ***Microsoft Graph API***

#2 ***Obtaining these logs***

#3 ***Key fields, Correlatable tables, Useful Functions***

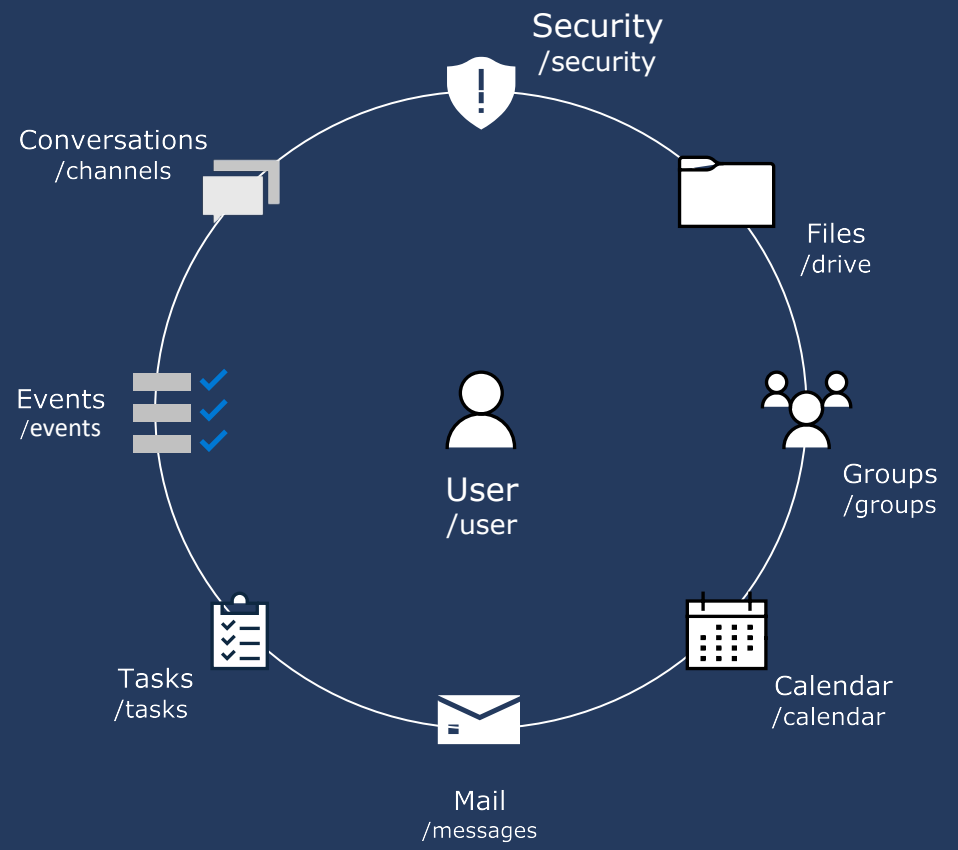
#4 ***Real-World Attack Scenarios***

#5 ***Auditing***



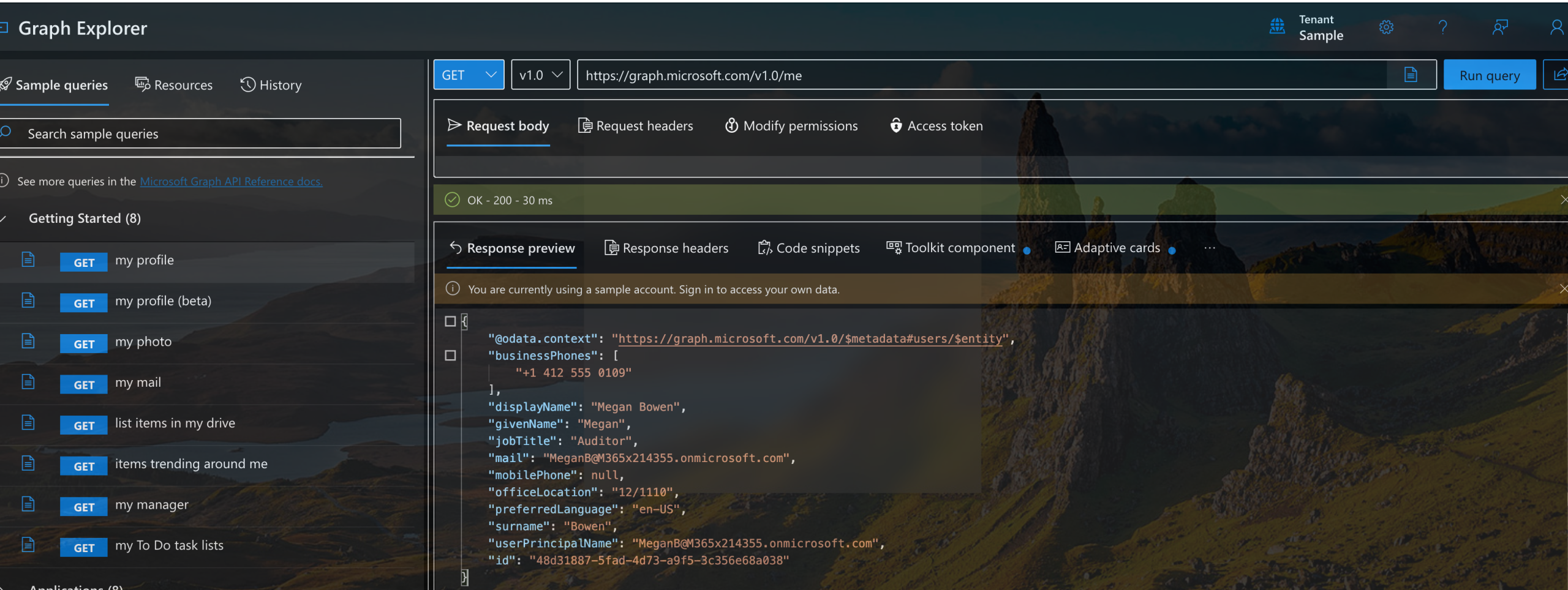
Microsoft Graph API

- Microsoft Graph is an unified API endpoint that provides access to Microsoft 365 services.
- Enables querying and interacting with data from Outlook, calendar, SharePoint, OneDrive, etc.
- Can be abused by threat actors for post-compromise activity.
- Detecting and responding to abuse of Graph API attacks is crucial.



