



Sign-in logs and auditing of Managed Identities and Service Principals

Recently, Microsoft added new categories for sign-in logs which finally included non-interactive, managed or service principals in Azure AD. In this blog post I will describe the configuration steps to forward the new collections to Azure Sentinel, some considerations from my first tests and a few examples of correlation to activity logs and potentially Azure Sentinel analytic rules.

September 19, 2020 • 10 minute read

Note: Currently there's no official statement or documentation around this feature. Please wait for Microsoft's documentation and consider the log volume estimation before enabling the new collection in your production environment.

Limited security options and controls

Most organization are using non-interactive sign-ins for automation tasks or deployment tasks in (Azure) DevOps pipelines. Until now, there are some missing security controls for managed identities or and service principals (comparing to user accounts):

- No restriction of access by conditions (similar to the “condition” and “control” approach in Conditional Access)
- Limited options of alternate authentication methods to existing key- or certificate-based credentials
 - Managed Identities in Azure aren’t supported by every service or scenario. Therefore fallback to Service principals with keys or secrets is sometimes necessary.
- No sign-in log events to audit the authentication requests (and sign-in attempts)

Particularly the last point was one of the most requested features. This is not only necessary for troubleshooting, it’s also important from aspects of security, compliance and comprehensive (privileged) identity audit.

This seems to be also the reason for several Azure Feedback/User voice posts that exists:

- [Azure AD sign-in activity log should record and show sign-ins from service principal](#)
- [Make SPN \(non-interactive\) login events logged and available](#)

Configuration of the new sign-in categories

In the past, Microsoft allows already to route Azure AD audit and sign-ins logs to storage accounts, event hubs or Log Analytics workspaces. This is well described in the Microsoft Docs article “[Azure AD logs in Azure Monitor](#)”. At the time of writing this article, the following new categories are not covered by Microsoft’s documentation. However, there are already visible and configurable in the diagnostic settings of the Azure AD portal (Azure Portal > Azure AD Blade > Diagnostic Settings):

On this page

Limited security options and controls

Configuration of the new sign-in categories

Data schema of the new sign-in logs

General considerations and notes from my tests

Scenarios and query examples

Managed Identity and Azure KeyVault Access

Service Principal and Azure Deployment

Service Principal und Logic Apps

Service Principal (certificate-based authentication) and Microsoft Graph API

Analytic rules of service principal sign-ins

Sign-in attempts from service principals with expired/wrong keys

Detection of generate new key even service principal sign-ins are successfully

Diagnostics settings

 Refresh  Provide feedback

Diagnostic settings are used to configure streaming export of platform logs and metrics for a resource to the destination storage account.

Diagnostics settings

Name	Storage account
AzureSentinel_secops-la	-


[+ Add diagnostic setting](#)

Click 'Add Diagnostic setting' above to configure the collection of the following data:

- AuditLogs
- SignInLogs
- NonInteractiveUserSignInLogs
- ServicePrincipalSignInLogs
- ManagedIdentitySignInLogs



The last three (new) categories can be configured as already described in the Docs article [“Create diagnostic settings to send platform logs and metrics to different destinations”](#).

You’ll find the following tables (and entries) in your destination resource after the first records of the respective logs was forwarded from Azure Monitor successfully. In the following examples I’ve used Azure Sentinel / Log Analytics workspace.



Azure Sentinel | Logs

Search (Cmd+ /)

New Query 1*  

General

Overview

Logs

News & guides

Threat management

Incidents

Workbooks

Hunting

Notebooks (Preview)

Entity behavior analytics (Preview)

Configuration

Data connectors

Tables


Queries

Filter

Search

Group by: Solution Filters: not selected

Favorites


You can add favorites by clicking on the  icon


Azure Sentinel


BehaviorAnalyticsInsights

DNS Analytics (Preview)

LogManagement

 AADManagedIdentitySignInLogs

 AADNonInteractiveUserSignInLogs

 AADServicePrincipalSignInLogs

Data schema of the new sign-in logs

Even if there are no further documentation, the data schema of the new sign-in logs has been already documented in the [Azure Monitor Logs reference](#). *Note: Name of the collection in the diagnostic settings and the name of the table are not equal.*

