

Custom Threat Intelligence into Azure Sentinel

Purpose:

The purpose of this document is to explain the steps involved in getting AlienVaultOTX Threat Intelligence into Azure Sentinel. Once the data is going into Sentinel it can be used with predefined analytic rules to alert when any of the TI IOCs matches something that is going on within your environment.

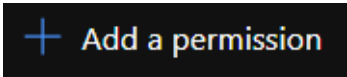
This can also be used in conjunction with Microsoft Defender ATP to block any of the IPv4/IPv6 found within the TI provided by AlienVault.

Creating Azure AD App

- Logon to Azure and navigate to Azure Active Directory
- Click App Registrations & New Registration
- Provide specific name, in this instance we used "App for sending AlienVault OTX Threat Intel to Sentinel"
- Supported Account types to use "Accounts in this organizational directory only (YourOrgName only – Single tenant)"
- Click Register

Now the app is created we need to assign rights and take note of specific information from the overview page.

Assigning Rights:

- On the left hand blade navigate to "API Permissions" and click "+ Add a permission"
- 
- A page will then pop out from the right and we need to click "Microsoft Graph" -> "Application Permissions" and then search for "Threat"
 - From here you will see "Threat Indicators" and you need to assign "ThreatIndicators.ReadWrite.OwnedBy" select that and apply permissions.

Request API permissions

< All APIs

Microsoft Graph
<https://graph.microsoft.com/> Docs

What type of permissions does your application require?

Delegated permissions
 Your application needs to access the API as the signed-in user.

Application permissions
 Your application runs as a background service or daemon without a signed-in user.

Select permissions expand all

Search: threat

Permission	Admin consent required
> ThreatAssessment	
▼ ThreatIndicators (1)	
☐ ThreatIndicators.Read.All Read all threat indicators	Yes
☒ ThreatIndicators.ReadWrite.OwnedBy Manage threat indicators this app creates or owns	Yes

Note: When the permissions have assigned it will need to be approved by a global admin. The admin will need to go into the app click the permissions you assigned and click “Grant Admin Consent for YourOrgName”

Creating secret key:

- On the left hand blade navigate to “Certificated & Secrets” and create a new “Secret”
- Please provide a relevant description in this instance we use “AlienVault OTX”
- Once created please note down the “Value” as this is the secret key.(Store in a safe place such as keepass and not plain text in your documents) We will need this “Value” later on in the process.

Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value	Secret ID
AlienVault OTX	8/19/2022	-W1*****	

Overview Page

- Take note of the Application (client) ID:
- Take note of the Directory (tenant) ID:

These will be needed later with the secret key we previously created. This is all for the app creation that we need. We will now move on to the automation (Azure Sentinel Playbook)

Azure Sentinel Playbook

- Navigate to Azure Sentinel and on the left hand blade select “Automation” -> “Playbooks”

- “Create New Playbook”
 - Subscription (Main subscription where sentinel is installed)
 - Resource Group (Sentinels Resource Group)
 - Logic App Name, in this instance we call it (AlienVaultOTX-TI)
 - Create
- Now the playbook is created we will use the template “Blank Logic App”
 - The first step will be “Recurrence” as we want this playbook to run every 24h to get new data from AlienVault.
 - Interval = 1 – Frequency = Day
 - Click the “+ New Step” (This will be the http request to GET the data from the AlienVault API) and Select “HTTP”
 - Method = GET
 - URI = <https://otx.alienvault.com/api/v1/indicators/export>
 - Headers = X-OTX-API-KEY and the box to the right paste your AlienVault OTX API Key which can be found on the AlienVault Website once you have logged in
 - Queries = modified since and in the box to the right add addDays(utcNow(),-1).
 - Next query box will be types and URL, Domain, FileHash-MD5 (this is the data we want to get from AlienVault. A list of all types can be found [here](#)) See screenshot to what the current setup should look like.

The screenshot displays a logic app configuration with two main sections: **Recurrence** and **HTTP**.