Graph API Explorer

aka.ms/ge

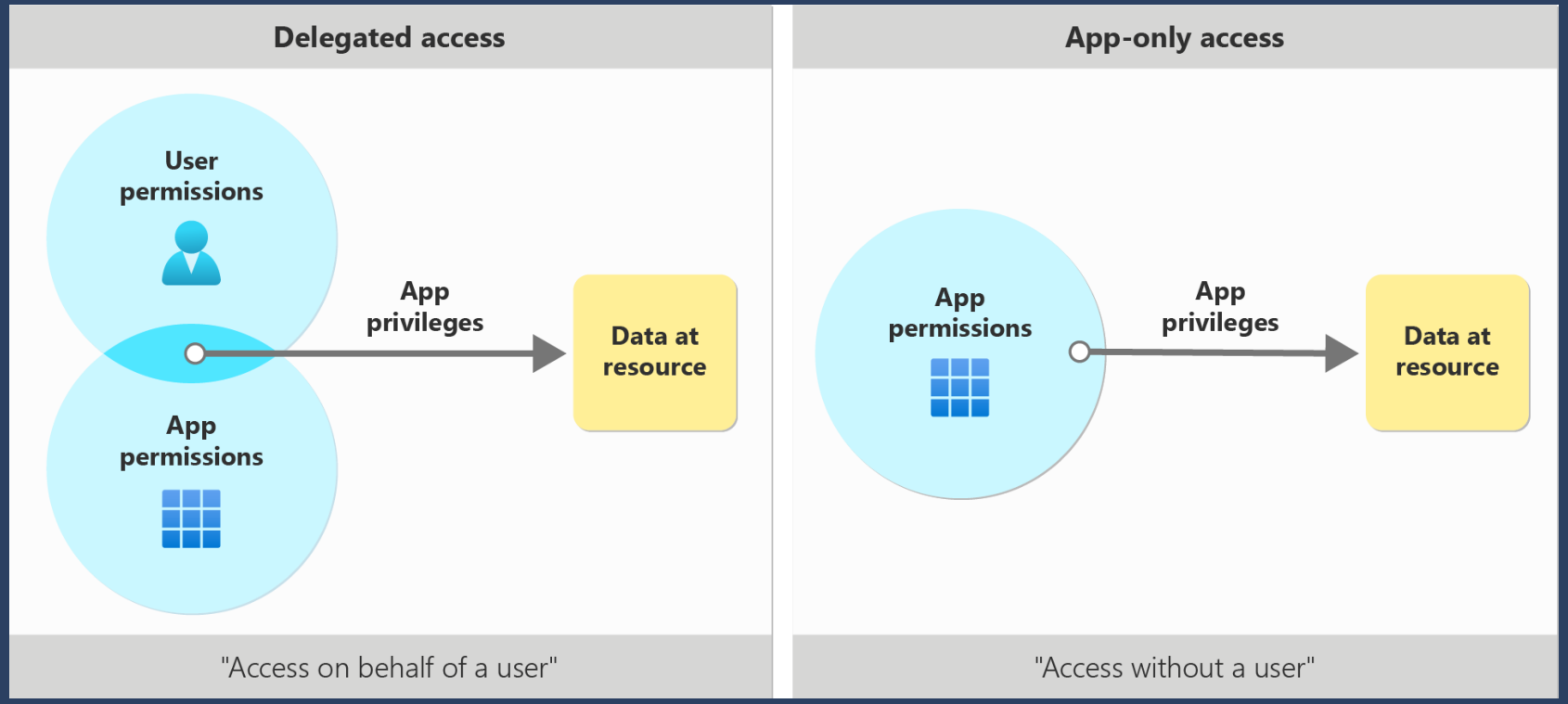


Authentication & Permissions

	Delegated permissions	Application permissions
User context	Requires a signed-in user	No user context needed
Consent	User consent (or admin on behalf of user)	Admin consent required
Scope	Limited to user's permissions	Broader, organization-wide scope
Typical use cases	Interactive applications (web, mobile, desktop)	Background services, daemons, administrative tools

- Graph API authentication is handled using OAuth protocol
- To access the Graph API, you need to have an application and OAuth tokens
- Two common access methods to access resources:
  1. Delegated access
     - An app acting on behalf of a signed-in user.
  2. App-only access
     - An app acting with its own identity.