- NonInteracticeUserSignInLogs → [AADNonInteractiveUserSignInLogs](#)
- ServicePrincipalSignInLogs → [AADServicePrincipalSignInLogs](#)
- ManagedIdentitySignInLogs → [AADManagedIdentitySignInLogs](#)

Tip: It's interesting to compare the data schema of all three non-interactive logs with the existing (user) sign-in log. Most field names are identical (CorrelationId) but some others (as expected) are not included (User risk- oder Conditional Access details, client/device information). Sign-in log schema of users is described in the following Microsoft Docs article: [Interpret the Azure AD sign-in logs schema in Azure Monitor](#)

General considerations and notes from my tests

- Check the pre-requisites and already known requirements concerning [Azure AD licensing](#)
- No indication seems to be available if service principal authentication attempt was made by client secret or certificate
- No direct correlation between sign-in and activity log entry (as in the most other cases in Azure logging)
- Be aware of the differences between application, object and service principal ID
 - Application and ServicePrincipal ID are included in the AADServicePrincipalSignInLogs
- ~~Currently only user sign-in logs are visible in the Azure Portal~~ UPDATE (Sept. 22th): Categories are also visible in the “Sign-ins” overview of the Azure AD blade.
- Table entries, size and size per entry should be actively monitored and considered in your test environment before enabling the new categories in your production. This also helps to calculate the estimated log volume.
 - I’m confident that Microsoft will update the [existing documentation about cost considerations soon](#).
 - I’ve used the “[Workspace Usage report](#)” during my early tests in the Azure Sentinel playground environment. The latest version of the workbook is [available from GitHub](#). It’s super-helpful to get insights of table entries and sizes but also about latency.

Table Name	↑↓	Table Entries	↑↓	Table Size	↑↓	Size per Entry	↑↓
SignInLogs		<div></div>	3.06K	<div></div>	16.73	<div></div>	5.6KiB
AADNonInteractiveUserSignInLogs		<div></div>	5.365K	<div></div>	1.153	<div></div>	225.3B
AADServicePrincipalSignInLogs		<div></div>	43	<div></div>	24.42	<div></div>	581.56B
AADManagedIdentitySignInLogs		<div></div>	30	<div></div>	14.96	<div></div>	510.9B

Scenarios and query examples

In the following scenarios, I want to give some examples of KQL queries to build a correlation between sign-in events and activity events.

Managed Identity and Azure KeyVault Access

Scenario:

Access from (system-assigned) managed identity of an Azure VM to Azure KeyVault (for reading a secret value of the vault).

I have used the Azure PowerShell module and including the parameter “Identity” for sign-in. Afterwards I’ve tried to access the vault with the cmdlet “Get-AzureKeyVaultSecret”.

Logs:

All Managed Identities sign-ins are located in the “AADManagedIdentitySignInLogs” table of the Log Analytics workspace. In the first query I’m looking for events of the system-assigned identity of the virtual machine:

```
AADManagedIdentitySignInLogs
| where ServicePrincipalName == <VM or application name>
| project TimeGenerated, Category, ResultType, CorrelationId, ServicePrincipalId, ResourceDisplayName
```

As we can see in the screenshot, two events will be displayed because of using the Az-Connect cmdlet and sending the request to the Azure KeyVault:

	<input type="checkbox"/> TimeGenerated [Amsterda...	<input type="checkbox"/> Category	<input type="checkbox"/> ResultType	<input type="checkbox"/> CorrelationId	<input type="checkbox"/> ServicePrincipalId	<input type="checkbox"/> ResourceDisplayName	<input type="checkbox"/> IPAddress	<input type="checkbox"/> LocationDetails
>	<input type="checkbox"/> 9/18/2020, 12:25:20.772 PM	ManagedIdentitySignInLogs	0	14fad4c1-8714-48...	56865b98-4385-...	Windows Azure Service Management API	40.113.32.32	("city":"Dublin",
>	<input type="checkbox"/> 9/18/2020, 12:25:33.832 PM	ManagedIdentitySignInLogs	0	2f3b9ecf-9fb4-488...	56865b98-4385-...	Azure Key Vault	40.113.32.32	("city":"Dublin",

Sign-in log entry of the resource “Azure KeyVault” includes a ResourceIdentity which is a fixed application ID of this service. Unfortunately it can not be used to find out which KeyVault instance in the Azure environment was be accessed. More details about how and when the key vault was accessed is available in the operational insights / diagnostics logs of the service.