Recurrence Section:

- Interval:** 1
- Frequency:** Day
- Add new parameter:** (dropdown)

HTTP Section:

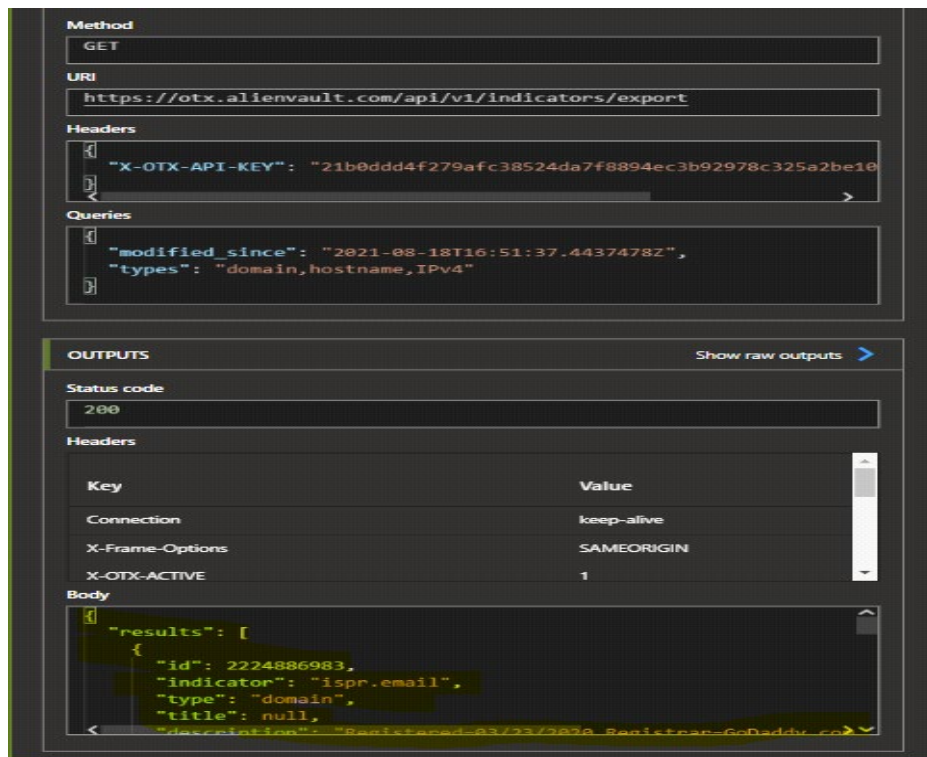
- Method:** GET
- URI:** <https://otx.alienvault.com/api/v1/indicators/export>
- Headers:**

X-OTX-API-KEY	:	
Enter key	Enter value	
- Queries:**

modified_since	<i>fx</i> addDays(...) x
types	URL, domain
Enter key	Enter value
- Body:** Enter request content
- Cookie:** Enter HTTP cookie
- Add new parameter:** (dropdown)

- Save the current logic app and click “Run Trigger” as we will need to copy the body text for the next part.

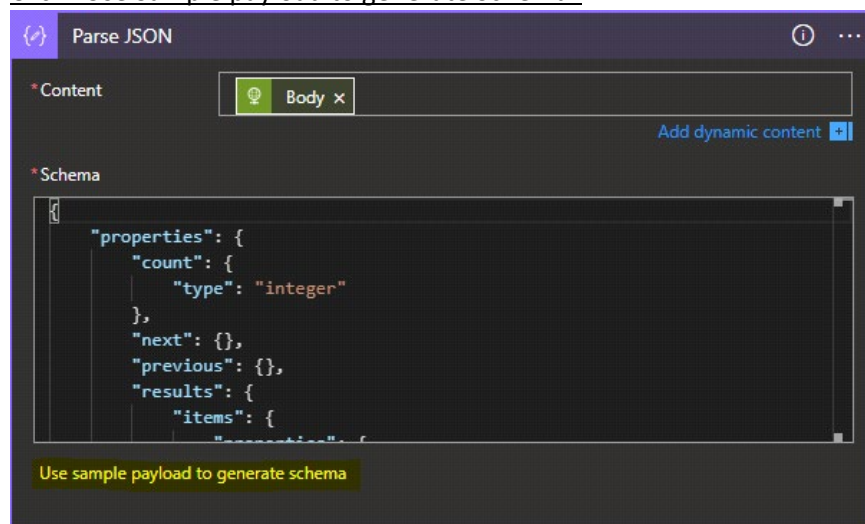
See screenshot of what need to be copied for the next part