Delegated vs. Application Permissions



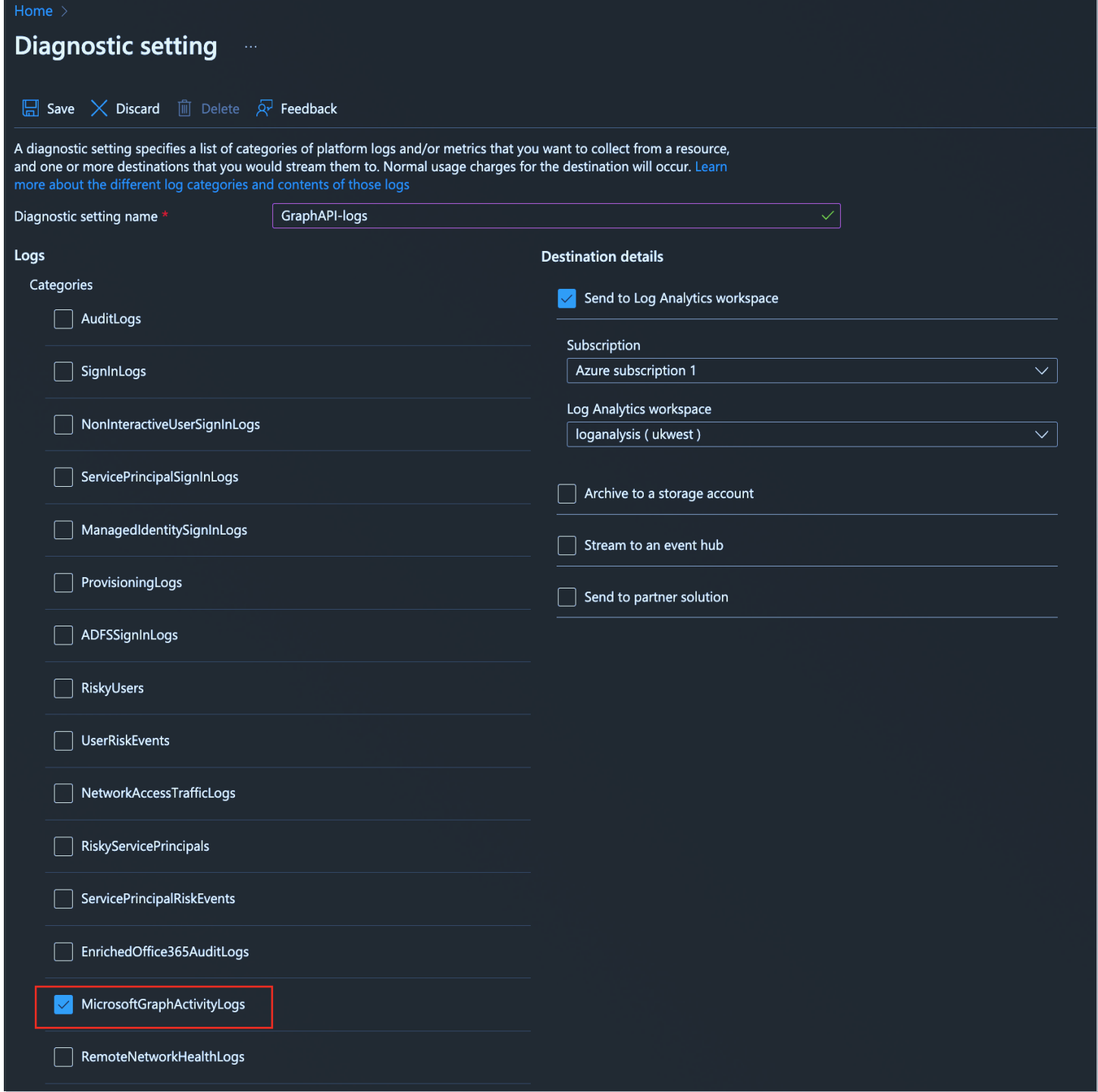
Obtain these logs

- Graph activity logs are not collected by default and must be enabled.
- Microsoft Entra ID P1 or P2 tenant license is a prerequisite to collecting these logs

Steps to collect these logs:

1. Navigate to Microsoft Entra ID on Azure Portal.
2. Go to Diagnostic Settings and add a new setting.
3. Enable MicrosoftGraphActivityLogs and choose a storage destination.

- Logs are crucial for analysis via Log Analytics or other SIEMs.



Key Fields in Graph API

Fields	Details
TenantId	The Log Analytics workspace ID.
Time Generated [UTC]	The date and time the request was received.
AppId	The identifier of the application
IPAddress	The IP address of the client from where the request occurred.
ServicePrincipalId	The identifier of the service principal making the request.
RequestId	The identifier representing the request.
RequestMethod	The HTTP method of the event.
ResponseStatusCode	The HTTP response status code for the event.
RequestUri	The URI of the request.
ResponseSizeBytes	The size of the response in bytes.
Roles	The roles in token claims

Useful Tips

Correlatable Tables

IdentityInfo

Information about the user

UserId == **AccountObjectId**

AadRiskyUser

Context on user risk details

UserId == **Id**

Useful KQL Functions

Geo IP information

Provides context on geo-location

| extend GeoIPInfo = geo_info_from_ip_address(IPAddress)

parse_url()

Provides an easy-to-read version

| extend ParsedURI = parse_url(RequestUri)