This resource-level logs includes the identity from the token that was presented in the REST API request to the vault. As prerequisite for our query, we need the ServicePrincipalId of the Managed Identity which is displayed in the result of the previous sign-in event or can be displayed from the Get-AzADServicePrincipal cmdlet.

Tip: Further details are described in the Microsoft docs article “[View the service principal of a managed identity using PowerShell](#)”.

```
AzureDiagnostics
| where identity_claim_http_schemas_microsoft_com_identity_claims_objectidentifier_g ==
| project TimeGenerated, Category, CorrelationId, Resource, ResourceType, OperationName,
```

AuditEvent with detailed information about the resource operation should be displayed:

<input type="checkbox"/>	TimeGenerated [Amsterd...	Category	Correlatio...	Resource	ResourceType	OperationName	ResultType	CallerIPAddress
> <input type="checkbox"/>	9/18/2020, 12:25:33.891 PM	AuditEvent	26820e79-8...	IDENTITY-KVA	VAULTS	SecretGet	Success	10.1.0.4

Note: In this sample, the CallerIPAddress will be shown as private IP address because of using a private endpoint for accessing Azure KeyVault.

Service Principal and Azure Deployment

Scenario:

Access from Azure DevOps CD pipeline to deploy SQL database via Azure Resource Manager (ARM) API endpoint. Assigned service connection to the deployment pipeline uses service principal and a client secret.

Logs:

The following query shows filtered service principal sign-ins by the Resource Identity of “Windows Azure Service Management API” and the ServicePrincipalId (ServicePrincipalName would have been also possible):

```
AADServicePrincipalSignInLogs
| where ServicePrincipalId == <ServicePrincipalId>
| where ResourceIdentity == "797f4846-ba00-4fd7-ba43-dac1f8f63013"
| project TimeGenerated, Category, ResultType, CorrelationId, ServicePrincipalId, Resour
```

Four different sign-in requests are visible in the logs and seems to be used from the CD pipeline during the deployment:

<input type="checkbox"/>	TimeGenerated [Amster...	Category	ResultType	CorrelationId	ServicePrincipalId	ResourceDisplayName	IPAddress	LocationDetails
> <input type="checkbox"/>	9/18/2020, 5:18:16.087 PM	ServicePrincipalSignInLogs	0	ad70d4a9-60a5...	9daeb307-ba3e-41bf...	Windows Azure Service Management API	13.79.17.252	("city":"Dublin","state":"Du
> <input type="checkbox"/>	9/18/2020, 5:17:21.047 PM	ServicePrincipalSignInLogs	0	97de9295-ce32...	9daeb307-ba3e-41bf...	Windows Azure Service Management API	52.236.147.236	("city":"Amsterdam","state
> <input type="checkbox"/>	9/18/2020, 5:17:20.733 PM	ServicePrincipalSignInLogs	0	a6a7ab18-71c8-...	9daeb307-ba3e-41bf...	Windows Azure Service Management API	52.236.147.236	("city":"Amsterdam","state
> <input type="checkbox"/>	9/18/2020, 5:17:20.622 PM	ServicePrincipalSignInLogs	0	fd4cdd3e-067e...	9daeb307-ba3e-41bf...	Windows Azure Service Management API	52.236.147.236	("city":"Amsterdam","state

Afterwards I’ve started a query to find all Azure Activity logs related to this service principal:

```
AzureActivity | where Caller == <ServicePrincipalId>
| project TimeGenerated, Type, CorrelationId, OperationNameValue, ActivityStatusValue, C
```