Parse JSON

At this current stage of the playbook we should have the “Recurrence” and the “HTTP” steps complete, next we will need to parse the json from the body of the HTTP response we had when the playbook was ran,

- Add a new step and select or search for “Data Operator” -> “Parse JSON”
- Click into the “Content” and from the dynamic content select “Body”
- The schema will now need to be generated from the “Body” section we copied earlier.
 - Click “Use Sample payload to generate Schema”



- Paste the body from the HTTP response we got earlier and click okay.

NOTE: Save and run the playbook once more to check the first 3 steps are working as we intended. If so move onto the next step. If it comes back with an error “ValidationFailed. The Schema validation failed.”

Please go back to the “Parse JSON” step and look in the schema for



```

},
  "description": {
    "type": [
      "string",
      "null"
    ]
  },

```

```

"description": {
  "type": "string"

```

And change it to (Should then look like this image)

```

"description": {
  "type": [
    "string",
    "null"
  ]

```

Rerun the playbook and that should fix the issue with the scheme.

Creating the Switch

This part is where we sent the data into Azure Sentinel that’s been pulled and parsed from AlienVault. We will also need the Azure AD App details that we created and noted down earlier in the process

- New Step -> Control -> Switch
- Switch On = type
- Case = URL (Case sensitive to all types of data pulled)
 - Add an Action -> HTTP
 - Method = POST
 - URI = <https://graph.microsoft.com/beta/security/tiIndicators>
 - Headers = content-type | Application/json
 - Body =

```

{
  "action": "alert",

```

```

"activityGroupNames": [],
"confidence": 0,
"description": "OTX Threat Indicator - @{{items('For_each')}['type']}",
"expirationDateTime": "@{{addDays(utcNow(),7)}}",
"externalId": "@{{items('For_each')}['id']}",
"killChain": [],
"malwareFamilyNames": [],
"severity": 0,
"tags": [],
"targetProduct": "Azure Sentinel",
"threatType": "WatchList",
"tlpLevel": "white",
"url": "@{{items('For_each')}['indicator']}"
}

```

- Authentication type = Active Directory OAuth
- Tenant = Tenant ID Noted earlier
- Audience = <https://graph.microsoft.com>
- Client ID = ID Noted earlier
- Secret = Value Noted earlier

This step is only for the type – URL that was added in the GET request. You will need to add another case which will show up as “Case 2” and follow the same but name xxx (what was next in your “types” at step 2.

Note: The body is different per type of data being sent into sentinel. Below will be a list of threat types with the “body” part that will be used in the “Case”

BODY FOR TYPES:



Playbook body types.txt

Playbook Complete Setup:

 Recurrence

* Interval

1

* Frequency

Day

Add new parameter

 HTTP

* Method

GET

* URI

https://otx.alienvault.com/api/v1/indicators/export

Headers

X-OTX-API-KEY	:		×	📄
	:			
Enter key		Enter value		

Queries

modified_since	<i>f_x</i> addDays(...) ×	×	📄
types	URL, domain	×	
Enter key	Enter value		

Body

Enter request content

Cookie

Enter HTTP cookie

Add new parameter

 Recurrence ...

 HTTP ...

 Parse JSON ...

* Content

 Body x

* Schema

```
{
  "properties": {
    "count": {
      "type": "integer"
    },
    "next": {},
    "previous": {},
    "results": {
      "items": {
        "type": "object"
      }
    }
  }
}
```

Use sample payload to generate schema