Real World ATT&CK Scenarios

#1 *Reconnaissance/Discovery*

#2 *Privilege Escalation*

#3 *Lateral Movement*

#4 *Collection*

#5 *Exfiltration*

Reconnaissance/Discovery

- We often encounter situations where reconnaissance tools are used to gather data about a tenant, finding ways to elevate privileges.
- To illustrate, let's explore open-source reconnaissance tools such as GraphRunner or AzureHound.

Query : The purpose of this query is to identify a surge in standard calls within a brief period that are characteristics of reconnaissance tools.

```
let calls = dynamic(["https://graph.microsoft.com/v1.0/users/<UUID>",
"https://graph.microsoft.com/v1.0/search/query",
"https://graph.microsoft.com/beta/policies/authorizationPolicy",
"https://graph.microsoft.com/v1.0/users",
"https://graph.microsoft.com/v1.0/groups","(...)",
"https://graph.microsoft.com/beta/users/<UUID>/roleManagement/
directorytransitiveRoleAssignments",
"https://graph.microsoft.com/v1.0/roleManagement/directory/roleDefinitions/<UUID>",
"https://graph.microsoft.com/beta/roleManagement/directory/estimateAccess",
"https://graph.microsoft.com/beta/users"]);
MicrosoftGraphActivityLogs
| where ResponseStatusCode == '200'
| summarize MinTime=min(TimeGenerated), MaxTime=max(TimeGenerated),
UniqueCalls=dcount(GeneralizedUri), CallsMade=count(),
UserAgents=make_set(UserAgent) by IPAddress, bin(TimeGenerated, 2m), Id, ObjectType
| where datetime_diff('second', MaxTime, MinTime) < 100 and
((UniqueCalls >= 3 and CallsMade >= 40) or CallsMade > 100)
```