ARM operations will be audited in details, as you can see in the following screenshot (even if successful activities will be displayed only):

<input type="checkbox"/>	TimeGenerated [Amster...	Type	CorrelationId	OperationNameValue	ActivityStat...	Caller	CallerIpAddress	ResourceProviderValue
>	<input type="checkbox"/>	9/18/2020, 5:18:16.614 PM	AzureActivity	112a58c8-273...	MICROSOFT.RESOURCES/SUBSCRIPTIONS/RESOURCEGROU...	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:18:17.383 PM	AzureActivity	a0bdf7be-149...	MICROSOFT.RESOURCES/DEPLOYMENTS/VALIDATE/ACTION	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:19:06.814 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.SQL/SERVERS/WRITE	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:19:10.165 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.SQL/SERVERS/FIREWALLRULES/WRITE	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:19:56.257 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.SQL/SERVERS/DATABASES/WRITE	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:20:01.346 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.SQL/SERVERS/DATABASES/TRANSPARENTDAT...	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:20:01.801 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.RESOURCES/DEPLOYMENTS/WRITE	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:28:22.759 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.AUTHORIZATION/POLICIES/AUDIT/ACTION	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:28:22.759 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.AUTHORIZATION/POLICIES/AUDIT/NOTEXISTS...	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:28:22.779 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.AUTHORIZATION/POLICIES/AUDIT/ACTION	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:28:22.779 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.AUTHORIZATION/POLICIES/AUDIT/NOTEXISTS...	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:28:22.779 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.AUTHORIZATION/POLICIES/AUDIT/ACTION	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:28:22.844 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.AUTHORIZATION/POLICIES/AUDIT/NOTEXISTS...	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:28:22.844 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.AUTHORIZATION/POLICIES/AUDIT/ACTION	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:28:22.854 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.AUTHORIZATION/POLICIES/AUDIT/NOTEXISTS...	Success	9daeb307-...	13.79.17.252
>	<input type="checkbox"/>	9/18/2020, 5:28:22.854 PM	AzureActivity	0c714f2b-23c5...	MICROSOFT.AUTHORIZATION/POLICIES/AUDIT/ACTION	Success	9daeb307-...	13.79.17.252

Note: I was able to find activities from one of the four service principal sign-ins. The other 3 sign-ins from a single IP address (Azure data center in “West Europe”) are without any activity log entry.

Service Principal und Logic Apps

Scenario:

Logic app is using a service principal for service / API connection. In this sample you will see an [Azure Sentinel playbook to sending enriched alerts](#) from Azure Sentinel to an Event Hub. Service principal is assigned to get information from the workspace via Log Analytics API.

Logs:

All Service Principal sign-ins are located in the “AADManagedIdentitySignInLogs” table. In this example I will use the “ServicePrincipalName” for filtering sign-in events:

```
AADServicePrincipalSignInLogs
| where ServicePrincipalName == "Azure Sentinel Alerts Evidence SIEM"
| project TimeGenerated, Category, CorrelationId, ServicePrincipalId, ResourceDisplayNam
```

Similar to the previous use case, the “ResourceIdentity” contains only the fixed Application ID of LogAnalytics API. It gives us no further details about the certain Logic App that was used:

<input type="checkbox"/>	TimeGenerated [Lo...	Category	Result...	CorrelationId	Appld	ResourceDisplayName	IPAddress	LocationDetails
>	<input type="checkbox"/>	2.9.2020, 16:57:18.466	ServicePrincipalSignInLogs	0	b844f334-d2e...	d13a2c8d-5f98...	Log Analytics API	52.166.78.89

However, in this use case we have another challenge then before. I was not able to find any relation between the Azure Logic App and the used service principal in the resource-level logs. So far as I can see, the timestamp and type of resource (based on the ResourceIdentity we know it is a Logic App) are the only indications to find the related resource activity log entry.