IPAddress	UserAgents
82.19.125.164	["TestHound/v0.0.0"]
IPAddress	82.19.125.164
TimeGenerated [UTC]	2024-08-22T15:52:00Z
Id	412ebaf5-c8a3-4ba9-961e-45facb4589f4
ObjectType	User
MinTime [UTC]	2024-08-22T15:52:13.4113708Z
MaxTime [UTC]	2024-08-22T15:52:19.7444184Z
UniqueCalls	14
CallsMade	1063
UserAgents	["TestHound/v0.0.0"]
82.19.125.164	["azurehound/v0.0.0"]
IPAddress	82.19.125.164
TimeGenerated [UTC]	2024-08-22T15:00:00Z
Id	bb2d7a3e-60c6-46eb-942c-22ccbffc86e0
ObjectType	User
MinTime [UTC]	2024-08-22T15:01:48.2955004Z
MaxTime [UTC]	2024-08-22T15:01:53.0648604Z
UniqueCalls	14
CallsMade	671
UserAgents	["azurehound/v0.0.0"]

Adjust the filter attributes as needed. You can add more Microsoft Graph API requests to the calls array.

Privilege Escalation

- After an initial compromise, having a specific set of privileges in an environment can allow for the assignment of higher privileges to other compromised accounts.
- A threat actor compromised a service principal of an Entra application with the "**RoleManagement.ReadWrite.Directory**" role.
- Using these permissions, they assigned the "**Global Administrator**" role to another compromised user identity

AccountName	AccountObjectId	AssignedRoles
hacker	4fb5a3e3-e86d-42ff-b8d9-51b4e6dccc46	[]

AccountName	AccountObjectId	AssignedRoles
hacker	4fb5a3e3-e86d-42ff-b8d9-51b4e6dccc46	["Global Administrator"]

Query : The query below detects role changes in Microsoft Graph ActivityLogsactivity logs.

```
MicrosoftGraphActivityLogs
| where RequestUri has_all ("https://graph.microsoft.com/", "/directoryRoles/", "members/$ref")
| where RequestMethod == "POST"
| where ResponseStatusCode in ("204")
| extend Role = tostring(split(RequestUri, "/")[-3]) //Role can be looked up in Auditlogs
| project  TimeGenerated, IPAddress, RequestUri, ResponseStatusCode, Role, UserAgent, AppId
```

IPAddress	54.86.50.139
RequestUri	https://graph.microsoft.com/v1.0/directoryRoles/ee557baa-af23-4ee5-a72a-343db6554bf5/members/$ref
ResponseStatusCode	204
Role	ee557baa-af23-4ee5-a72a-343db6554bf5
UserAgent	PostmanRuntime/7.41.2
AppId	ce184bd1-2ccd-4ac7-8890-68e020c0bcf1

Investigators should examine this result using AuditLogs or other available logs to provide further context and to distinguish between legitimate and unauthorized activity.

Lateral Movement

In this scenario, a rogue application was created, and a phishing link was sent to a user. After the user's token was captured through a phish, the actor used delegated permissions to send emails to other users by using the “sendMail” API function.

Query: Identifies the use of sendMail in the URI and lists all emails sent via GraphAPI.

```
MicrosoftGraphActivityLogs
| where TimeGenerated >= ago(30d)
| where ResponseStatusCode == "202"
| where RequestUri contains "/sendMail"
| extend EmailSentFrom = tostring(parse_url(RequestUri).Path).substring(1).split("/")[-2]
| extend Id = iff(isempty(UserId), ServicePrincipalId, UserId)
| extend Type = iff(isempty(UserId), "ServicePrincipal", "User")
| extend JoinKey = case(Type == "ServicePrincipal", EmailSentFrom, Type == "User", UserId, "")
| join kind=leftouter (IdentityInfo | extend JoinKey = AccountObjectId) on JoinKey
| project-reorder TimeGenerated, Type, AppId, MailAddress, RequestUri, ResponseStatusCode, UserAgent, AccountUPN
```

TimeGenerated [UTC]	2024-08-12T13:26:32.6474102Z
Type	ServicePrincipal
AppId	ce184bd1-2ccd-4ac7-8890-68e020c0bcf1
MailAddress	test@MngTest.onmicrosoft.com
RequestUri	https://graph.microsoft.com/v1.0/users/34699233-67e9-45a1-a5ff-06087fa1ec74/sendMail
ResponseStatusCode	202
UserAgent	PostmanRuntime/7.40.0
AccountUPN	test@MngTest.onmicrosoft.com

Query: Reviewing the AppID and service principal can help verify if the applications are allowed to send emails. This query summarizes the emails sent by service principals in the past 30 days.

```
MicrosoftGraphActivityLogs
| where TimeGenerated >= ago(30d)
| where ResponseStatusCode == "202"
| where RequestUri contains "/sendMail"
| extend EmailSentFrom =  tostring(split(RequestUri, "/")[-2])
| extend Id = iff(isempty(UserId), ServicePrincipalId, UserId)
| extend Type = iff(isempty(UserId), "ServicePrincipal", "User")
| where Type == "ServicePrincipal"
| join kind=leftouter IdentityInfo on $left.EmailSentFrom == $right.AccountObjectId
| summarize count() by AppId, AccountUPN, UserAgent
```

AppId	ce184bd1-2ccd-4ac7-8890-68e020c0bcf1
AccountUPN	test@MngTest.onmicrosoft.com
UserAgent	PostmanRuntime/7.40.0
count_	3

Collection

- We often handle cases where a threat actor targets a specific user's mailbox by abusing delegated permissions or accesses multiple users' emails through applications access with broader permissions.
- To illustrate, let's go over a scenario where the threat actor abused an application with excessive permissions, allowing them to gain unauthorized access to the mailboxes of users.
- **Query:** The query below reveals statistics about the applications or users used for reading emails, along with the number of unique mailboxes accessed and their respective timeframes.

```
MicrosoftGraphActivityLogs
| where TimeGenerated >= (30d)
| where RequestMethod == "GET"
| where RequestUri has_all ("https://graph.microsoft.com", "/users/", "/messages")
| where ResponseStatusCode == "200"
| extend Id = iff(isempty(UserId), ServicePrincipalId, UserId)
| extend ObjectType = iff(isempty(UserId), "ServicePrincipal", "User")
| extend MailboxTargetUPN = tostring(extract_all( @'https://graph.microsoft.com/v.../users/([^/]*)/mailFolders/
', RequestUri)[0]) //Parses the AccountUPN
| extend UserGuid= tostring(extract_all( @'*.(\b[0-9a-fA-F]{8}(?:-[0-9a-fA-F]{4}){3}\b-[0-9a-fA-F]{12}).*',
RequestUri)[0]) //Parses the object-ID of an targeted identity
| join kind=leftouter (IdentityInfo |  where TimeGenerated > ago(30d) | summarize arg_max(TimeGenerated, *) by
AccountObjectId | project TargetUPN=AccountUPN, AccountObjectId) on $left.UserGuid==$right.AccountObjectId
| extend TargetUPN = coalesce(TargetUPN, MailboxTargetUPN)
| summarize MinTime=min(TimeGenerated), MaxTime=max(TimeGenerated), MailBoxAccessCount=dcount(TargetUPN),
Targets=make_set(TargetUPN) by AppId, ObjectType, Id
```

Results | Chart

AppId	ObjectType	Id	MinTime [UTC]	MaxTime [UTC]	Targets	MailBoxAccessCount
ce184bd1-2ccd-4ac7-8890-68e020c0bcf1	ServicePrincipal	09b92560-bd66-44d...	8/20/2024, 12:51:04.674 PM	8/20/2024, 12:51:04.674 PM	["test@MngTest.onmicrosoft.com"]	1

Exfiltration

- Actors abuse an application with Files Read/Write and Sites Read/Write permissions. These excessive permissions allow them to search through users' OneDrive and SharePoint files to download confidential documents.