<input type="checkbox"/>	TimeGenerated [Local Time]	Stable	OperationName	Resource
>	<input type="checkbox"/>	2.9.2020, 17:09:52.034	AzureDiagnostics	Microsoft.Logic/workflows/workflowRunStarted
>	<input type="checkbox"/>	2.9.2020, 17:09:52.040	AzureDiagnostics	Microsoft.Logic/workflows/workflowTriggerStarted
>	<input type="checkbox"/>	2.9.2020, 17:09:52.114	AzureDiagnostics	Microsoft.Logic/workflows/workflowTriggerCompleted
>	<input type="checkbox"/>	2.9.2020, 17:09:52.638	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionStarted
>	<input type="checkbox"/>	2.9.2020, 17:09:52.675	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionCompleted
>	<input type="checkbox"/>	2.9.2020, 17:09:52.685	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionStarted
>	<input type="checkbox"/>	2.9.2020, 17:09:52.695	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionCompleted
>	<input type="checkbox"/>	2.9.2020, 17:09:52.702	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionStarted
>	<input type="checkbox"/>	2.9.2020, 17:09:52.747	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionCompleted
>	<input type="checkbox"/>	2.9.2020, 17:09:52.755	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionStarted
>	<input type="checkbox"/>	2.9.2020, 17:09:53.388	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionCompleted
>	<input type="checkbox"/>	2.9.2020, 17:09:53.399	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionStarted
>	<input type="checkbox"/>	2.9.2020, 17:09:53.446	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionCompleted
>	<input type="checkbox"/>	2.9.2020, 17:09:53.453	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionStarted
>	<input type="checkbox"/>	2.9.2020, 17:09:53.758	AzureDiagnostics	Microsoft.Logic/workflows/workflowActionCompleted
>	<input type="checkbox"/>	2.9.2020, 17:09:53.776	AzureDiagnostics	Microsoft.Logic/workflows/workflowRunCompleted

Although there are detailed information about the Logic App workflow in the diagnostic logs, but any indications to use the service principal is missing (e.g. Service Principal ID or caller IP address).

SourceSystem	Azure
ResourceId	/PROVIDERS/MICROSOFT.LOGIC/WORKFLOWS/GET-S
OperationName	Microsoft.Logic/workflows/workflowTriggerStarted
Category	WorkflowRuntime
Resource	WHEN_A_RESPONSE_TO_AN_AZURE_SENTINEL_ALERT_IS_TRIGGERED
ResourceGroup	
ResourceProvider	MICROSOFT.LOGIC
Level	Information
Type	AzureDiagnostics
SubscriptionId	
_ResourceId	/providers/microsoft.logic/workflows/get-sentinelalertsevidence/runs/0
ResourceType	WORKFLOWS/RUNS/TRIGGERS
status_s	Succeeded
startTime_t [UTC]	2020-09-02T15:09:51.993Z
workflowId_s	/PROVIDERS/MICROSOFT.LOGIC/WORKFLOWS/GET-S
resource_location_s	westeurope
resource_workflowId_g	59e9dc53-03c8-4982-a341-02e1e72c4f04
resource_resourceGroupName_s	
resource_subscriptionId_g	
resource_runId_s	08586025474934701472082286799CU101
resource_workflowName_s	Get-SentinelAlertsEvidence
_schema_s	2016-06-01
correlation_clientTrackingId_s	
resource_triggerName_s	When_a_response_to_an_Azure_Sentinel_alert_is_triggered
tags_LogicAppsCategory_s	security

All security-related logs of a Logic App can be filtered by the following query:

```
AzureDiagnostics
| where resource_workflowName_s == <LogicAppName> and tags_LogicAppsCategory_s == "security"
```

Service Principal (certificate-based authentication) and Microsoft Graph API

Scenario:

REST API calls to Microsoft Graph API for automation of Azure AD or Microsoft 365 related administrative tasks. This example shows the management of Conditional Access policies as part of an automation job and certificate-based authentication to the Microsoft Graph. In this case I’ve used a Graph API operation to delete a Conditional Access Policy.