Query: This is a good starting point for investigating Microsoft Graph API calls related to download activities. Analyze the UserAgent and AppID to determine if whether these activities are expected in your environment.

```
MicrosoftGraphActivityLogs
| where TimeGenerated >= (30d)
| where RequestMethod == "GET"
| where ResponseStatusCode in ("302")
| where RequestUri matches regex @"https://graph\.microsoft\.com/.*/items/.*/content"
    and RequestUri matches regex @"/drives?/.*"
| project  TimeGenerated, ResponseStatusCode, RequestMethod, IPAddress, UserAgent, RequestUri, AppId
```

Analyze the UserAgent and AppID to determine if whether these activities are expected in your environment.

ResponseStatusCode	302
RequestMethod	GET
IPAddress	54.86.50.139
UserAgent	PostmanRuntime/7.41.2
RequestUri	https://graph.microsoft.com/v1.0/sites/mngtest.sharepoint.com,9f65f433-fcf2-4e9f-ae76-e7a0e9812cec,1092efee-d807-4bb9-a620-c42127b421ad/drive/items/01UDZKGPKXG7HQWAVSIRHJLPPX2LZZ4LVO/content
AppId	ce184bd1-2ccd-4ac7-8890-68e020c0bcf1

Resolving the Item ID of a downloaded item can be cumbersome, but correlating CloudApp events offers additional context for the download activity.

Auditing Graph API Usage

- Regular audits of Entra applications using the Microsoft Graph API can reveal excessive permissions or unexpected access, indicating possible service principal compromise.
- Auditing helps create a safe list of approved applications with excessive permissions.
- Continuous monitoring can then be applied to detect new applications with high privileges, ensuring timely identification of potential security threats.

Query: Identifies Entra applications with high-impact permissions.

```
let PrivilegeAbuse = datatable (Type: string, Permission: string, Privilege: string, Reason:
string) [
    "Application","Mail.ReadWrite","High","BroadImpact",
    "Application","Mail.Read","High","Collection",
    "Application","Contacts","High","Phishing",
    "Application","MailboxSettings","High","Phishing",
    //"(.....)",
    "Application","User.ReadWrite.All","High","BroadImpact",
    "Application","User.ManageCreds.All","High","BroadImpact",
    "Application","AppRoleAssignment.ReadWrite.All","High","PrivEscalation"
];
MicrosoftGraphActivityLogs
| where TimeGenerated between (ago(7d) .. now())
| extend ObjectType = iff(isempty(UserId), "ServicePrincipal", "User")
| where ObjectType == 'ServicePrincipal'
| extend RolesTemp = split(Roles, " ")
| mv-expand RolesTemp
| where RolesTemp has_any (( PrivilegeAbuse | distinct Permission ))
| extend Role = tostring(RolesTemp)
| summarize Calls=count(), MinTime=min(TimeGenerated), MaxTime=max(TimeGenerated) by AppId, Role
```

AppId	Role	Calls
ce184bd1-2ccd-4ac7-8890-68e020c0bcf1	Files.ReadWrite.All	9
ce184bd1-2ccd-4ac7-8890-68e020c0bcf1	Files.Read.All	9
c9a559d2-7aab-4f13-a6ed-e7e9c52aec87	Files.ReadWrite.All	1
00000005-0000-0ff1-ce00-000000000000	Files.ReadWrite.All	3
4787c7ff-7cea-43db-8d0d-919f15c6354b	MailboxSettings.Read	4
ce184bd1-2ccd-4ac7-8890-68e020c0bcf1	User.ReadWrite.All	19
ce184bd1-2ccd-4ac7-8890-68e020c0bcf1	Mail.ReadWrite	23
ce184bd1-2ccd-4ac7-8890-68e020c0bcf1	Mail.Read	23
9ea1ad79-fdb6-4f9a-8bc3-2b70f96e34c7	People.Read.All	2
fc03f97a-9db0-4627-a216-ec98ce54e018	MailboxSettings.Read	66
cc15fd57-2c6c-4117-a88c-83b1d56b4bbe	EduRoster.ReadWrite.All	36
cc15fd57-2c6c-4117-a88c-83b1d56b4bbe	Group.ReadWrite.All	36
ab3be6b7-f5df-413d-ac2d-abf1e3fd9c0b	Member.Read.Hidden	8
00000002-0000-0ff1-ce00-000000000000	Domain.ReadWrite.All	85

Summary

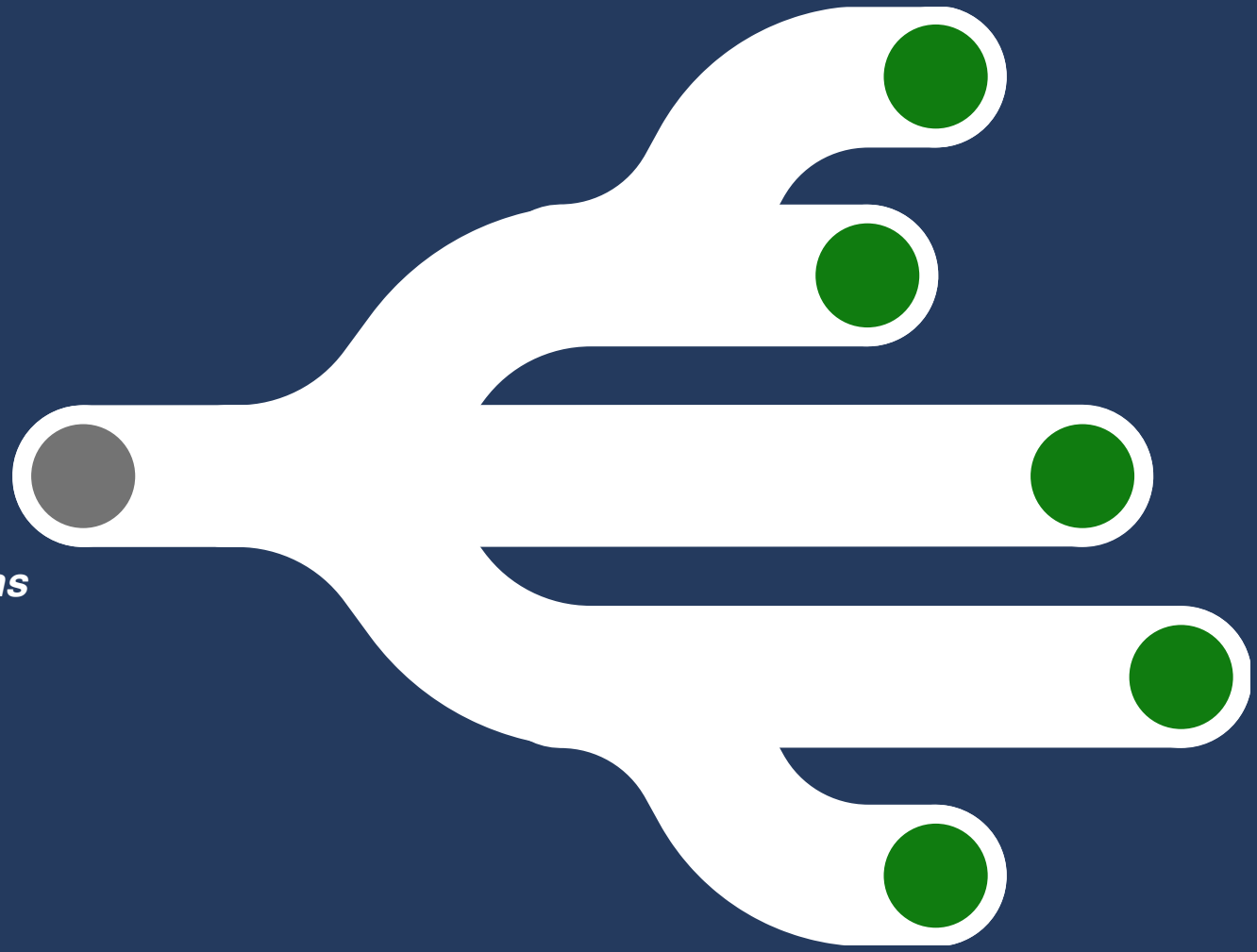
#1 ***Microsoft Graph API***

#2 ***Obtaining these logs***

#3 ***Key fields, Correlatable tables, Useful Functions***

#4 ***Real-World Attack Scenarios***

#5 ***Auditing***



Insights from the community

- A Defenders Guide to GraphRunner — Part I
- A Defenders Guide to GraphRunner — Part II
- Detect threats using Microsoft Graph activity logs - Part 1
- Detect threats using Microsoft Graph activity logs - Part 2
- Abuse of OAuth app | Microsoft Security Blog

Thank you !

To learn more about Microsoft Incident Response, visit aka.ms/microsoftIR

Hunting with Microsoft Graph activity logs