Logs:

In this example, we are looking for all sign-in activities to the Microsoft Graph:

```
AADServicePrincipalSignInLogs
| where ResourceDisplayName == "Microsoft Graph"
```

The sign-in attempt of my Graph API call will be shows as follows:

<input type="checkbox"/>	TimeGenerated [Amste...	Category	CorrelationId	ServicePrincipalId	ResourceDisplayName	IPAddress	Location
> <input type="checkbox"/>	9/18/2020, 4:24:39.819 PM	ServicePrincipalSignInLogs	3493883e-6f...	656dd442-b41d-4...	Microsoft Graph		DE

Now we are able to find the related operational tasks from the Azure AD Audit Log by filtering the ServicePrincipalId:

```
AuditLogs
| where InitiatedBy.app.servicePrincipalId == <ServicePrincipalId>
| extend TargetResource = tostring(TargetResources[0].id), InitiatedBy = tostring(InitiatedBy.displayName), Result = tostring(Result)
| project TimeGenerated, Category, AADOperationType, CorrelationId, Result, InitiatedBy, TargetResource
```

Details of the operation including Objectid of the deleted policy are visible in the audit log entries:

<input type="checkbox"/>	TimeGenerated [Amste...	Category	AADO...	CorrelationId	Result	InitiatedBy	TargetResource	AdditionalDetails
> <input type="checkbox"/>	9/18/2020, 4:24:40.181 PM	Policy	Delete	c79b0200-389...	success	656dd442-b...	2193bc33-ca13-...	[{"key":"User-Agent","value":"Microsoft Azure Graph Client...
> <input type="checkbox"/>	9/18/2020, 4:24:40.299 PM	Policy	Update	8c6aa6b9-63c...	success	656dd442-b...	bdcc93a4-e714-...	[{"key":"User-Agent","value":"Microsoft Azure Graph Client...

Tip: Azure AD audit log is not showing the source IP address if modification was initiated by a service principal. Correlation between Audit and Service Principal sign-in log allows to discover the IP address now.

Analytic rules of service principal sign-ins

Service principal sign-in logs gives us many opportunities to build simple but also complex analytic rules. In the next section I want to give two different examples that was part of my early tests. Advanced scenarios will be part of my next work such as correlation between IP address from the service principal sign-in and suspicious IP addresses from Microsoft's Threat intelligence.

Sign-in attempts from service principals with expired/wrong keys

Rule logic:

Sign-in logs of service principals includes the ResultType which give us an indication if the sign-in was successfully. There could be many reasons for sign-in failures (ResultType is not equal "0") such as wrong client secrets or an expired certificate. An overview of error codes are listed in the Microsoft Docs about "[Azure AD Authentication and authorization error codes](#)".

The following query checks for failure sign-in events of service principals and summarize the events by service principal, resource (target), result type (error) and IP address.

Rule query:

```
let timeframe = 1d;
let threshold = 1;

AADServicePrincipalSignInLogs
| where TimeGenerated >= ago(timeframe)
| where ResultType != "0"
| summarize StartTimeUtc = min(TimeGenerated), EndTimeUtc = max(TimeGenerated), count() as applicationCount, applicationSet = makeset(ResourceDisplayName) by ServicePrincipalName, IPAddress, ResultType
| where applicationCount >= threshold
| extend timestamp = StartTimeUtc, AccountCustomEntity = ServicePrincipalName, IPCustomEntity = IPAddress
```

Detection of generate new key even service principal sign-ins are successfully

Rule logic:

This analytic rules have a look on audit events of updating certificates or secrets on service principals in a specific time range. Alert will be triggered if a service principal has successfully sign-in events within a period of time before changing credentials. This could be a suspicious event that someone generates credentials without any obvious reasons (service principal exists and is able to sign-in).

<https://github.com/Cloud-Architekt/azuresentinel/blob/master/ResetMFAAuthCredByAdmin.kusto>

Rule query:

```
let timeRange = 1d;
let maxTimeBetweenSecretMgmtandSigninInMinutes=3*24*60; // per Default max. difference between secret management and sign-in

AuditLogs
| where TimeGenerated >= ago(timeRange)
| where OperationName == "Update application - Certificates and secrets management"
| extend ModifiedCred = TimeGenerated, ServicePrincipalName = tostring(TargetResources[0].id),
| extend objectId = tostring(TargetResources[0].id),
  ServicePrincipalName = tolower(tostring(TargetResources[0].displayName)), ActorIP = tolower(tostring(TargetResources[0].ipAddress)),
| project ServicePrincipalName, objectId, ActorIP, ModifiedCredTime = TimeGenerated, CorrelationId = CorrelationId
| join kind= leftouter (
  AADServicePrincipalSignInLogs
  | where datetime_add('minute',maxTimeBetweenSecretMgmtandSigninInMinutes,TimeGenerated) <= TimeGenerated
  | where ResultType == "0"
  | extend ServicePrincipalName = tolower(tostring(ServicePrincipalName))
  | project ServicePrincipalId, ServicePrincipalName, SignInTime = TimeGenerated
) on ServicePrincipalName
```

```
| where ModifiedCredTime > SignInTime
| extend TimeDifferenceInMinutes= datetime_diff("Minute",SignInTime,ModifiedCredTime)
| where TimeDifferenceInMinutes >= -maxTimeBetweenSecretMgmtandSigninInMinutes
| summarize min(TimeDifferenceInMinutes) by ServicePrincipalName, ModifiedCredTime, AccountCustomEntity
| project timestamp = ModifiedCredTime, AccountCustomEntity = ServicePrincipalName, IPCustomEntity = AccountCustomEntity
```

Note: I haven't used the ServicePrincipalID for correlation to the Azure AD Audit Log because it isn't part of the data schema. The audit log includes only ObjectID and ServicePrincipalName. The disadvantage is that the attacker is able to rename the service principal to bypass this check. Therefore you should build an advanced correlation or adding the service principal rename activity in your advanced checks.

Original cover image by [mohamed Hassan / Pixabay](#)

Tags:

AzureAD

WorkloadIdentities

Categories:

Azure AD

Updated: September 19, 2020

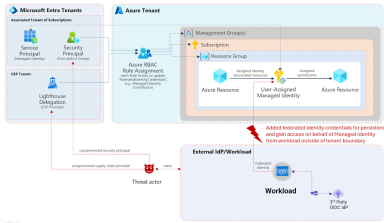
Twitter

Facebook

LinkedIn

Previous	Next
----------	------

YOU MAY ALSO ENJOY



Identify and prevent abuse of Managed Identities with Federated Credentials from unauthorized entities

August 2, 2024
13 minute read

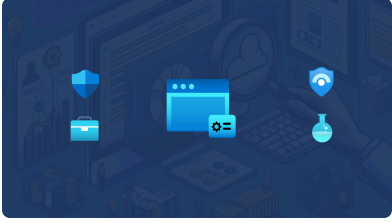
In this article, I would like to point out options to identify, monitor and avoid persistent access on Managed Identities privileges by adding federated credentials on User-Assigned Managed Identities (UAMI) from malicious or unauthorized entities. We will also have a quick look at atta...



Microsoft Entra Workload ID - Incident Response with Microsoft Sentinel Playbooks and Conditional Access

January 12, 2024
7 minute read

In the recent parts of the blog post series, we have gone through the various capabilities to detect threats and fine-tune incident enrichment of Workload Identities in Microsoft Entra. This time, we will start to automate the incident response for tackling malicious activities and thre...



Microsoft Entra Workload ID - Advanced Detections and Enrichment in Microsoft Sentinel

December 18, 2023
18 minute read

Collecting details of all workload identities in Microsoft Entra ID allows to build correlation and provide enrichment data for Security Operation Teams. In addition, it also brings new capabilities for creating custom detections. In this blog post, I will show some options on how to im...



Microsoft Entra Workload ID - Threat detection with Microsoft Defender XDR and Sentinel

December 3, 2023
13 minute read

Attack techniques has shown that service principals will be used for initial and persistent access to create a backdoor in Microsoft Entra ID. This has been used, for example as part of the NOBELIUM attack path. Abuse of privileged Workload identities for exfiltration and privilege esca...